MX

macromedia®

DIRECTOR® MX 2004

Référence de scripting de Director

Marques

Afterburner, AppletAce, Attain, Attain Enterprise Learning System, Attain Essentials, Attain Objects for Dreamweaver, Authorware, Authorware Attain, Authorware Interactive Studio, Authorware Star, Authorware Synergy, Backstage, Backstage Designer, Backstage Desktop Studio, Backstage Enterprise Studio, Backstage Internet Studio, Contribute, Design in Motion, Director, Director Multimedia Studio, Doc Around the Clock, Dreamweaver, Dreamweaver Attain, Drumbeat, Drumbeat 2000, Extreme 3D, Fireworks, Flash, Fontographer, FreeHand, FreeHand Graphics Studio, Generator, Generator Developer's Studio, Generator Dynamic Graphics Server, Knowledge Objects, Knowledge Stream, Knowledge Track, LikeMinds, Lingo, Live Effects, MacRecorder Logo and Design, Macromedia, Macromedia Contribute, Macromedia Coursebuilder for Dreamweaver, Macromedia M Logo & Design, Macromedia Flash, Macromedia Xres, Macromind, Macromind Action, MAGIC, Mediamaker, Multimedia is the Message, Object Authoring, Power Applets, Priority Access, Roundtrip HTML, Scriptlets, SoundEdit, ShockRave, Shockmachine, Shockwave, shockwave.com, Shockwave Remote, Shockwave Internet Studio, Showcase, Tools to Power Your Ideas, Universal Media, Virtuoso, Web Design 101, Whirlwind et Xtra sont des marques de Macromedia, Inc. et peuvent être déposées aux Etats-Unis et dans d'autres pays. Les autres noms de produits, logos, graphiques, mises en page, titres, mots ou phrases mentionnés dans cette publication peuvent être des marques de commerce, des marques de service ou des noms de marque appartenant à Macromedia, Inc. ou à d'autres entités et peuvent être déposés dans certains pays, états ou provinces.

Cette publication contient des liens conduisant à des sites web qui ne sont pas sous le contrôle de Macromedia, qui n'est aucunement responsable de leur contenu. L'accès à ces sites se fait sous votre seule responsabilité. Macromedia mentionne ces liens pour référence, ce qui ne signifie pas son soutien, accord ou responsabilité quant au contenu des sites en question.

Limite de garantie et de responsabilité Apple

APPLE COMPUTER, INC. N'OFFRE AUCUNE GARANTIE, EXPRES OU IMPLICITE, CONCERNANT CE LOGICIEL, SA CAPACITE A ETRE COMMERCIALISE OU A REPONDRE A UN BESOIN PARTICULIER. L'EXCLUSION DES GARANTIES IMPLICITES EST INTERDITE PAR CERTAINS PAYS, ETATS OU PROVINCES. L'EXCLUSION ENONCEE CI-DESSUS PEUT NE PAS S'APPLIQUER A VOTRE CAS PARTICULIER. CETTE GARANTIE VOUS ASSURE DES DROITS SPECIFIQUES. D'AUTRES DROITS VARIANT D'UN PAYS A L'AUTRE PEUVENT EGALEMENT VOUS ETRE ACCORDES.

Première édition : Janvier 2004

Macromedia, Inc.
600 Townsend St.
San Francisco, CA 94103
États-Unis

TABLE DES MATIERES

CHAPITRE 1
Introduction

Ce référentiel propose des informations conceptuelles et pratiques sur le scripting dans Director MX 2004 de Macromedia ainsi que des descriptions et des exemples concernant les API (Interfaces de Programmation d'Applications) de scripting que vous utilisez pour rédiger des scripts.

Les API de scripting vous permettent d'accéder aux fonctionnalités de Director et d'augmenter le caractère interactif d'une animation. Ces API vous permettent non seulement de créer des fonctionnalités interactives identiques à celles qui correspondent aux comportements prédéfinis de Director, mais aussi d'en créer d'autres, à la fois plus puissantes et plus variées.

Les comportements prédéfinis vous permettent d'ajouter un caractère interactif de base à une animation. Il est ainsi possible de déplacer la tête de lecture sur un numéro d'image ou de repère, ou d'effectuer un zoom lorsqu'on clique sur une image-objet. Ils vous permettent également d'utiliser des fonctionnalités non interactives telles que l'animation d'images-objets, le chargement de médias et la navigation au sein des images. Les API de scripting vous permettent de développer et de personnaliser ces types de fonctionnalités.

Public visé

Ce référentiel vous concerne si vous envisagez d'effectuer une ou plusieurs des opérations suivantes :

- Développer les fonctionnalités existantes des comportements prédéfinis à l'aide de scripts.
- Ajouter des fonctionnalités à une animation avec des scripts plutôt qu'à l'aide des comportements prédéfinis.
- Ajouter des fonctionnalités plus puissantes, plus variées et plus personnalisées à une animation que celles offertes par les comportements prédéfinis.

Le but de ce référentiel est de vous fournir toutes les informations dont vous avez besoin pour ajouter de l'interactivité à vos animations à l'aide de scripts. Il n'est donc pas nécessaire d'être un programmeur chevronné pour rédiger des scripts efficaces dans Director.

Quel que soit votre niveau de familiarité avec la syntaxe de Director, Lingo ou JavaScript, consultez le Chapitre 2, *Principes de base du scripting dans Director*, page 9 et le Chapitre 3, *Rédaction de scripts dans Director*, page 55 avant de commencer à rédiger des scripts. Comme tout autre produit, Director possède ses propres conventions de scripting, types de données, etc. Pour pouvoir rédiger de bons scripts, il est essentiel que vous vous familiarisiez de façon approfondie avec ces caractéristiques de Director.

Nouveautés concernant le scripting de Director

Si vous avez rédigé des scripts avec les versions précédentes de Director, veuillez prendre note des innovations et changements importants apportés à cette nouvelle version de scripting.

Format de scripting avec syntaxe à base de points

Dans les versions précédentes de Director, il était possible de rédiger des scripts Lingo en utilisant deux types de syntaxe : la syntaxe verbose ou celle à points. La syntaxe verbose est très similaire à l'anglais, et les programmeurs débutants pouvaient l'apprendre facilement. Toutefois, les scripts avec syntaxe verbose sont vite devenus longs, complexes et difficiles à déboguer.

À présent que le modèle de scripting de Director est compatible avec les syntaxes Lingo et JavaScript, les scripts doivent en principe être rédigés uniquement avec la syntaxe à points. Beaucoup de programmeurs connaissent la syntaxe à points puisqu'elle est utilisée dans la plupart des langages orientés objet tels que Java ou C++, et dans plusieurs langages de scripting web tels que Microsoft JScript ou JavaScript.

Pour des raisons de compatibilité en amont, les scripts rédigés en utilisant la syntaxe verbose dans les versions précédentes de Director continueront à être pris en charge. Toutefois, à partir de la version Director MX 2004, la syntaxe à points est la syntaxe recommandée, et la seule qui sera prise en charge et documentée à l'avenir.

Pour plus d'informations concernant l'utilisation de la syntaxe à points dans l'écriture de scripts, consultez *Format de scripting à syntaxe à points*, page 56.

Compatibilité avec la syntaxe JavaScript

Director est désormais compatible avec la syntaxe JavaScript. Cela signifie que vous pouvez non seulement créer et déboguer des scripts à l'aide de Lingo (le langage de scripting habituel de Director) mais aussi choisir de créer et déboguer des scripts en utilisant la syntaxe JavaScript. La version de JavaScript utilisée dans Director est appelée *syntaxe JavaScript* dans l'ensemble de ce référentiel.

La prise en charge de la syntaxe JavaScript dans Director permet de créer un environnement de scripting familier non seulement aux développeurs en JavaScript mais aussi à ceux habitués à travailler avec les outils Macromedia Flash ActionScript, Flash Communication Server, Dreamweaver, Authorware, ainsi qu'un certain nombre d'autres environnements.

La version de syntaxe JavaScript utilisée dans Director est JavaSript 1.5, ce qui signifie qu'elle est entièrement compatible avec le standard ECMAScript Language Specification ECMA-262, 3ème Édition. Pratiquement toutes les fonctions et fonctionnalités de JavaScript 1.5 sont désormais disponibles dans Director.

Pour plus d'informations concernant le choix de la syntaxe Lingo ou JavaScript, reportez-vous à la section *Choisir entre Lingo et la syntaxe JavaScript*, page 55.

Remarque : Ce référentiel se contente d'aborder les principales fonctions et fonctionnalités de la version JavaScript 1.5 nécessaires à la rédaction de scripts utilisant la syntaxe JavaScript dans Director. Il ne constitue pas un référentiel complet de JavaScript 1.5. Pour une information plus détaillée sur JavaScript 1.5, consultez l'un des nombreux documents externes consacrés à ce sujet.

Nouvelles manières d'accéder aux API de scripting

Dans cette version de Director, les API de scripting ont été groupées par objets, et vous pouvez y accéder par le biais de ces objets. Les objets en question comportent les fonctionnalités requises pour ajouter de l'interactivité à vos animations et sont accessibles aux syntaxes Lingo et JavaScript au sein de Director, des projections et de l'outil Macromedia Shockwave Player.

Même si la méthode pour y accéder peut être différente, vous avez accès aux mêmes API que dans les versions précédentes, sans compter les toutes nouvelles API. La seule différence, c'est que vous y accédez en utilisant les nouveaux objets.

Pour plus d'informations sur les objets et les API de scripting correspondantes, consultez *Introduction aux objets de Director*, page 57.

Nouveautés dans cette documentation

Si vous avez appris à rédiger des scripts dans les versions précédentes de Director, veuillez prendre note des changements apportés à la documentation de scripting dans cette nouvelle version. La *Référence de scripting de Director* remplace le *Dictionnaire Lingo* accompagnant les versions précédentes de Director. L'organisation de ce référentiel est différente de celle du *Dictionnaire Lingo*.

Dans le *Dictionnaire Lingo*, les informations concernant le modèle de scripting étaient classées par fonction. Par exemple, pour apprendre à utiliser les images-objets dans un script, vous deviez consulter l'une des sections se trouvant sous l'en-tête Images-objets, telles que Glissement d'images-objets, Dimensions des images-objets, etc. De plus, toutes les API de scripting étaient répertoriées par ordre alphabétique dans une seule liste, et par conséquent les fonctions, propriétés, événements, etc. figuraient tous dans la même liste.

Dans la *Référence de scripting de Director*, les informations concernant le modèle de scripting sont classées par objet. Cette classification reflète fidèlement l'organisation des objets de scripting que vous utilisez dans vos scripts. Par exemple, pour apprendre à utiliser les images-objets dans un script, vous pouvez consulter la section *Images-objets* du chapitre *Objets principaux de Director*.

Les API de scripting sont toujours répertoriées par ordre alphabétique, mais sont classées par type d'API. Par exemple, les méthodes sont toutes répertoriées par ordre alphabétique sous l'en-tête Méthodes, les propriétés sont toutes classées par ordre alphabétique sous l'en-tête Propriétés, etc.

Recherche d'informations en matière de scripting dans Director

La nouvelle *Référence de scripting de Director* est organisé comme suit :

Principes de base du scripting dans Director fournit des informations sur les concepts et composants de base que vous utilisez lorsque vous rédigez des scripts dans Director.

Rédaction de scripts dans Director fournit des informations sur l'environnement de scripting de Director, en plus de celles concernant les concepts et techniques de scripting avancés.

Débogage de scripts dans Director fournit des informations sur la manière de résoudre les problèmes liés à vos scripts.

Objets principaux de Director fournit une liste des objets et API utilisés pour accéder aux principales fonctionnalités et fonctions de Director, telles que le moteur du lecteur de Director, les fenêtres d'animation, les images-objets, les sons, etc.

Types de médias fournit une liste des types de médias et API que vous utilisez pour accéder aux fonctionnalités des divers types de médias (RealMedia, DVD, GIF animé, etc.) et qui sont ajoutés aux animations en tant qu'acteurs.

Objets de scripting fournit une liste d'objets de scripting, aussi appelés Xtras, et d'API utilisées pour étendre les fonctionnalités de base de Director. Les Xtras permettent certaines actions telles que l'importation de filtres et la connexion à Internet.

Objets 3D fournit une liste des objets utilisés pour ajouter des fonctionnalités 3D à une animation.

Constantes fournit une liste des constantes disponibles dans Director.

Evénements et messages fournit une liste des événements disponibles dans Director.

Mots-clés fournit une liste des mots clés disponibles dans Director.

Méthodes fournit une liste des méthodes disponibles dans Director.

Opérateurs fournit une liste des opérateurs disponibles dans Director.

Propriétés fournit une liste des propriétés disponibles dans Director.

CHAPITRE 2
Principes de base du scripting dans Director

Si vous êtes novice en matière de programmation avec Macromedia Director MX 2004, prenez le temps d'apprendre les principaux concepts de scripting qui vous aideront à créer des scripts dans Director avant de commencer. Certains de ces principes de base comprennent des définitions de termes importants, les règles de syntaxe, les types de données disponibles et des informations sur les éléments de base de scripting dans Director, tels que les variables, les tableaux et les opérateurs.

Types de scripts

Une animation Director peut contenir quatre types de scripts : les comportements, les scripts d'animation, les scripts parents et les scripts associés aux acteurs. Les comportements, les scripts d'animation et les scripts parents figurent tous dans la fenêtre Distribution sous la forme d'acteurs indépendants. Un script associé à un acteur figure dans la fenêtre Distribution et n'apparaît pas indépendamment.

- Les comportements sont des scripts associés à des images-objets ou images dans le scénario et sont appelés comportements d'image-objet ou comportements d'image. La miniature de la fenêtre Distribution de chaque comportement contient une icône de comportement dans l'angle inférieur droit.

  Lorsqu'il est utilisé dans la *Référence de scripting de Director*, le terme *comportement* se rapporte à un script associé à une image-objet ou une image. Cette définition diffère de celle des comportements figurant dans la Palette des bibliothèques de Director. Pour plus d'informations sur ces comportements de Director, consultez les rubriques Utilisation de Director du centre de support de Director.

  Tous les comportements ajoutés à une bibliothèque de distribution figurent dans le menu local Comportements de l'inspecteur de comportement. Les autres types de scripts ne figurent pas ici.

  Vous pouvez placer le même comportement à plusieurs endroits du scénario. Lorsque vous modifiez un comportement, la nouvelle version de ce comportement est appliquée à tous les endroits auxquels il est associé dans le scénario.
- Les scripts d'animations contiennent des gestionnaires disponibles globalement ou au niveau d'une animation. Les gestionnaires d'événement d'un script d'animation peuvent être appelés depuis n'importe quel script de l'animation pendant la lecture.

  Une icône de script d'animation apparaît dans l'angle inférieur droit de la miniature correspondante dans la fenêtre Distribution.

Les scripts d'animation sont disponibles pour l'animation entière, quelle que soit l'image où se trouve la tête de lecture ou l'image-objet que l'utilisateur manipule. Lors de la lecture d'une animation dans une fenêtre ou comme animation liée, un script d'animation n'est disponible que pour sa propre animation.

- Les scripts parents sont des scripts spéciaux contenant les éléments Lingo utilisés pour créer des objets enfants. Vous pouvez utiliser des scripts parents pour générer des objets scripts qui ont une réponse et un comportement similaires tout en se comportant indépendamment les uns des autres. Une icône de script parent apparaît dans l'angle inférieur droit de la miniature correspondante dans la fenêtre Distribution.

  Pour obtenir des informations sur l'utilisation des scripts parents et des objets enfants, consultez *Programmation orientée objet avec Lingo*, page 61.

  La syntaxe JavaScript n'utilise pas de scripts parents ou d'objets enfants. Elle utilise des techniques de programmation orientée objet de style JavaScript normales. Pour plus d'informations sur la programmation orientée objet dans la syntaxe JavaScript, consultez *Programmation orientée objet avec la syntaxe JavaScript*, page 72.
- Les scripts associés aux acteurs sont des scripts directement associés à un acteur, indépendamment du scénario. Lorsque ces acteurs sont affectés à des images-objets, leurs scripts sont mis à la disposition de ces dernières.

  Contrairement aux comportements, aux scripts parents et aux scripts d'animation, les scripts d'acteurs n'apparaissent pas dans la fenêtre Distribution. Cependant, si l'option Afficher les icônes de script des acteurs est sélectionnée dans la boîte de dialogue Préférences de la fenêtre Distribution, les acteurs auxquels sont associés des scripts sont identifiés par une petite icône de script dans l'angle inférieur gauche de leur miniature dans la fenêtre Distribution.

Terminologie de scripting

La syntaxe Lingo et la syntaxe JavaScript utilisent toutes deux des termes qui sont propres à chaque langage, en plus de certains termes qui sont partagés par ces langages.

Les termes importants sont présentés ici par ordre alphabétique. Ces termes sont couramment utilisés dans la *Référence de scripting de Director*, et il est donc nécesaire de bien les assimiler avant de continuer.

- Les constantes sont des éléments dont les valeurs ne changent jamais. Par exemple, en Lingo, les constantes telles que `TAB`, `EMPTY` et `RETURN` ont toujours les mêmes valeurs et ne peuvent pas être modifiées. Dans la syntaxe JavaScript, les constantes telles que `Math.PI` et `Number.MAX_VALUE` ont toujours les mêmes valeurs et ne peuvent pas être modifiées. Vous pouvez également créer vos propres constantes dans la syntaxe JavaScript à l'aide du mot clé `const`.

  Pour plus d'informations sur les constantes, consultez *Constantes*, page 21.
- Les événements sont des actions qui se produisent pendant la lecture d'une animation. Les événements se produisent à l'arrêt d'une animation, au démarrage d'une image-objet, à l'entrée de la tête de lecture dans une image, lors de l'emploi du clavier par l'utilisateur, etc. Les événements de Director sont tous prédéfinis et ont toujours le même sens.

  Pour plus d'informations sur les événements, consultez *Evénements*, page 36.

- Les expressions sont des parties d'instruction produisant une valeur. Par exemple, `2 + 2` est une expression.
- Les fonctions font référence à des fonctions de haut niveau ou à des types spécifiques de codes de la syntaxe JavaScript.

  Une fonction de haut niveau ordonne à une animation de faire quelque chose pendant la lecture de l'animation ou renvoie une valeur, mais elle n'est pas appelée à partir d'un objet spécifique. Par exemple, vous appelez la fonction de haut niveau `list()` en utilisant la syntaxe `list()`. Comme c'est le cas pour une fonction, une méthode ordonne également à une animation de faire quelque chose pendant la lecture de l'animation ou renvoie une valeur, mais elle est toujours appelée à partir d'un objet.

  En JavaScript, une fonction représente un gestionnaire d'événement, un objet personnalisé, une méthode personnalisée, etc. L'utilisation des fonctions JavaScripts dans des cas pareils est expliquée dans les rubriques appropriées, plus loin dans cette référentiel.
- Les gestionnaires ou gestionnaires d'événement sont des ensembles d'instructions placées dans un script et exécutées en réponse à un événement déterminé et à un message subséquent. Lorsqu'un événement se produit, Director génère et envoie un message correspondant aux scripts et un gestionnaire approprié est exécuté en réponse au message. Les noms des gestionnaires sont toujours les mêmes que les événements et messages auxquels ils répondent.

  Remarque : Bien qu'un événement soit géré par une fonction dans la syntaxe JavaScript, le terme *gestionnaire* désigne, dans ce référentiel, à la fois les gestionnaires Lingo et les fonctions de la syntaxe JavaScript qui gèrent les événements.

  Pour plus d'informations sur les gestionnaires, consultez *Gestionnaires*, page 38.
- Les instructions sont des instructions valides que Director peut exécuter. Les scripts sont composés d'ensembles d'instructions. L'instruction Lingo suivante est une instruction complète.

  ```
  _movie.go("Auteur")
  ```

  Pour plus d'informations sur la rédaction d'instructions, consultez *Format de scripting à syntaxe à points*, page 56.
- Les listes (Lingo) ou tableaux (syntaxe JavaScript) sont des ensembles ordonnés de valeurs qui permettent le suivi et la mise à jour d'un ensemble de données, comme une série de noms ou de valeurs affectés à un ensemble de variables. Un exemple simple de liste est une liste de nombres tels que `[1, 4, 2]`.

  Pour plus d'informations sur l'utilisation des listes en Lingo et dans la syntaxe JavaScript, consultez *Listes linéaires et listes de propriétés*, page 42.

  Pour plus d'informations sur l'utilisation des tableaux de la syntaxe JavaScript, consultez *Tableaux de la syntaxe JavaScript*, page 50.
- Les messages sont des avertissements envoyés aux scripts par Director lorsque des événements déterminés se produisent dans une animation. Par exemple, lorsque la tête de lecture entre dans une image donnée, l'événement `enterFrame` se produit et Director envoie un message `enterFrame`. Si un script contient un gestionnaire `enterFrame`, les instructions de ce gestionnaire seront exécutées puisque le gestionnaire a reçu le message `enterFrame`. Si aucun script ne contient de gestionnaire de message, le message est ignoré dans le script.

  Pour plus d'informations sur les messages, consultez *Messages*, page 37.

- Les méthodes sont des termes entraînant une action pendant la lecture de l'animation ou le renvoi d'une valeur, et sont appelées à partir d'un objet. Par exemple, vous appelez la méthode `insertFrame()` de l'objet Animation en utilisant la syntaxe `_movie.insertFrame()`. Bien que leurs fonctionnalités soient similaires aux fonctions de haut niveau, les méthodes sont toujours appelées à partir d'un objet, contrairement aux fonctions de haut niveau.
- Mots-clés : mots réservés ayant un sens particulier. Par exemple, en Lingo, le mot clé `end` indique la fin d'un gestionnaire. Dans la syntaxe JavaScript, le mot clé `var` indique que le terme suivant est une variable.
- Les opérateurs calculent une nouvelle valeur à partir d'une ou de plusieurs valeurs. Par exemple, l'opérateur d'addition (+) additionne deux valeurs ou plus pour produire une nouvelle valeur.

  Pour plus d'informations sur les opérateurs, consultez *Opérateurs*, page 27.
- Les paramètres sont des repères permettant de transmettre des valeurs aux scripts. Les paramètres s'appliquent aux méthodes et gestionnaires d'événement, mais pas aux propriétés. Ils sont exigés par certaines méthodes, mais ne sont pas obligatoires pour d'autres.

  Par exemple, la méthode `go()` de l'objet Animation envoie la tête de lecture à une image précise, et indique éventuellement le nom de l'animation dans laquelle se trouve l'image. Pour effectuer cette tâche, la méthode `go()` nécessite au moins un paramètre et en accepte un autre. Le premier paramètre requis précise l'image à laquelle doit être envoyée la tête de lecture, le second paramètre facultatif indiquant l'animation dans laquelle se trouve l'image. Vu que le premier paramètre est obligatoire, une erreur de script sera renvoyée s'il n'est pas présent au moment où la méthode `go()` est invoquée. Étant donné que le second paramètre est facultatif, la méthode effectuera sa tâche même si ce paramètre n'est pas présent.
- Les propriétés sont des attributs définissant un objet. Par exemple, l'image-objet d'une animation possède des attributs précis tels que sa largeur, sa hauteur, la couleur de son arrière-plan, etc. Pour accéder aux valeurs de ces trois attributs, vous devez utiliser les propriétés `width`, `height` et `backColor` de l'objet de l'image-objet.

  Pour plus d'informations sur l'attribution de propriétés aux variables, consultez *Stockage et mise à jour de valeurs dans des variables*, page 22.
- Les variables sont des éléments servant à stocker et à mettre à jour des valeurs. Les variables doivent commencer par une lettre, un trait de soulignement (_) ou le signe dollar ($). Les caractères suivants du nom d'une variable peuvent être des chiffres (0-9). Pour attribuer des valeurs aux variables ou changer les valeurs de plusieurs propriétés, utilisez l'opérateur égal à (=).

  Pour plus d'informations sur l'utilisation des variables, consultez *Variables*, page 22.

Syntaxe de scripting

Les règles générales suivantes s'appliquent à la syntaxe Lingo et JavaScript.

- Les repères de commentaires varient de Lingo à la syntaxe JavaScript.

  Les commentaires Lingo sont tous précédés de doubles tirets (`--`). Chaque ligne d'un commentaire constitué de plusieurs lignes doit être précédée de doubles tirets.

  ```
  -- Ceci est un commentaire Lingo à une seule ligne.

  -- Ceci est un
  -- commentaire Lingo à plusieurs lignes.
  ```

  Les commentaires d'une seule ligne utilisant la syntaxe JavaScript sont précédés de deux barres obliques (`//`). Les commentaires à plusieurs lignes sont précédés du signe `/*` suivi du signe `*/`.

  ```
  // Ceci est un commentaire JavaScript d'une seule ligne.

  /* Ceci est un
  commentaire JavaScript de plusieurs lignes.*/
  ```

  Vous pouvez placer un commentaire sur une ligne séparée ou après une instruction. Le texte qui suit les repères de commentaires sur la même ligne sera ignoré.

  Les commentaires peuvent être variés : il peut s'agir de notes concernant un script ou d'un gestionnaire particulier ou d'une instruction dont l'objectif n'est pas évident. Ces commentaires vous permettent de vous familiariser avec une procédure que vous n'avez pas utilisée depuis longtemps.

  L'ajout d'un grand nombre de commentaires n'augmente pas la taille de votre fichier d'animation lorsqu'il est enregistré sous forme d'un fichier compressé DCR ou DXR. Les commentaires seront supprimés du fichier lors du processus de décompression.

  Vous pouvez également utiliser les repères de commentaires pour ignorer les sections de code que vous souhaitez désactiver à des fins de test ou de débogage. En ajoutant des repères de commentaires à votre code au lieu de le supprimer, vous le transformez provisoirement en commentaires. Sélectionnez le code à activer ou à désactiver et utilisez les boutons Insérer une marque de commentaire ou Supprimer la marque de commentaire de la fenêtre Script pour ajouter ou retirer rapidement les repères de commentaires.

- Les parenthèses sont nécessaires après chaque nom de méthode et de fonction. Par exemple, lorsque vous appelez la méthode `beep()` de l'objet Son, vous devez insérer des parenthèses après le mot beep. Autrement, une erreur de script se produira.

  ```
  // Syntaxe JavaScript
  _sound.beep(); // cette instruction fonctionnera correctement
  _sound.beep; // cette instruction entraînera une erreur de script
  ```

  Lorsque vous appelez une méthode, une fonction ou un gestionnaire à partir d'une autre méthode ou fonction ou d'un autre gestionnaire, vous devez inclure des parenthèses dans l'instruction d'appel. Dans l'exemple suivant, la méthode `modifierImage-objet()` contient un appel de gestionnaire `image-objetCliquée`. Si l'appel du gestionnaire `image-objetCliquée` ne contient pas de parenthèses, vous obtenez une erreur de script.

```
// Syntaxe JavaScript
function modifierImage-objet() {
  image-objetCliquée(); // cet appel du gestionnaire fonctionnera
  correctement
  image-objetCliquée; // cet appel du gestionnaire entraînera une erreur de
  script
}

function image-objetCliquée() {
  // code gestionnaire ici
}
```

Vous pouvez également utiliser des parenthèses pour ignorer l'ordre de priorité dans les opérations mathématiques ou pour simplifier la lecture de vos instructions. Par exemple, la première expression mathématique ci-dessous renverra le résultat 13, tandis que la seconde expression renverra le résultat 5 :

```
5 * 3 - 2 -- renvoie 13
5 * (3 - 2) -- renvoie 5
```

- La syntaxe des gestionnaires d'événements varie de Lingo à JavaScript. En Lingo, les gestionnaires utilisent la syntaxe `on nomDeGestionnaire`. En JavaScript, les gestionnaires sont considérés comme des fonctions et utilisent la syntaxe `function nomDeGestionnaire()`. Par exemple, les instructions suivantes contiennent un gestionnaire qui émet un bip lorsque l'utilisateur clique sur la souris :

```
--Syntaxe Lingo
on mouseDown
  _sound.beep()
end

// Syntaxe JavaScript
function mouseDown() {
  _sound.beep();
}
```

- La syntaxe des paramètres des gestionnaires d'événements peut varier de Lingo à la syntaxe JavaScript. Lingo et JavaScript supportent tous deux les paramètres passés à un gestionnaire et mis entre parenthèses. Si plusieurs paramètres sont passés, chaque paramètre est séparé par une virgule. En Lingo, vous pouvez également passer des paramètres qui ne sont pas mis entre parenthèses. Par exemple, le gestionnaire `additionParam` suivant reçoit les deux paramètres `a` et `b`.

```
--Syntaxe Lingo
on additionParam a, b -- sans parenthèses
  c = a + b
end

on additionParam(a, b) -- avec parenthèses
  c = a + b
end

// Syntaxe JavaScript
function additionParam(a, b) {
  c = a + b;
}
```

- Le mot-clé `const` peut être utilisé dans la syntaxe JavaScript pour préciser une constante dont la valeur ne change pas. Lingo possède son propre ensemble de constantes prédéfinies (`TAB`, `EMPTY`, etc.) ; le mot-clé `const` ne s'applique donc pas à Lingo.

  Par exemple, l'instruction suivante indique une constante appelée `intAuthors` et définit sa valeur comme étant 12. Cette valeur sera toujours 12 et ne pourra pas être modifiée dans le script.

  ```
  // Syntaxe JavaScript
  const intAuthors = 12;
  ```

- Le mot-clé `var` dans la syntaxe JavaScript peut être placé devant un mot pour préciser que ce terme est une variable. L'instruction suivante crée une variable appelée `valeurDébut`.

  ```
  // Syntaxe JavaScript
  var valeurDébut = 0;
  ```

  Remarque : Bien que l'utilisation de `var` dans la syntaxe JavaScript soit facultative, il est recommandé de toujours déclarer les variables JavaScript locales ou celles au sein d'une fonction à l'aide de `var`. Pour plus d'informations sur l'utilisation des variables, consultez *Variables*, page 22.

- Le symbole de continuation (\) de Lingo indique qu'une longue ligne d'exemples de codes s'étale sur deux ou plusieurs lignes. Les lignes Lingo divisées de cette manière ne représentent pas des lignes de codes séparées. Par exemple, le code suivant sera valide.

  ```
  --Syntaxe Lingo
  tTexture = member("3D").model("Boîte") \
    .shader.texture
  ```

  La syntaxe JavaScript ne comporte pas de symbole de continuation de ligne. Pour diviser une longue chaîne de codes JavaScript en plusieurs lignes, il vous suffit d'ajouter un retour à la ligne à la fin d'une ligne.

- Le point-virgule peut être utilisé pour indiquer la fin d'une instruction en code JavaScript. Le point-virgule ne s'applique pas à Lingo.

  L'utilisation du point-virgule est facultative. Lorsqu'il est utilisé, il doit être placé à la fin d'une instruction complète. Par exemple, les deux instructions suivantes créent une variable appelée `valeurDébut`.

  ```
  // Syntaxe JavaScript
  var valeurDébut = 0
  var valeurDébut = 0;
  ```

  Un point-virgulevirgule n'indique pas nécessairement la fin d'une ligne de codes JavaScript, et plusieurs instructions peuvent être placées sur une ligne. Toutefois, et pour des raisons de clarté, il est conseillé de placer des instructions sur des lignes séparées. Par exemple, les trois instructions suivantes occupent une seule ligne de codes et sont valides, mais la lecture des codes n'est pas aisée.

  ```
  // Syntaxe JavaScript
  _movie.go("Auteur"); var valeurDébut = 0; _sound.beep();
  ```

- Les espaces entre les caractères au sein des expressions et instructions sont ignorés en Lingo et en JavaScript. Dans les chaînes de caractères entre guillemets droits, les espaces sont considérés comme des caractères. Si vous souhaitez insérer des espaces dans une chaîne, vous devez les y placer explicitement. Par exemple, la première instruction ci-dessous ignore les espaces entre les éléments de la liste, et la seconde inclut les espaces.

  ```
  --Syntaxe Lingo
  maListe1 = ["1",      "2",      "3"] -- renvoie ["1", "2", "3"]
  maListe2 = ["  1  ", "  2  ", "  3  "] -- renvoie ["  1  ", "  2  ", "  3  "]
  ```

- La sensibilité à la casse varie de Lingo à la syntaxe JavaScript.

  Lingo ne fait pas la différence entre les majuscules et les minuscules, ce qui vous permet d'utiliser les majuscules et les minuscules comme bon vous semble. Par exemple, les quatre instructions suivantes sont équivalentes :

  ```
  --Syntaxe Lingo
  member("Chat").hilite = true
  member("chat").hiLite = True
  MEMBER("CHAT").HILITE = TRUE
  Member("Chat").Hilite = true
  ```

  Bien que Lingo ne soit pas sensible à la casse, il est recommandé de choisir une casse et de l'utiliser de manière cohérente dans les scripts. Cette habitude permet d'identifier plus facilement les noms de gestionnaires, variables, acteurs, etc.

  La syntaxe JavaScript est sensible à la casse lorsqu'elle fait référence à des objets, propriétés de haut niveau ou méthodes liées aux variables définies par l'utilisateur. Par exemple, la méthode de haut niveau `sprite()` renvoie un référence à une image-objet précise, et est implementée dans Director avec des lettres minuscules. La première instruction ci-dessous se rapporte au nom de la première image-objet d'une animation, tandis que les deux instructions suivantes entraînent une erreur de script.

  ```
  // Syntaxe JavaScript
  sprite(1).name // Cette instruction fonctionne normalement
  Sprite(1).name // Cette instruction entraîne une erreur de script
  SPRITE(1).name // Cette instruction entraîne une erreur de script
  ```

  Les chaînes littérales sont toujours sensibles à la casse en Lingo et en JavaScript.

  Pour plus d'informations sur l'utilisation des chaînes, consultez *Chaînes*, page 19.

Types de données

Un type de données est un ensemble de données dont les valeurs possèdent des caractéristiques similaires et prédéfinies. Chaque valeur de variable et de propriété dans Directeur est d'un type de données précis, et les valeurs renvoyées par les méthodes sont d'un type de données précis.

Par exemple, considérons les deux instructions suivantes. Dans la première instruction, il a été attribué à la variable `entX` le nombre entier 14. La variable `entX` est donc du type de données nombre entier. Dans la seconde instruction, il a été attribué à la variable `chaîneX` une séquence de valeurs de caractères, c'est-à-dire une chaîne. La variable `chaîneX` est donc du type de données chaîne.

```
--Syntaxe Lingo
entX = 14
chaîneX = "Nouvelles du jour"

// Syntaxe JavaScript
var entX = 14;
var chaîneX = "Nouvelles du jour";
```

Les valeurs renvoyées par les méthodes ou fonctions appartiennent également à un type de données. Par exemple, la méthode `windowPresent()` de l'objet Lecteur renvoie une valeur indiquant si une fenêtre est présente. La valeur renvoyée est `TRUE` (1) ou `FALSE` (0).

Certains types de données sont partagés par Lingo et JavaScript, et d'autres se rapportent à l'un ou l'autre langage. L'ensemble de types de données pris en charge par Director est fixe et ne peut être modifié, ce qui signifie qu'il n'est pas possible d'ajouter de nouveaux types de données ou de supprimer les types de données existants. Director supporte les types de données suivants.

Type de données	Description
# (symbole)	Unité autonome pouvant représenter une condition ou un indicateur. Par exemple, `#list` ou `#word`.
Array	(Syntaxe JavaScript seulement) Bien qu'il ne s'agisse pas exactement d'un type de données, un objet Array (tableau) peut être utilisé sur des listes linéaires de valeurs. La fonctionnalité d'un objet Array est identique à celle du type de données Liste de Lingo.
Boolean	Une valeur `TRUE` (1) ou `FALSE` (0). En Lingo, toutes les valeurs `TRUE` ou `FALSE` sont de simples constantes de nombres entiers, 1 pour `TRUE`, 0 pour `FALSE`. Dans la syntaxe JavaScript, toutes les valeurs `true` ou `false` sont, par défaut, les vraies valeurs booléennes`true` ou `false`, mais sont automatiquement converties, le cas échéant, en simples constantes de nombres entiers dans Director. En Lingo, `TRUE` et `FALSE` peuvent être en majuscules ou minuscules. Dans la syntaxe JavaScript, `true` et `false` doivent toujours être en minuscules.
Color	Représente la couleur d'un objet.
Constant	Une donnée dont la valeur ne change pas.
Date	Bien qu'il ne s'agisse pas exactement d'un type de données, un objet Date peut être utilisé avec les dates en JavaScript. En Lingo, utilisez la méthode date() pour travailler avec les dates.
Float	(Lingo seulement) Nombre à virgule flottante. Par exemple, 2,345 ou 45,43.
Function	(Syntaxe JavaScript seulement) Bien qu'il ne s'agisse pas exactement d'un type de données, un objet Function peut être utilisé pour préciser une chaîne ou un code à exécuter.
Integer	(Lingo seulement) Nombre entier. Par exemple, 5 ou 298.
List	Une liste linéaire ou de propriétés composée respectivement de noms de valeurs ou de propriétés, et de valeurs.
Math	(Syntaxe JavaScript seulement) Bien qu'il ne s'agisse pas exactement d'un type de données, un objet Math peut être utilisé pour effectuer des opérations mathématiques.
null	(Syntaxe JavaScript seulement) Indique une variable dont la valeur correspond à 0 dans les contextes numériques et à `FALSE` dans les contextes booléens.
Number	(Syntaxe JavaScript seulement) Bien qu'il ne s'agisse pas exactement d'un type de données, un objet Number peut être utilisé pour représenter des constantes numériques telles qu'une valeur maximale, une valeur not-a-number (NaN) et une infinité.

Type de données	Description
Object	Bien qu'il ne s'agisse pas exactement d'un type de données, un objet Object peut être utilisé pour créer un conteneur personnalisé de données et des méthodes agissant sur les données.
Point	Point sur la scène ayant une coordonnée à la fois horizontale et verticale.
Rect	Rectangle sur la scène.
RegExp	(JavaScript seulement) Modèle d'expression régulier correspondant à des combinaisons de caractères dans des chaînes.
String	Chaîne de symboles clavier ou de valeurs de caractères. Par exemple, "Director" ou "$5.00".
undefined	(Syntaxe JavaScript seulement) Indique une variable qui n'a pas de valeur.
Vector	Point dans l'espace 3D.
VOID	(Lingo seulement) Indique une valeur vide.

Remarque : Plusieurs types de données et d'objets de la syntaxe JavaScript possèdent leur propre ensemble de méthodes et propriétés qui peuvent être utilisées pour manipuler ces types. Bien que la *Référence de scripting de Director* donne parfois des informations sur certaines de ces méthodes et propriétés, ces informations ne sont pas complètes. Pour plus d'informations sur ces types de données et objets, et leur méthodes et propriétés, consultez l'une des nombreuses ressources sur le sujet.

Les propriétés intégrées de Director, telles que la propriété `name` de l'objet Acteur, ne peuvent recevoir que des valeurs qui sont du même type de données que celui de la propriété. Par exemple, le type de données de propriété du `name` de l'objet Auteur est une chaîne, et la valeur doit donc être une chaîne telle que `Nouvelle du jour`. Si vous essayez d'attribuer une valeur de type de données différent à cette propriété, telle que le nombre entier 20, vous obtenez une erreur de script.

Si vous créez vos propres propriétés, leurs valeurs peuvent être de n'importe quel type de données, quel que soit le type de données de la valeur initiale.

Les syntaxes Lingo et JavaScript sont toutes deux dynamiques. Cela signifie que vous n'avez pas à préciser le type de données d'une variable lorsque vous la déclarez et que les types de données sont automatiquement convertis lors de l'exécution d'un script.

Par exemple, la syntaxe JavaScript suivante définit la variable `monAnimation` comme étant un nombre entier, et plus tard, est définie comme étant une chaîne. Lors de l'exécution du script, le type de données de `monAnimation` est converti automatiquement.

```
--Syntaxe Lingo
monAnimation = 15 -- monAnimation est défini, au départ, en tant que nombre
  entier
...
monAnimation =  "Animations" -- monAnimation est défini, par la suite, en tant
  que chaîne

// Syntaxe JavaScript
var monAnimation = 15; // monAnimation est défini, au départ, en tant que
  nombre entier
...
monAnimation =  "Animations " -- monAnimation est défini, par la suite, en tant
  que chaîne
```

Valeurs littérales

Une valeur littérale est une partie d'instruction ou d'expression qui doit être traitée telle quelle, et non comme une variable ou un élément de script. Les valeurs littérales que vous pourrez rencontrer dans un script sont les chaînes de caractères, les entiers, les nombres décimaux, les noms et numéros d'acteurs, les noms et numéros d'images et d'animations, les symboles et les constantes.

Chaque type de valeur littérale est régi par ses propres règles.

Chaînes

Les chaînes sont des mots ou des groupes de caractères que le script traite en tant que mots standard, et non en tant que variables. Elles doivent être encadrées de guillemets droits. Par exemple, vous pouvez utiliser des chaînes pour transmettre des messages aux utilisateurs de votre animation ou pour attribuer des noms aux acteurs. Dans l'instruction suivante, `Bonjour` et `Salutations` sont des chaînes. `Bonjour` est le texte littéral placé dans l'acteur texte et `Salutations` est le nom de ce dernier.

```
--Syntaxe Lingo
member("Salutations").text = "Bonjour"
```

De même, lorsque vous testez une chaîne, vous devez l'encadrer de guillemets droits, comme dans l'exemple suivant :

```
--Syntaxe Lingo
if "Bonjour M. Dupont" contains "Bonjour" then gestionnaireAudio
```

Lingo et JavaScript considèrent les espaces figurant au début ou à la fin d'une chaîne comme une partie littérale de la chaîne. L'expression suivante comprend un espace après le mot *à* :

```
// Syntaxe JavaScript
trace("Mes pensées se résument à ");
```

Bien que Lingo ne distingue pas les majuscules des minuscules lorsqu'il fait référence aux acteurs, aux variables, etc., les chaînes littérales sont sensibles à la casse. Par exemple, les deux instructions suivantes placent un texte différent dans l'acteur indiqué, car `Bonjour` et `BONJOUR` sont des chaînes littérales.

```
--Syntaxe Lingo
member("Salutations").text = "Bonjour"
member("Salutations").text = "BONJOUR"
```

En Lingo, la fonction `string()` peut convertir une valeur numérique en une chaîne. Dans la syntaxe JavaScript, la méthode `toString()` peut convertir une valeur numérique en une chaîne.

Remarque : Si vous essayez d'utiliser la méthode `toString()` de la syntaxe JavaScript sur une valeur `null` ou `undefined`, vous obtenez une erreur de script. Ceci n'est pas le cas en Lingo, dont la fonction `string()` s'applique à toutes les valeurs, y compris celles qui sont `VOID`.

Nombres

Il existe deux types de nombres en Lingo : entiers et décimaux.

Un nombre entier ne comporte ni fraction ni décimale, dans les plages -2 147 483 648 et +2 147 483 647. Entrez des nombres entiers sans utiliser de virgule. Utilisez le signe moins (`-`) pour les nombres négatifs.

Un nombre décimal, également appelé nombre à virgule flottante, est un nombre qui inclut une virgule décimale. En Lingo, la propriété `floatPrecision` détermine le nombre de décimales utilisées pour l'affichage de ces nombres. Director utilise toujours le nombre tout entier, jusqu'à 15 chiffres utiles , dans ses calculs ; il arrondit tout nombre comportant plus de 15 chiffres utiles.

JavaScript ne fait pas de distinction entre les nombres entiers et les nombres à virgule flottante, et n'utilise que des nombres. Par exemple, les instructions suivantes montrent que le nombre 1 est un nombre entier en Lingo et un nombre dans la syntaxe JavaScript, et que le nombre décimal 1.05 est un nombre à virgule flottante en Lingo et un nombre en JavaScript :

```
--Syntaxe Lingo
put(ilk(1)) -- #integer
put(ilk(1.05)) -- #float

// Syntaxe JavaScript
trace(typeof(1)); // nombre
trace(typeof(1.05)); // nombre
```

En Lingo, vous pouvez convertir un nombre décimal en nombre entier avec la fonction `integer()`. Vous pouvez également convertir un nombre entier en nombre décimal en effectuant une opération mathématique sur le nombre entier, par exemple en multipliant le nombre entier par le nombre décimal. En JavaScript, la fonction `parseInt()` vous permet de convertir une chaîne ou un nombre décimal en nombre entier. Contrairement à la fonction `integer()` de Lingo, `parseInt()` arrondit au plus petit. Par exemple, l'instruction suivante arrondit le nombre décimal 3.9 et le convertit au nombre entier 4 (Lingo) et 3 (JavaScript).

```
--Syntaxe Lingo
leNombre = integer(3.9) - renvoie la valeur 4

// Syntaxe JavaScript
var leNombre = parseInt(3.9); // renvoie la valeur 3
```

En Lingo, la fonction `value()` peut convertir une chaîne en une valeur numérique.

Vous pouvez également utiliser une notation exponentielle avec les nombres décimaux : par exemple, `-1.1234e-100` ou `123.4e+9`.

En Lingo, vous pouvez convertir un nombre entier ou une chaîne en nombre décimal avec la fonction `float()`. En JavaScript, la fonction `parseFloat()` vous permet de convertir une chaîne en nombre décimal. Par exemple, l'instruction suivante enregistre la valeur 3.0000 (Lingo) et 3 (syntaxe JavaScript) dans la variable `leNombre`.

```
--Syntaxe Lingo
leNombre = float(3) - renvoie une valeur de 3.0000

// Syntaxe JavaScript
var leNombre = parseFloat(3) // renvoie une valeur de 3
```

Constantes

Une constante est une valeur déclarée dont le contenu ne change jamais.

En Lingo, les termes prédéfinis `TRUE`, `FALSE`, `VOID` et `EMPTY` sont des constantes car leur valeur ne change jamais. Les termes prédéfinis `BACKSPACE`, `ENTER`, `QUOTE`, `RETURN`, `SPACE` et `TAB` sont des constantes qui font référence aux touches du clavier. Par exemple, pour tester si la dernière touche enfoncée par l'utilisateur était la barre Espace, utilisez l'instruction suivante :

```
--Syntaxe Lingo
if _key.keyPressed() = SPACE then beep()
```

En JavaScript, vous pouvez accéder à des constantes prédéfinies à l'aide de types de données propres à la syntaxe JavaScript. Par exemple, objet Number contient des constantes telles que `Number.MAX_VALUE` et `Number.NaN`, l'objet Math renferme des constantes telles que `Math.PI` et `Math.E`, etc.

Remarque : Ce référentiel ne fournit pas d'informations complètes sur les constantes prédéfinies dans la syntaxe JavaScript. Pour plus d'informations sur ces constantes, consultez les nombreuses autres ressources consacrées à ce sujet.

Dans la syntaxe JavaScript, vous pouvez également définir vos propres constantes à l'aide du mot-clé `const`. Par exemple, l'instruction suivante crée une constante appelée `items` et lui attribue la valeur 20. Cette valeur ne peut pas être modifiée après sa création.

```
// Syntaxe JavaScript
const items = 20;
```

Pour plus d'informations sur les constantes, consultez le Chapitre 9, *Constantes*, page 165.

Symboles

En Lingo, un symbole est une chaîne ou toute autre valeur précédée du signe dièse (#).

Les symboles sont des constantes définies par l'utilisateur. Les comparaisons utilisant des symboles s'effectuent très rapidement, créant ainsi un code plus efficace.

Par exemple, la première instruction ci-dessous s'exécute plus rapidement que la seconde :

```
--Syntaxe Lingo
niveauUtilisateur = #novice
niveauUtilisateur = "novice"
```

Les symboles ne peuvent contenir ni espace ni ponctuation.

En Lingo et en JavaScript, convertissez une chaîne en symbole à l'aide de la méthode `symbol()`.

```
--Syntaxe Lingo
x = symbol("novice") - renvoie le résultat #novice

// Syntaxe JavaScript
var x = symbol("novice"); // renvoie le résultat #novice
```

Reconvertissez un symbole en chaîne à l'aide de la fonction `string()` (Lingo) ou la méthode `toString()` (syntaxe JavaScript).

```
--Syntaxe Lingo
x = string(#novice) -- renvoie "novice"

// Syntaxe JavaScript
var x = symbol("novice").toString(); // renvoie le résultat "novice"
```

Dans la syntaxe JavaScript, vous ne pouvez pas comparer des symboles du même nom pour déterminer s'ils font référence au même symbole. Pour comparer des symboles du même nom, vous devez d'abord les convertir en chaîne en utilisant la méthode `toString()`.

Variables

Director utilise des variables pour conserver et actualiser les valeurs. Comme son nom l'indique, une variable contient une valeur qui peut être modifiée ou mise à jour pendant la lecture de l'animation. En modifiant la valeur d'une variable pendant la lecture de l'animation, vous pouvez par exemple stocker une URL, mémoriser le nombre de fois qu'un utilisateur prend part à une session de discussion en ligne, enregistrer si une opération réseau est terminée ou non, etc.

Il est conseillé de toujours attribuer une valeur connue à une variable la première fois que vous la déclarez. Cette opération est appelée « initialisation d'une variable ». L'initialisation d'une variable facilite le suivi de cette variable et permet de comparer ses différentes valeurs au fur et à mesure de la lecture de l'animation.

Les variables peuvent être globales ou locales. Une variable locale n'existe que tant que le gestionnaire dans lequel elle a été définie est en cours d'exécution. Une variable globale peut exister et conserver sa valeur tant que l'application Director est en cours d'exécution, notamment lorsqu'une animation passe à une autre animation. Une variable peut être globale au sein d'un gestionnaire individuel, un script spécifique ou une animation entière ; la portée dépend de la manière dont la variable globale est initialisée.

Puisque les variables globales doivent en général être disponibles durant toute l'animation, il est d'usage de les déclarer dans un gestionnaire `on prepareMovie` (Lingo) ou `function prepareMovie()` (syntaxe JavaScript). Elles sont ainsi disponibles dès le début de l'animation.

Pour plus d'informations sur l'utilisation des variables globales et locales, consultez *Utilisation de variables globales*, page 24 et *Utilisation de variables locales*, page 26.

Stockage et mise à jour de valeurs dans des variables

Les variables peuvent contenir des données pour tous les types de données de Director, qu'il s'agisse de nombres entiers, de chaînes, de valeurs `TRUE` ou `FALSE`, de symboles, de listes ou du résultat d'un calcul. Pour stocker les valeurs des propriétés et des variables, utilisez l'opérateur égal à (=).

Comme indiqué dans la section Types de données de ce référentiel, les variables de Lingo et JavaScript sont dynamiques, ce qui signifie qu'elles contiennent des types de données différents à des moments différents. (La possibilité de modifier le type d'une variable distingue Lingo d'autres langages tels que Java et C++, où cette possibilité n'existe pas.)

Par exemple, l'instruction `set x = 1` crée la variable `x`, qui est une variable nombre entier, car vous lui avez attribué un entier. Si vous utilisez ensuite l'instruction `set x = "un"`, la variable `x` devient une variable à chaîne puisqu'elle contient maintenant une chaîne.

Vous pouvez convertir une chaîne en nombre à l'aide de la fonction `value()` (Lingo) ou de la méthode `parseInt()` (syntaxe JavaScript), ou un nombre en chaîne à l'aide de la fonction `string()` (Lingo) ou de la méthode `toString()` (syntaxe JavaScript).

Les valeurs de certaines propriétés peuvent être définies (la valeur est attribuée) et renvoyées (la valeur est récupérée), et les valeurs de certaines propriétés ne peuvent être que renvoyées. Les propriétés dont les valeurs peuvent être à la fois définies et renvoyées sont appelées lecture/écriture, celles qui peuvent seulement être renvoyées étant appelées lecture seule.

Il s'agit en général de propriétés décrivant une condition échappant au contrôle de Director. Par exemple, vous ne pouvez pas définir la propriété d'acteur `numChannels`, qui indique le nombre de pistes d'une animation possédant un contenu Macromedia Shockwave. Par contre, vous pouvez récupérer le nombre de pistes en faisant référence à la propriété `numChannels` d'un acteur.

Pour attribuer une valeur à une variable :

- Utilisez l'opérateur égal à (=).

Par exemple, l'instruction suivante attribue une URL à la variable `endroitsAvisiter` :

```
// Syntaxe JavaScript
var endroitsAvisiter = "http://www.macromedia.com";
```

Les variables peuvent également contenir le résultat d'opérations mathématiques. Par exemple, l'instruction suivante ajoute le résultat d'une addition à la variable `maSomme`:

```
--Syntaxe Lingo
maSomme = 5 + 5 -- ceci définit maSomme comme étant égale à 10
```

En guise d'exemple supplémentaire, l'instruction suivante renvoie l'acteur affecté à l'image-objet 2 en récupérant la valeur de la propriété `member` de l'image-objet, et la place dans la variable `ActeurTexte`.

```
--Syntaxe Lingo
ActeurTexte = sprite(2).member
```

Il est recommandé d'utiliser des noms de variables indiquant le rôle de ces variables. Vos scripts n'en seront que plus faciles à lire. Par exemple, la variable `maSomme` indique que cette variable contient le résultat d'une addition.

Pour tester la valeur de propriétés ou de variables :

- Utilisez la fonction `put()` ou `trace()` dans la fenêtre Messages ou cochez les valeurs de la fenêtre Surveillance ; `put()` et `trace()` proposent des fonctionnalités identiques et sont disponibles en Lingo et la syntaxe JavaScript.

Par exemple, l'instruction suivante affiche la valeur attribuée à la variable `monNuméro` dans la fenêtre Messages.

```
--Syntaxe Lingo
monNuméro = 20 * 7
put(monNuméro) -- affiche 140 dans la fenêtre Messages

// Syntaxe JavaScript
var monNuméro = 20 * 7;
trace(monNuméro); // affiche 140 dans la fenêtre Messages
```

Utilisation de variables globales

Les variables globales peuvent être partagées par les gestionnaires, les scripts ou les animations. Une variable globale existe et garde sa valeur tant que Director est en cours d'exécution ou jusqu'à ce que vous appeliez la méthode `clearGlobals`.

Dans Macromedia Shockwave Player, les variables globales subsistent dans les animations affichées par la méthode `goToNetMovie()`, mais pas dans celles affichées par la méthode `goToNetPage()`.

Chaque gestionnaire déclarant une variable globale peut utiliser la valeur de cette variable. S'ils modifient cette valeur, la nouvelle valeur est accessible à tous les autres gestionnaires qui considèrent cette variable comme globale.

Il est d'usage de démarrer les noms de toutes les variables globales par un *g* minuscule. Cette convention permet d'identifier plus facilement les variables globales lors de l'examen du code.

Director propose une manière d'afficher toutes les variables globales actuelles et leurs valeurs courantes, et d'effacer les valeurs de toutes les variables globales.

Pour afficher toutes les variables globales et leurs valeurs actuelles :

- Utilisez la méthode `showGlobals()` de l'objet Global dans la fenêtre Messages.

Pour plus d'informations sur la fenêtre Messages, consultez *Débogage dans la fenêtre Messages*, page 98.

Pour supprimer toutes les variables globales actuelles :

- Utilisez la méthode `clearGlobals()` de l'objet Global de la fenêtre Messages pour attribuer à toutes les variables globales la valeur `VOID` (Lingo) ou `undefined` (syntaxe JavaScript).

Le contrôle des valeurs des variables globales, lors de la lecture d'une animation, s'effectue par le biais de l'inspecteur d'objet. Pour plus d'informations sur l'inspecteur d'objet, consultez *Débogage de l'inspecteur d'objet*, page 102.

Variables globales en Lingo

En Lingo, les variables sont, par défaut, considérées comme des variables locales, et vous n'avez donc pas besoin d'insérer un mot-clé avant le nom d'une variable. Pour déclarer une variable globale, vous devez insérer le mot-clé `global` avant la variable.

Si vous déclarez une variable globale en haut de la fenêtre Script et avant un gestionnaire, cette variable sera disponible pour tous les gestionnaires de ce script. Si vous déclarez une variable globale au sein d'un gestionnaire, cette variable sera à la disposition de ce gestionnaire uniquement ; toutefois, si vous déclarez une variable globale du même nom au sein de deux gestionnaires distincts, et que vous mettez à jour la valeur de la variable dans un gestionnaire, vous mettez également à jour la valeur de la variable dans l'autre gestionnaire.

Les exemples suivants illustrent ce qui se passe avec deux variables globales : `gScript`, qui est à la disposition de tous les gestionnaires dans le script, et `gGestionnaire`, qui est disponible au sein de son propre gestionnaire et de tout autre gestionnaire qui la déclare à sa première ligne.

```
--Syntaxe Lingo
global gScript -- gScript est à la disposition de tous les gestionnaires

on mouseDown
  gGestionnaire global
  gScript = 25
  gGestionnaire = 30
end

on mouseUp
  gGestionnaire global
  trace(gGestionnaire) -- affiche 30
end
```

En Lingo, lorsque vous utilisez le terme `global` pour définir les variables globales, ces dernières reçoivent automatiquement la valeur initiale `VOID`.

Variables globales dans la syntaxe JavaScript

Dans la syntaxe JavaScript, les variables sont considérées comme des variables globales par défaut. La portée d'une variable globale peut être déterminée par la manière dont elle est déclarée et la position à laquelle elle est déclarée.

- Si vous déclarez une variable au sein d'une fonction de la syntaxe JavaScript sans la faire précéder du mot-clé `var`, cette variable sera disponible pour toutes les fonctions du script qui la contient.
- SI vous déclarez une variable en dehors d'une fonction de la syntaxe JavaScript, avec ou sans le mot-clé `var`, cette variable sera disponible pour toutes les fonctions du script qui la contient.
- Si vous déclarez une variable à l'intérieur ou à l'extérieur d'une fonction de la syntaxe JavaScript en utilisant la syntaxe `_global.`*`nomVar`*, cette variable sera disponible pour tous les scripts d'une animation.

L'exemple suivant utilise la syntaxe `_global.gFilm` dans un script pour déclarer la variable `gFilm` en tant que variable globale. Cette variable est à la disposition de tous les scripts d'une animation.

```
// Syntaxe JavaScript
_global.gFilm = 1; // déclare gFilm dans un script

// Crée une fonction dans un script séparé qui agit sur gFilm
function mouseDown() {
  _global.gFilm++;
  return(_global.gFilm);
}
```

L'exemple suivant déclare la variable globale `gScript` dans un script. Cette variable est uniquement à la disposition des fonctions de ce script.

```
// Syntaxe JavaScript
var gScript = 1; // déclare gScript dans un script

// gScript est uniquement à la disposition des fonctions du script qui le
  définit
function mouseDown() {
  gScript++;
  return(gScript);
}
```

Dans la syntaxe JavaScript, lorsque vous définissez des variables avant les gestionnaires, elles reçoivent automatiquement la valeur intiale `undefined`.

Utilisation de variables locales

Une variable locale n'existe que tant que le gestionnaire dans lequel elle a été définie est en cours d'exécution. Toutefois, après avoir créé un variable locale, vous pouvez l'utiliser dans d'autres expressions ou modifier sa valeur tant qu'un script se trouve dans le gestionnaire qui a défini la variable.

Il est préférable de définir une variable comme locale lorsque vous ne souhaitez l'utiliser que provisoirement dans un gestionnaire. Vous limitez ainsi les risques de modification accidentelle de sa valeur dans d'autres gestionnaires utilisant le même nom de variable.

Pour créer une variable locale :

- En Lingo, attribuez une valeur à la variable avec l'opérateur égal à (=).
- Dans la syntaxe JavaScript, et à l'intérieur d'une fonction, insérez le mot-clé `var` avant le nom de la variable puis attribuez-lui une valeur à l'aide de l'opérateur égal à.

  Remarque : Vu que les variables de la syntaxe JavaScript sont, par défaut, globales, si vous essayez de déclarer une variable locale à l'intérieur d'une fonction sans utiliser le mot-clé `var`, votre script risque de créer un comportement inattendu. Ainsi, et bien que l'utilisation de `var` soit facultative, il est fortement recommandé de déclarer toutes les variables JavaScript locales à l'aide de `var` pour éviter tout comportement inattendu.

Pour afficher toutes les variables locales d'un gestionnaire :

- En Lingo seulement, utilisez la fonction `showLocals()`.

En Lingo, vous pouvez utiliser cette méthode dans la fenêtre Messages ou dans des gestionnaire pour faciliter le débogage des scripts. Le résultat apparaît dans la fenêtre Messages. La méthode `showLocals()` ne s'applique pas à la syntaxe JavaScript.

Pour contrôler les valeurs des variables locales lors de la lecture d'une animation, utilisez l'inspecteur d'objet. Pour plus d'informations sur l'inspecteur d'objet, consultez *Débogage de l'inspecteur d'objet*, page 102.

Opérateurs

Les opérateurs sont des éléments indiquant à Lingo et JavaScript comment combiner, comparer ou modifier les valeurs d'une expression. Plusieurs des opérateurs de Director sont partagés par la syntaxe Lingo et la syntaxe JavaScript, et certains sont propres à un langage.

Certains types d'opérateurs comprennent :

- Des opérateurs arithmétiques (tels que +, -, / et *)
- Des opérateurs de comparaison (tels que <, > et >=), qui comparent deux arguments
- Des opérateurs logiques (`not`, `and`, `or`) qui combinent des conditions simples en conditions composées
- Des opérateurs de chaînes (tels que `&`, `&&`, et +) qui relient ou concatènent des chaînes de caractères.

Remarque : Il existe beaucoup plus de types d'opérateurs dans la syntaxe JavaScript qu'en Lingo, et ils ne sont pas tous décrits dans ce référentiel. Pour plus d'informations sur les autres opérateurs en JavaScript 1.5, consultez les nombreuses autres ressources consacrées à ce sujet.

Les éléments sur lesquels les opérateurs agissent sont appelés des opérandes. En Lingo, il n'existe que des opérateurs binaires. Dans la syntaxe JavaScript, il existe des opérateurs binaires et unaires. Un opérateur binaire nécessite deux opérandes, l'un placé avant l'opérateur et l'autre après. Un opérateur unaire nécessite un seul opérande, placé soit avant soit après l'opérateur.

Dans l'exemple suivant, la première instruction illustre un opérateur binaire où les variables `x` et `y` sont des opérandes et le signe plus (+) l'opérateur. La seconde instruction illustre un opérateur unaire où la variable `i` est l'opérande et ++ l'opérateur.

```
// Syntaxe JavaScript
x + y; // opérateur binaire
i++; // opérateur unaire
```

Pour plus d'informations sur les opérateurs, consultez le Chapitre 13, *Opérateurs*, page 635.

Ordre de priorité des opérateurs

Lorsqu'un ou plusieurs opérateurs sont utilisés dans la même instruction, certains opérateurs sont prioritaires par rapport à d'autres selon une hiérarchie précise qui détermine les opérateurs à exécuter en premier. Cette hiérarchie est appelée « ordre de priorité des opérateurs ». Par exemple, une multiplication est toujours effectuée avant une addition. Cependant, les éléments entre parenthèses sont prioritaires par rapport à la multiplication. Dans l'exemple suivant, en l'absence de parenthèses, la multiplication de cette instruction se produit en premier :

```
--Syntaxe Lingo
total = 2 + 4 * 3 - renvoie une valeur de 14
```

Lorsque l'addition apparaît entre parenthèses, l'addition s'effectue en premier :

```
--Syntaxe Lingo
total = (2 + 4) * 3 - renvoie une valeur de 18
```

Vous trouverez ci-dessous une description des types d'opérateurs et leur ordre de priorité. Les opérateurs possédant une priorité élevée sont exécutés en premier. Par exemple, un opérateur dont l'ordre de priorité est 5 est exécuté avant un opérateur dont l'ordre de priorité est 4. Les opérations qui ont le même ordre de priorité sont exécutées de gauche à droite.

Opérateurs arithmétiques

Les opérateurs arithmétiques additionnent, soustraient, multiplient, divisent et effectuent d'autres opérations arithmétiques. Les parenthèses et le signe moins sont aussi des opérateurs arithmétiques.

Opérateur	Effet	Ordre de priorité
`( )`	Opérations permettant de contrôler la priorité.	5
`-`	Lorsque placé devant un chiffre, en inverse la valeur.	5
`*`	Effectue une multiplication.	4
`mod`	(Lingo seulement) Effectue des opérations de modulos.	4
`/`	Effectue une division.	4
`%`	(Syntaxe JavaScript seulement) Renvoie le reste du nombre entier résultant d'une division de deux opérandes.	4
`++`	(Syntaxe JavaScript seulement) Ajoute un à son opérande. Lorsque utilisé comme opérateur préfixe (`++x`), renvoie la valeur de son opérande après avoir ajouté un. Lorsque utilisé comme opérateur postfixe (`x++`), renvoie la valeur de son opérande avant d'ajouter un.	4
`--`	(Syntaxe JavaScript seulement) Soustrait un de son opérande. La valeur renvoyée est analogue à celle de l'opérateur d'incréments.	4
`+`	Lorsque placé entre deux chiffres, effectue une addition.	3
`-`	Lorsque placé entre deux chiffres, effectue une soustraction.	3

Remarque : En Lingo, lorsque seuls des nombres entiers sont utilisés dans une opération, le résultat est un nombre entier. Si vous utilisez des entiers et des nombres à virgule flottante dans la même opération, le résultat est toujours un nombre à virgule flottante. Dans la syntaxe JavaScript, tous les calculs renvoient comme résultat des nombres à virgule flottante.

Si le résultat de la division d'un entier par un autre entier n'est pas un nombre entier, Lingo arrondit le résultat au nombre entier inférieur le plus proche. Par exemple, le résultat de `4/3` est `1`. Dans la syntaxe JavaScript, la valeur à virgule flottante, 1.333, est renvoyée.

Pour forcer Lingo à calculer une valeur sans arrondir le résultat, utilisez `float()` avec une ou plusieurs des valeurs dans l'expression. Par exemple, le résultat de `4/float(3)` est 1.333.

Opérateurs de comparaison

Les opérateurs de comparaison comparent deux valeurs et déterminent si la comparaison est vraie (true) ou fausse (false).

Opérateur	Signification	Ordre de priorité
==	(Syntaxe JavaScript seulement) Deux opérandes sont égaux. Si les opérandes ne sont pas du même type de données, la syntaxe JavaScript essaie de les convertir en type de données approprié afin d'effectuer une comparaison.	1
===	(Syntaxe JavaScript seulement) Deux opérandes sont égaux et du même type.	1
!=	(Syntaxe JavaScript seulement) Deux opérandes ne sont pas égaux. Si les opérandes ne sont pas du même type de données, la syntaxe JavaScript essaie de les convertir en type de données approprié afin d'effectuer une comparaison.	1
!==	(Syntaxe JavaScript seulement) Deux opérandes ne sont pas égaux et/ou du même type.	1
<>	(Syntaxe Lingo seulement) Deux opérandes ne sont pas égaux.	1
<	L'opérande de gauche est inférieur à l'opérande de droite.	1
<=	L'opérande de gauche est inférieur ou égal à l'opérande de droite.	1
>	L'opérande de gauche est supérieur à l'opérande de droite.	1
>=	L'opérande de gauche est supérieur ou égal à l'opérande de droite.	1
=	(Lingo seulement) Deux opérandes sont égaux	1

Opérateurs d'affectation

Un opérateur d'affectation attribue une valeur à son opérande de gauche en fonction de la valeur de son opérande de droite. A l'exception de l'opérateur d'affectation de base égal à (=), tous les opérateurs d'affectation de raccourcis suivants ne s'appliquent qu'à la syntaxe JavaScript.

Opérateur	Signification	Ordre de priorité
=	Egal à	1
x += y	(Syntaxe JavaScript seulement) x = x + y	1
x -= y	(Syntaxe JavaScript seulement) x = x - y	1
x *= y	(Syntaxe JavaScript seulement) x = x * y	1
x /= y	(Syntaxe JavaScript seulement) x = x / y	1
x %= y	(Syntaxe JavaScript seulement) x = x % y	1

Opérateurs logiques

Les opérateurs logiques testent si deux expressions logiques sont vraies (true) ou fausses (false).

Faites attention lorsque vous utilisez les opérateurs logiques et les opérateurs de chaînes en Lingo et en JavaScript. Par exemple, dans la syntaxe JavaScript, `&&` est un opérateur logique qui détermine si deux expressions sont vraies, mais en Lingo, `&&` est un opérateur de chaînes qui concatène deux chaînes et insère un espace entre deux expressions.

Opérateur	Effet	Ordre de priorité
`and`	(Lingo seulement) Détermine si les deux expressions sont vraies.	4
`&&`	(Syntaxe JavaScript seulement) Détermine si les deux expressions sont vraies.	4
`ou`	(Lingo seulement) Détermine si une ou les deux expressions sont vraies.	4
`||`	(Syntaxe JavaScript seulement) Détermine si une ou les deux expressions sont vraies.	4
`not`	(Lingo seulement) Inverse une expression	5
`!`	(JavaScript seulement) Inverse une expression	5

L'opérateur `not` (Lingo) ou ! (Syntaxe JavaScript) est utile lorsqu'il s'agit de faire passer la valeur `TRUE` ou `FALSE` à la valeur opposée. Par exemple, l'instruction suivante active le son s'il est désactivé, ou le désactive s'il est activé :

```
--Syntaxe Lingo
_sound.soundEnabled = not (_sound.soundEnabled)

// Syntaxe JavaScript
_sound.soundEnabled = !(_sound.soundEnabled);
```

Opérateurs de chaînes

Les opérateurs de chaînes combinent et définissent des chaînes.

Faites attention lorsque vous utilisez les opérateurs logiques et les opérateurs de chaînes en Lingo et en JavaScript. Par exemple, dans la syntaxe JavaScript, `&&` est un opérateur logique qui détermine si deux expressions sont vraies, mais en Lingo, `&&` est un opérateur de chaînes qui concatène deux chaînes et insère un espace entre deux expressions.

Opérateur	Effet	Ordre de priorité
&	(Lingo seulement) Concatène deux chaînes	2
+	(Syntaxe JavaScript seulement) Concatène deux valeurs de chaîne et renvoie une troisième chaîne qui unit les deux opérandes.	2
+=	(Syntaxe JavaScript seulement) Concatène une variable de chaîne et une valeur de chaîne et attribue la valeur renvoyée à la variable de chaîne.	2
&&	(Lingo seulement) Concatène deux chaînes et insère un espace entre elles.	2
"	Indique le début ou la fin d'une chaîne.	1

Constructions conditionnnelles

Par défaut, Director exécute toujours les instructions d'un script en commençant par la première instruction et en continuant dans l'ordre dans lequel elles apparaissent, jusqu'à ce qu'il rencontre la dernière instruction, ou une instruction l'envoyant à un autre endroit.

L'ordre d'exécution des instructions est tributaire de l'ordre dans lequel vous les placez. Par exemple, si vous rédigez une instruction nécessitant une valeur calculée, vous devez d'abord placer une instruction calculant cette valeur.

Dans l'exemple suivant, la première instruction additionne deux nombres et la seconde affecte une représentation de la somme sous forme de chaîne à un acteur champ appelé `Réponse`, affiché sur la scène. La seconde instruction n'a pu être placée avant la première parce que la variable `x` n'a pas encore été définie.

```
--Syntaxe Lingo
x = 2 + 2
member("Réponse").text = string(x)

// Syntaxe JavaScript
var x = 2 + 2;
member("Réponse").text = x.toString();
```

Lingo et JavaScript proposent tous deux des conventions servant à modifier l'ordre d'exécution ou les instructions de script par défaut et aussi à effectuer des actions en fonction de conditions spécifiques. Par exemple, vous pouvez choisir d'effectuer les actions suivantes dans vos scripts :

- Exécuter une série d'instructions si une condition logique est vraie ou exécuter d'autres instructions si la condition logique est fausse.
- Evaluer une expression et essayer d'adapter sa valeur à une condition spécifique.
- Exécuter une série d'instructions plusieurs fois de suite jusqu'à ce qu'une condition spécifique soit remplie.

Test de conditions logiques

Pour exécuter une instruction ou une série d'instructions lorsqu'une condition donnée est vraie ou fausse, vous utilisez les structures `if...then...else` (Lingo) ou `if...else` (syntaxe JavaScript). Par exemple, vous pouvez créer une structure `if...then...else` ou `if...then` qui vérifie que le téléchargement du texte à partir d'Internet est terminé et qui, le cas échéant, le formate. Ces structures utilisent le modèle suivant pour tester les conditions logiques :

- En Lingo et JavaScript, les instructions vérifiant si une condition est vraie ou fausse commencent par l'élément `if`.
- En Lingo, si la condition existe, les instructions suivant l'élément `then` sont exécutées. Dans la syntaxe JavaScript, les accolades (`{ }`) remplacent l'élément Lingo `then` et doivent encadrer chaque instruction `if`, `else` ou `else if`.
- En Lingo et la syntaxe JavaScript, si la condition n'existe pas, les scripts passent à l'instruction suivante du gestionnaire en utilisant l'élément `else` ou `else if`.
- En Lingo, l'élément `end if` indique la fin du test `if`. Dans la syntaxe JavaScript, le test `if` se termine automatiquement.

Pour optimiser les performances de vos scripts, commencez par tester les conditions les plus vraisemblables.

Les instructions suivantes testent plusieurs conditions. Le terme `else if` spécifie l'exécution d'autres tests si les conditions précédentes se sont avérées fausses :

```
--Syntaxe Lingo
if _mouse.mouseMember = member(1) then
  _movie.go("Caire")
else if _mouse.mouseMember = member(2) then
  _movie.go("Nairobi")
else
  _player.alert("Vous vous êtes perdu.")
end if

// Syntaxe JavaScript
if (_mouse.mouseMember = member(1)) {
  _movie.go("Caire");
}
else if (_mouse.mouseMember = member(2)) {
  _movie.go("Nairobi");
}
else {
  _player.alert("Vous vous êtes perdu.")
}
```

Lorsque vous rédigez des structures `if...then` en Lingo, vous pouvez placer l'instruction ou les instructions après `then` sur la même ligne que celle de `then`, ou les placer sur leur propre ligne en insérant un retour de chariot après `then`. Si vous insérez un retour de chariot, vous devez également placer une instruction `end if` à la fin de la structure `if…then`.

Lorsque vous rédigez des structures `if` dans la syntaxe JavaScript, vous pouvez placer l'instruction ou les instructions après `if` sur la même ligne que celle de `if`, ou les placer sur leur propre ligne en insérant un retour de chariot après `if`.

Par exemple, les instructions suivantes sont équivalentes :

```
--Syntaxe Lingo
if _mouse.mouseMember = member(1) then _movie.go("Caire")

if _mouse.mouseMember = member(1) then
  _movie.go("Caire")
end if

// Syntaxe JavaScript
if (_mouse.mouseMember = member(1)) { _movie.go("Caire"); }

if (_mouse.mouseMember = member(1)) {
  _movie.go("Caire");
}
```

Pour obtenir des informations concernant l'utilisation des structures `if...then...else` et `if...else`, consultez *if*, page 227.

Evaluation et concordance d'expressions

Les structures `case` (Lingo) et `switch...case` (syntaxe JavaScript) sont des raccourcis permettant d'éviter l'utilisation des structures `if...then...else` ou `if...then` dans les structures à branchements multiples. Les structures `case` et `switch...case` sont souvent plus efficaces et plus lisibles que beaucoup de structures `if...then...else` ou `if...then`.

En Lingo, la condition devant être testée suit le terme `case` dans la première ligne de la structure `case`. La comparaison commence par la première ligne, puis passe à la suivante, etc. jusqu'à ce que Lingo rencontre une expression correspondant à la condition testée. Lorsqu'une correspondance est trouvée, Director exécute les instructions Lingo qui suivent l'expression.

Dans la syntaxe JavaScript, la condition devant être testée suit le terme `switch` dans la première ligne de la structure. Chaque comparaison du test suit le terme `case` dans chaque ligne contenant un test. Vous pouvez mettre fin à une comparaison `case` en utilisant le terme facultatif `break`. Lorsque vous insérez le terme `break`, vous excluez le programme de la structure `switch` et exécutez toute instruction qui suit la structure. Si vous n'insérez pas `break`, la comparaison `case` suivante est exécutée.

Une structure `case` ou `switch...case` peut utiliser des comparaisons comme conditions de test.

Par exemple, les structures `case` et `switch...case` suivantes testent la dernière touche sur laquelle l'utilisateur a appuyé, et répond en conséquence.

- Si l'utilisateur a appuyé sur A, l'animation passe à l'image Pomme.
- Si l'utilisateur a appuyé sur B ou C, l'animation exécute la transition demandée et passe à l'image Oranges.

- Si l'utilisateur a appuyé sur n'importe quelle autre touche, l'ordinateur émet un bip sonore.

```
--Syntaxe Lingo
case (_key.key) of
  "a" : _movie.go("Pomme")
  "b", "c":
     _movie.puppetTransition(99)
     _movie.go("Oranges")
  otherwise: _sound.beep()
end case

// Syntaxe JavaScript
switch (_key.key) {
  case "a" :
    _movie.go("Pomme");
    break;
  case "b":
    _movie.puppetTransition(99);
    _movie.go("Oranges");
    break;
  case "c":
    _movie.puppetTransition(99);
    _movie.go("Oranges");
    break;
  default: _sound.beep()
}
```

Remarque : Dans la syntaxe JavaScript, vous ne pouvez effectuer qu'une seule comparaison par instruction `case`.

Pour obtenir des informations concernant l'utilisation des structures `case`, consultez *case*, page 222.

Répétition d'actions

En Lingo et dans la syntaxe JavaScript, vous pouvez répéter une action un certain nombre de fois, ou tant qu'une condition spécifique existe.

Pour répéter une action un certain nombre de fois en Lingo, vous utilisez une structure `repeat with`. Spécifiez le nombre de répétitions sous forme de plage après l'instruction `repeat with`.

Pour répéter une action un certain nombre de fois dans la syntaxe JavaScript, vous utilisez la structure `for`. La structure `for` nécessite trois paramètres : le premier paramètre initialise généralement une variable de compteur, le second précise une condition à évaluer à chaque fois dans la boucle et le troisième est généralement utilisé pour mettre à jour ou incrémenter la variable de compteur.

Les structures `repeat with` et `for` servent à effectuer la même opération sur une série d'objets. Par exemple, la boucle suivante applique l'encre Fond transparent aux images-objets 2 à 10 :

```
--Syntaxe Lingo
repeat with n = 2 to 10
  sprite(n).ink = 36
end repeat

// Syntaxe JavaScript
for (var n=2; n<=10; n++) {
  sprite(n).ink = 36;
}
```

Cet exemple exécute une action similaire, mais avec des nombres décroissants :

```
--Syntaxe Lingo
repeat with n = 10 down to 2
  sprite(n).ink = 36
end repeat

// Syntaxe JavaScript
for (var n=10; n>=2; n--) {
  sprite(n).ink = 36;
}
```

En Lingo, pour répéter une série d'instructions tant qu'une condition spécifique existe, utilisez la structure `repeat while`.

Dans la syntaxe JavaScript, pour répéter une série d'instructions tant qu'une condition spécifique existe, utilisez la structure `while`.

Par exemple, les instructions suivantes font émettre un bip sonore continu à l'animation à chaque fois que l'utilisateur appuie sur le bouton de la souris :

```
--Syntaxe Lingo
repeat while _mouse.mouseDown
  _sound.beep()
end repeat

// Syntaxe JavaScript
while (_mouse.mouseDown) {
  _sound.beep();
}
```

Les scripts Lingo et JavaScript continuent à effectuer une boucle sur les instructions de la boucle jusqu'à ce que la condition ne soit plus vraie, ou jusqu'à ce que l'une des instructions envoie le script à l'extérieur de la boucle. Dans l'exemple précédent, le script quitte la boucle lorsque l'utilisateur relâche le bouton de la souris puisque la condition `mouseDown` cesse d'exister.

Pour quitter une boucle en Lingo, utilisez l'instruction `exit repeat`.

Pour quitter une boucle dans la syntaxe JavaScript, vous pouvez utiliser le terme `break`. Le script quitte automatiquement la boucle lorsqu'une condition n'est plus vraie.

Par exemple, les instructions suivantes font émettre un bip sonore à l'animation lorsque l'utilisateur maintient le bouton de la souris enfoncé, sauf si le pointeur de la souris se trouve au dessus de l'image-objet 1. Dans ce cas, le script quitte la boucle et le bip sonore s'arrête. La méthode `rollover()` indique si le pointeur se trouve sur l'image-objet spécifiée.

```
--Syntaxe Lingo
repeat while _mouse.stillDown
  _sound.beep()
  if _movie.rollOver(1) then exit repeat
end repeat

// Syntaxe JavaScript
while (_mouse.stillDown) {
  _sound.beep();
  if (_movie.rollOver(1)) {
    break;
  }
}
```

Pour obtenir des informations sur les structures `repeat while` et `while`, consultez *repeat while*, page 238.

Evénements, messages et gestionnaires

Pour créer des scripts solides et utiles, il est essentiel de maîtriser les concepts et les fonctionnalités des événements, messages et gestionnaires. Si vous comprenez l'ordre dans lequel les événements et les messages sont envoyés et reçus, vous pouvez déterminer exactement le moment auquel des scripts donnés ou des parties de scripts devraient être exécutés. Ceci vous aidera à déboguer vos scripts si certaines actions ne se produisent pas au moment prévu.

Les événements suivants se produisent durant la lecture d'une animation :

1 Des événements se produisent en réaction à une action du système ou à une action définie par l'utilisateur

2 Des messages correspondant à ces événements sont envoyés aux scripts d'une animation

3 Les gestionnaires compris dans les scripts contiennent les instructions qui sont exécutées lorsqu'un message est reçu

Le nom d'un événement correspond au nom du message qu'il produit, et le gestionnaire de cet événement correspond à la fois à l'événement et au message. Par exemple, lorsque l'événement `mouseDown` se produit, Director crée et envoie aux scripts un message appelé `mouseDown` qui est ensuite pris en charge par un gestionnaire appelé `mouseDown`.

Evénements

Durant la lecture d'une animation, deux catégories d'événements se produisent :

- **Les événements système** se produisent sans qu'il n'y ait d'interaction entre un utilisateur et l'animation, et sont prédéfinis et nommés dans Director. Lorsque la tête de lecture entre dans une image, par exemple, lorsque vous cliquez sur une image-objet, etc.
- **Les événements définis par l'utilisateur** se produisent en réponse aux actions que vous définissez. Par exemple, vous pouvez créer un événement qui se produit lorsque la couleur de fond d'une image-objet passe du rouge au bleu, après qu'un son ait été lu cinq fois, etc.

Plusieurs événements système tels que `prepareFrame`, `beginSprite`, etc., se produisent automatiquement et dans un ordre prédéfini durant la lecture d'une animation. D'autres événements système, en particulier des événements liés à la souris tels que `mouseDown`, `mouseUp` et ainsi de suite, ne se produisent pas nécessairement de manière automatique durant la lecture d'une animation, mais plutôt après qu'ils aient été déclenchés par un utilisateur.

Par exemple, au début d'une animation, l'événement `prepareMovie` est toujours le premier à se produire, l'événement `prepareFrame` toujours le second, etc. Toutefois, les événements `mouseDown` et `mouseUp` peuvent ne jamais se produire dans une animation, à moins qu'un utilisateur ne les déclenche en cliquant sur l'animation.

La liste suivante répertorie les événements système qui se produisent toujours durant une animation, et l'ordre dans lequel ils se produisent.

Les événements se produisent dans l'ordre suivant au démarrage de l'animation :

1 `prepareMovie`

2 `prepareFrame` Immédiatement après l'événement `prepareFrame`, Director lit les sons, dessine les images-objets et effectue les transitions ou les effets de palette. Cet événement se produit avant l'événement `enterFrame`. L'utilisation d'un gestionnaire `on prepareFrame` est conseillée pour exécuter un script avant que l'image ne soit dessinée.

3 `beginSprite` Cet événement se produit lorsque la tête de lecture pénètre dans la zone d'une image-objet.

4 `startMovie` Cet événement se produit dans la première image qui est lue.

Lorsqu'une animation rencontre une image, les événements se produisent dans l'ordre suivant :

1 `beginSprite` Cet événement ne se produit que si de nouvelles images-objets commencent dans l'image.

2 `stepFrame`

3 `prepareFrame`

4 `enterFrame` Après `enterFrame` mais avant `exitFrame`, Director gère les délais exigés par les réglages de cadence, les événements d'inactivité et les événements clavier et souris.

5 `exitFrame`

6 `endSprite` Cet événement ne se produit que lorsque la tête de lecture sort d'une image-objet de l'image.

Les événements se produisent dans l'ordre suivant lorsque la lecture de l'animation s'arrête :

1 `endSprite` Cet événement ne se produit que si les images-objets existent actuellement dans l'animation.

2 `stopMovie`

Pour plus d'informations sur les événements système prédéfinis de Director, consultez le Chapitre 10, *Evénements et messages*, page 173.

Messages

Pour exécuter au bon moment le groupe d'instructions approprié, Director doit déterminer ce qui se passe dans l'animation et les instructions à exécuter pour répondre à certains événements.

Director utilise des messages pour indiquer que des événements spécifiques se produisent dans l'animation, tels qu'un clic sur une image-objet, l'enfoncement des touches du clavier, le démarrage de l'animation, l'entrée ou la sortie de la tête de lecture dans une image, ou encore le renvoi d'une valeur spécifique par un script.

L'ordre dans lequel les messages sont envoyés aux objets de l'animation est le suivant :

1 Les messages sont d'abord envoyés aux comportements associés aux images-objets affectées par l'événement. Si plusieurs comportements sont associés à une image-objet, ceux-ci répondent au message dans l'ordre dans lequel ils ont été associés à cette image-objet.

2 Les messages sont ensuite envoyés au script de l'acteur affecté à l'image-objet.

3 Les messages sont ensuite envoyés aux comportements associés à l'image courante.

4 Enfin, les messages sont envoyés aux scripts de l'animation.

Bien qu'il vous soit possible de définir le nom de vos messages, la plupart des événements communs survenant dans une animation possèdent des noms de message prédéfinis.

Pour plus d'informations sur les messages intégrés à Director, consultez le Chapitre 10, *Evénements et messages*, page 173.

Définition de messages personnalisés

En plus d'utiliser les noms de messages prédéfinis, vous pouvez définir vos propres messages et les noms des gestionnaires correspondants. Un message personnalisé peut appeler un autre script, un autre gestionnaire ou le gestionnaire de l'instruction même. Lorsque le gestionnaire appelé a terminé son exécution, l'exécution du gestionnaire qui l'a appelé reprend.

Un nom de message et gestionnaire personnalisé doit répondre aux critères suivants :

- Il doit débuter par une lettre.
- Il doit inclure uniquement des caractères alphanumériques (pas de caractères spéciaux ou de ponctuation).
- Il doit être composé d'un ou de plusieurs mots reliés par un trait de soulignement (_). Les espaces ne sont pas autorisés.
- Il ne peut pas être identique au nom d'un élément Lingo ou JavaScript prédéfini.

L'utilisation de mots-clés Lingo ou JavaScript prédéfinis pour les noms de messages et gestionnaires peut prêter à confusion. Bien qu'il soit possible de remplacer ou d'augmenter explicitement la fonctionnalité d'un élément Lingo ou JavaScript en l'utilisant comme nom de gestionnaire ou de message, cette opération ne devrait être effectuée que dans certaines situations avancées.

Lorsque vous utilisez plusieurs gestionnaires remplissant des fonctions similaires, donnez-leur des noms commençant de la même manière afin de les regrouper dans les listes alphabétiques, comme par exemple la liste qui apparaît lorsque vous sélectionnez l'option Edition > Rechercher > Gestionnaire.

Gestionnaires

Un gestionnaire est un ensemble d'instructions placées dans un script et exécutées en réponse à un événement déterminé et à un message subséquent. Bien que Director contienne des événements et des messages prédéfinis, vous devez créer vos propres gestionnaires pour chaque paire d'événements/images que vous voulez gérer.

Stratégie de positionnement des gestionnaires

Vous pouvez placer des gestionnaires dans n'importe quel type de script, un script pouvant contenir plusieurs gestionnaires. Toutefois, il est recommandé de regrouper les gestionnaires apparentés au même endroit afin d'en simplifier la gestion.

Les recommandations suivantes s'appliquent aux situations les plus courantes :

- Pour associer un gestionnaire à une image-objet déterminée ou pour exécuter un gestionnaire en réponse à une action dans une image-objet précise, placez le gestionnaire dans un comportement affecté à l'image-objet.

- Pour définir un gestionnaire disponible à tout moment lorsque l'animation se trouve dans une image déterminée, placez-le dans un comportement affecté à l'image.

  Par exemple, pour qu'un gestionnaire réponde à un clic de la souris lorsque la tête de lecture est dans une image, quel que soit l'endroit où se produit le clic, placez un gestionnaire `mouseDown` ou `mouseUp` dans le comportement de l'image plutôt que dans un comportement d'image-objet.
- Pour définir un gestionnaire exécuté en réponse à des messages d'événements se produisant n'importe où dans l'animation, placez-le dans un script d'animation.
- Pour définir un gestionnaire exécuté en réponse à un événement affectant un acteur, quelles que soient les images-objets utilisant cet acteur, placez-le dans un script d'acteur.

Identification du moment auquel les gestionnaires reçoivent un message

Après avoir envoyé un message aux scripts, Director vérifie la présence de gestionnaires dans un ordre défini.

1. Director vérifie d'abord l'existence de gestionnaires dans l'objet à partir duquel le message a été envoyé. Si un gestionnaire est trouvé, le message est intercepté et le script du gestionnaire est exécuté.
2. S'il ne trouve pas de gestionnaire, Director vérifie, dans un ordre croissant, les scripts d'animation associés à un acteur et pouvant contenir un gestionnaire lié au message. Si un gestionnaire est trouvé, le message est intercepté et le script du gestionnaire est exécuté.
3. Si aucun gestionnaire n'a été trouvé, Director vérifie si un script d'image contient un gestionnaire pour le message. Si un gestionnaire est trouvé, le message est intercepté et le script du gestionnaire est exécuté.
4. S'il ne trouve pas de gestionnaire, Director vérifie, dans un ordre croissant, les scripts associés aux images-objets et pouvant contenir un gestionnaire lié au message. Si un gestionnaire est trouvé, le message est intercepté et le script du gestionnaire est exécuté.

Le message n'est pas automatiquement transmis aux emplacements restants après son interception par le gestionnaire. Toutefois, en Lingo, vous pouvez utiliser la méthode `pass()` pour ignorer cette règle par défaut et passer le message à d'autres objets.

Si le gestionnaire recherché n'est pas trouvé après l'envoi du message à tous les emplacements possibles, Director ignore le message.

L'ordre exact dans lequel Director envoie un message aux objets dépend du message même. Pour plus d'informations sur l'ordre des objets auxquels Director envoie des messages spécifiques, consultez l'entrée correspondant à chaque message dans Chapitre 10, *Evénements et messages*, page 173.

Utilisation de paramètres pour passer des valeurs à un gestionnaire

L'utilisation de paramètres comme valeurs vous permet de transmettre à un gestionnaire les valeurs exactes nécessaires au moment voulu, sans tenir compte de l'endroit ni du moment auxquels vous appelez le gestionnaire dans l'animation. Les paramètres peuvent être facultatifs ou obligatoires, selon le cas.

Pour créer des paramètres pour un gestionnaire :

- En Lingo, placez les paramètres après le nom du gestionnaire. Dans la syntaxe JavaScript, mettez les paramètres entre parenthèses puis placez-les après le nom du gestionnaire. Utilisez des virgules pour séparer les arguments multiples.

Lorsque vous appelez un gestionnaire, vous devez fournir des valeurs spécifiques pour les paramètres qu'il utilise. Vous pouvez utiliser n'importe quel type de valeur, tel qu'un nombre, une variable à laquelle une valeur est affectée ou une chaîne. Les valeurs de l'instruction appelante doivent suivre le même ordre que dans les paramètres du gestionnaire et être encadrées de parenthèses.

Dans l'exemple suivant, la variable `maSomme` appelle la méthode `additionParam` qui reçoit les deux valeurs 2 et 4. Le gestionnaire `additionParam` remplace les repères d'emplacement de paramètres `a` et `b` par les deux valeurs qu'il a reçues, stocke le résultat dans la variable locale `c` puis utilise le mot-clé `return` pour renvoyer le résultat à la méthode originale qui est ensuite attribuée à `maSomme`.

Étant donné que 2 figure le premier dans la liste des paramètres, il remplace `a` dans le gestionnaire. De même, étant donné que 4 est le second dans la liste des paramètres, il remplace `b` dans le gestionnaire.

```
--Syntaxe Lingo
maSomme = additionParam(2, 4) -- instructions d'appel

on additionParam a, b -- gestionnaire
  c = a + b
  return c - renvoie le résultat à l'instruction d'appel
end

// Syntaxe JavaScript
var maSomme = additionParam(2, 4); // instructions d'appel

function additionParam(a, b) { // gestionnaire
  c = a + b;
  return c; // renvoie le résultat à l'instruction d'appel
}
```

En Lingo, lorsque vous appelez une méthode personnalisée à partir d'un objet, une référence à l'objet script de la mémoire est transmise en tant que premier paramètre implicite au gestionnaire de la méthode. Cela signifie que vous devez prendre en charge l'objet script dans votre gestionnaire.

Par exemple, imaginez que vous ayez rédigé une méthode d'image-objet personnalisée appelée `jump()` dont le paramètre est un nombre entier simple et que vous l'ayez placée dans un comportement. Lorsque vous appelez `jump()` d'une référence d'un objet image-objet, le gestionnaire doit également inclure un paramètre représentant la référence de l'objet image-objet, et non pas uniquement le nombre entier. Dans ce cas, le paramètre désigné est représenté par le mot-clé `me`, mais tout autre terme serait également acceptable.

```
--Syntaxe Lingo
maHauteur = sprite(2).jump(5)

on jump(me,a)
  return a + 15 - ce gestionnaire fonctionne correctement et renvoie 20
end

on jump(a)
  return a + 15 -- ce gestionnaire ne fonctionne pas correctement et renvoie 0
end
```

Vous pouvez également utiliser des expressions comme valeurs. Par exemple, l'instruction suivante utilise `3+6` pour remplacer `a` et `8>2` (ou `1`, représentant `TRUE`) pour remplacer `b` et renvoie `10` :

```
--Syntaxe Lingo
maSomme = additionParam(3+6, 8>2)
```

En Lingo, tous les gestionnaires commencent par le mot `on`, suivi du message auquel ils doivent répondre. La dernière ligne du gestionnaire est le mot `end`. Vous pouvez répéter le nom du gestionnaire après `end`, ce qui n'est pas obligatoire.

Dans la syntaxe JavaScript, tous les gestionnaires commencent par le mot `function`, suivi du message auquel ils doivent répondre. Les instructions comprenant le gestionnaire sont mises entre crochets et sont toutes des fonctions de la syntaxe JavaScript.

Renvoi de résultats des gestionnaires

Il est souvent utile qu'un gestionnaire vous indique la présence d'une condition ou le résultat d'une action.

Pour renvoyer des résultats avec un gestionnaire :

- Utilisez le mot-clé `return` pour qu'un gestionnaire retourne la présence d'une condition ou le résultat d'une action. Par exemple, le gestionnaire `findColor` suivant retourne la couleur actuelle de l'image-objet 1 :

```
--Syntaxe Lingo
on findColor
  return sprite(1).foreColor
end

// Syntaxe JavaScript
function findColor() {
  return(sprite(1).foreColor);
}
```

Vous pouvez également utiliser le mot-clé `return` tout seul pour quitter le gestionnaire actuel et ne renvoyer aucune valeur. Par exemple, le gestionnaire `jump` suivant ne renvoie rien si le paramètre `aVal` est égal à 5 ; autrement, il renvoie une valeur.

```
--Syntaxe Lingo
on jump(aVal)
  if aVal = 5 then return

  aVal = aVal + 10
  return aVal
end

// Syntaxe JavaScript
function jump(aVal) {
  if(aVal == 5) {
    return;
  }
  else {
    aVal = aVal + 10;
    return(aVal);
  }
}
```

Lorsque vous définissez un gestionnaire renvoyant un résultat, vous devez le faire suivre de parenthèses quand vous l'appelez à partir d'un autre gestionnaire. Par exemple, l'instruction `put(findColor())` appelle le gestionnaire `on findColor`, puis affiche le résultat dans la fenêtre Messages.

Listes linéaires et listes de propriétés

Dans vos scripts, vous pouvez choisir de faire le suivi et la mise à jour de listes de données, telles qu'une série de noms ou les valeurs affectées à un ensemble de variables. Lingo et la syntaxe JavaScript ont tous deux accès aux listes linéaires et aux listes de propriétés. Dans une liste linéaire, chaque élément est une valeur unique. Dans une liste de propriétés, chaque élément contient deux valeurs ; la première est un nom de propriété, la seconde est la valeur associée à cette propriété.

Vu que Lingo et la syntaxe JavaScript ont tous deux accès aux listes linéaires et aux listes de propriétés, il est recommandé d'utiliser les listes linéaires et les listes de propriétés si les valeurs de votre code sont partagées par les scripts Lingo et JavaScript.

Si certaines valeurs de votre code sont utilisées uniquement dans les scripts de la syntaxe JavaScript, nous vous recommandons d'utiliser des objets Tableau de JavaScript avec vos listes de données. Pour plus d'informations sur l'utilisation des tableaux, consultez *Tableaux de la syntaxe JavaScript*, page 50.

Création de listes linéaires

Pour créer une liste linéaire, utilisez l'une des méthodes suivantes :

- En Lingo, utilisez soit la fonction de haut niveau `list()` ou l'opérateur de liste (`[ ]`), et séparez les éléments de la liste par des virgules.
- Dans la syntaxe JavaScript, utilisez la fonction de haut niveau `list()`, et séparez les éléments de la liste par des virgules.

L'index d'une liste linéaire commence toujours par 1.

Lorsque vous utilisez la fonction de haut niveau `list()`, vous spécifiez que les éléments de la liste sont des paramètres de la fonction. Cette fonction peut s'avérer pratique si vous utilisez un clavier ne possédant pas de touches de crochets.

Les instruction suivantes créent toutes une liste linéaire de trois noms et l'affectent à une variable.

```
--Syntaxe Lingo
listeCollaborateurs = ["Raymond", "Françoise", "Paul"] - en utilisant
  l'opérateur de liste de Lingo
listeCollaborateurs = list("Raymond", "Françoise", "Paul") - en utilisant
  list()

// Syntaxe JavaScript
var listeCollaborateurs = list("Raymond", "Françoise", "Paul"); // en
  utilisant list()
```

Vous pouvez également créer des listes linéaires vides. Les instructions suivantes créent des listes linaires vides.

```
--Syntaxe Lingo
listeCollaborateurs = [] - en utilisant l'opérateur de liste Lingo
listeCollaborateurs = list() - en utilisant list() sans paramètre

// Syntaxe JavaScript
var listeCollaborateurs = list(); // en utilisant list() sans paramètre
```

Création de listes de propriétés

Pour créer une liste de propriétés, effectuez l'une des opérations suivantes :

- En Lingo, utilisez soit la fonction de haut niveau `propList()` soit l'opérateur de liste (`[:]`). Lorsque vous utilisez l'opérateur de liste pour créer une liste de propriétés, vous pouvez utiliser soit un deux-points pour désigner des éléments de nom/valeur et des virgules pour séparer les élément de la liste soit des virgules pour désigner des éléments de nom/valeur et séparer les éléments dans la liste.
- Dans la syntaxe JavaScript, utilisez la fonction de haut niveau `propList()` et insérez des virgules pour désigner des éléments de nom/valeur et séparer les éléments dans la liste.

Lorsque vous utilisez la fonction de haut niveau `propList()`, vous spécifiez que les éléments de la liste de propriétés sont des paramètres de la fonction. Cette fonction peut s'avérer pratique si vous utilisez un clavier ne possédant pas de touches de crochets.

Les propriétés peuvent apparaître plusieurs fois dans une liste de propriétés donnée.

Les instruction suivantes créent toutes une liste de propriétés de quatre noms—`left`, `top`, `right` et `bottom`—et leurs valeurs correspondantes.

```
--Syntaxe Lingo
locImage-objet1 = [#left:100, #top:150, #right:300, #bottom:350]
locImage-objet1 = ["left",400,"top",550, "righ",500, "bottom",750]
locImage-objet1 = propList("left",400, "top",550, "right",500, "bottom",750)

// Syntaxe JavaScript
var locImage-objet1 = propList("left",400, "top",550, "right",500,
  "bottom",750);
```

Vous pouvez également créer des listes de propriétés vides. Les instructions suivantes créent des listes de propriétés vides.

```
--Syntaxe Lingo
locImage-objet1 = [:] -- en utilisant l'opérateur de liste de propriétés Lingo
locImage-objet1 = propList() - en utilisant propList() sans paramètre

// Syntaxe JavaScript
var locImage-objet1 = propList(); // en utilisant propList() sans paramètre
```

Définition et récupération d'éléments de listes

Vous pouvez définir et récupérer des éléments individuels d'une liste. La syntaxe est différente en fonction du type de liste.

Pour définir une valeur dans une liste linéaire, effectuez l'une des opérations suivantes :

- Utilisez l'opérateur égal à (=).
- Utilisez la méthode `setAt()`.

Les instructions suivantes montrent comment définir la liste linéaire `listeCollaborateurs` contenant une valeur, `Françoise`, puis ajoute `Paul` en tant que seconde valeur dans la liste.

```
--Syntaxe Lingo
listeCollaborateurs = ["Françoise"] - définit une liste linéaire
listeCollaborateurs[2] = "Paul" - définit la seconde valeur à l'aide de
  l'opérateur égal à
listeCollaborateurs.setAt(2, "Paul") -- définit la seconde valeur à l'aide de
  setAt()

// Syntaxe JavaScript
var listeCollaborateurs = liste("Françoise"); // définit une liste linéaire
listeCollaborateurs[2] = "Paul"; // définit la seconde valeur à l'aide de
  l'opérateur égal à.
listeCollaborateurs.setAt(2, "Paul"); // définit la seconde valeur à l'aide de
  setAt()
```

Pour récupérer une valeur d'une liste linéaire :

- Utilisez la variable de la liste suivie du numéro indiquant la position de la valeur dans la liste. Encadrez ce nombre de crochets.
- Utilisez la méthode `getAt()`.

L'instruction suivante crée une liste linéaire `listeCollaborateurs` puis attribue la seconde valeur de la liste à la variable `nom2`.

```
--Syntaxe Lingo
listeCollaborateurs = ["Raymond", "Françoise", "Paul"] - définit une liste
  linéaire
nom2 = listeCollaborateurs[2] -- utilisez un accès par crochets pour récupérer
  "Françoise"
nom2 = listeCollaborateurs.getAt(2) -- utilisez getAt() pour récupérer
  "Françoise"

// Syntaxe JavaScript
var listeCollaborateurs = list("Raymond", "Françoise", "Paul");
var nom2 = listeCollaborateurs[2] // utilisez un accès par crochets pour
  récupérer "Françoise"
var nom2 = listeCollaborateurs.getAt(2) // utilisez getAt() pour récupérer
  "Françoise"
```

Pour définir une valeur dans une liste de propriétés, effectuez l'une des opérations suivantes :

- Utilisez l'opérateur égal à (=).
- En Lingo seulement, utilisez la méthode `setaProp()`.
- Utilisez la syntaxe à points.

L'instruction Lingo suivante utilise l'opérateur égal à pour faire de `soupe` la nouvelle valeur associée à la propriété `Raymond`.

```
--Syntaxe Lingo
listeNourriture = [:] -- définit une liste de propriétés vide
listeNourriture[#Raymond] = "soupe" - associe soupe à Raymond
```

L'instruction Lingo suivante utilise `setaprop()` pour faire de `soupe` la nouvelle valeur associée à la propriété `Raymond`.

```
--Syntaxe Lingo
listeNourriture = [:] -- définit une liste de propriétés vide
listeNourriture.setaProp(#Raymond, "soupe") -- utilisez setaProp()

// Syntaxe JavaScript
listeNourriture = propList() - définit une liste de propriétés vide
listeNourriture.setaProp("Raymond", "soupe") -- utilisez setaProp()
```

L'instruction suivante utilise la syntaxe à points pour définir la valeur associée à `Raymond` de `soupe` à `teriyaki`.

```
--Syntaxe Lingo
listeNourriture = [#Raymond:"soupe"] - définit une liste de propriétés
trace(listeNourriture) -- affiche [#Raymond: "soupe"]
listeNourriture.Raymond = "teriyaki" - utilisez la syntaxe à points pour
  définir la valeur de Raymond
trace(listeNourriture) -- affiche [#Raymond: "teriyaki"]

// syntaxe JavaScript
var listeNourriture = propList("Raymond", "soupe"); // définit une liste de
  propriété
trace(listeNourriture); // affiche ["Raymond": "soupe"]
listeNourriture.Bruno = "teriyaki"; // utilisez la syntaxe à points pour
  définir la valeur de Raymond
trace(listeNourriture); // affiche [#Raymond: "teriyaki"]
```

Pour récupérer une valeur d'une liste de propriétés, effectuez l'une des opérations suivantes :

- Utilisez la variable de la liste, suivie du nom de la propriété associée à cette valeur. Encadrez cette propriété de crochets.
- Utilisez la méthode `getaProp()` ou `getPropAt()`.
- Utilisez la syntaxe à points.

Les instructions suivantes utilisent un accès par crochets pour récupérer les valeurs associées aux propriétés `petitdéjeuner` et `déjeuner`.

```
--Syntaxe Lingo
-- définir une liste de propriétés
listeNourriture = [#petitdéjeuner:"Gauffres", #petitdéjeuner:"Hamburger"]
trace(listeNourriture[#petitdéjeuner]) -- affiche"Gauffres"
trace(listeNourriture[#petitdéjeuner]) -- affiche "Hamburger"

// Syntaxe JavaScript
// définit une liste de propriétés
var listeNourriture = propList("petitdéjeuner", "Gauffres", "déjeuner",
  "Hamburger");
trace(listeNourriture["petitdéjeuner"]); // affiche Gauffres
trace(listeNourriture["déjeuner"]); // affiche Hamburger
```

Les instructions suivantes utilisent `getaProp()` pour récupérer la valeur associée à la propriété `petitdéjeuner`, et `getPropAt()` pour récupérer la propriété à la seconde position d'index de la liste.

```
--Syntaxe Lingo
-- définir une liste de propriétés
listeNourriture = [#petitdéjeuner:"Gauffres", #déjeuner:"Hamuburger"]
trace(listeNourriture.getaProp(#petitdéjeuner)) -- affiche "Gauffres"
trace(listeNourriture.getPropAt(2)) -- affiche #déjeuner

// Syntaxe JavaScript
// définir une liste de propriétés
var listeNourriture = propList("petitdéjeuner", "Gauffres", "déjeuner",
  "Hamburger");
trace(listeNourriture.getaProp("petitdéjeuner")); // affiche Gauffres
trace(listeNourriture.getPropAt(2)); // affiche déjeuner
```

Les instructions suivantes utilisent la syntaxe à points pour accéder aux valeurs associées aux propriétés dans la liste de propriétés.

```
--Syntaxe Lingo
-- définir une liste de propriétés
listeNourriture = [#petitdéjeuner:"Gauffres", #déjeuner:"Hamburger"]
trace(listeNourriture.petitdéjeuner) -- affiche "Gauffres"

// Syntaxe JavaScript
// définir une liste de propriétés
var listeNourriture = propList("petitdéjeuner", "Gauffres", "déjeuner",
  "Hamburger");
trace(listeNourriture.déjeuner); // affiche Hamburger
```

Vérification des éléments de listes

Vous pouvez déterminer les caractéristiques d'une liste et le nombre d'éléments qu'elle contient en utilisant les méthodes suivantes.

- Pour afficher le contenu d'une liste, utilisez la fonction `put()` ou `trace()` en passant la variable contenant la liste en tant que paramètre.
- Pour déterminer le nombre d'éléments contenus dans une liste, utilisez la méthode `count()` (Lingo seulement) ou la propriété `count`.
- Pour déterminer le type d'une liste, utilisez la méthode `ilk()`.
- Pour déterminer la valeur maximale d'une liste, utilisez la méthode `max()`.
- Pour déterminer la valeur minimale d'une liste, utilisez la fonction `min()`.
- Pour déterminer la position d'une propriété spécifique, utilisez la commande `findPos`, `findPosNear` ou `getOne`.

Les instructions suivantes utilisent `count()` et `count` pour afficher le nombre d'éléments d'une liste.

```
--Syntaxe Lingo
listeCollaborateurs = ["Raymond", "Françoise", "Paul"] - définit une liste
  linéaire
trace(listeCollaborateurs.count()) -- affiche 3
trace(listeCollaborateurs.count) -- affiche 3

// Syntaxe JavaScript
var listeCollaborateurs = list("Raymond", "Françoise", "Paul"); // définit une
  liste linéaire
trace(listeCollaborateurs.count); // affiche 3
```

Les instructions suivantes utilisent `ilk()` pour déterminer le type d'une liste.

```
--Syntaxe Lingo
x = ["1", "2", "3"]
trace(x.ilk()) // renvoie #liste

// Syntaxe JavaScript
var x = list("1", "2", "3");
trace(x.ilk()) // renvoie #liste
```

Les instructions suivantes utilisent `max()` et `min()` pour déterminer les valeurs maximales and minimales d'une liste.

```
--Syntaxe Lingo
listeCollaborateurs = ["Raymond", "Françoise", "Paul"] - définit une liste
  linéaire
trace(listeCollaborateurs.max()) -- affiche "Françoise"
trace(listeCollaborateurs.min()) -- affiche "Raymond"

// Syntaxe JavaScript
var listeCollaborateurs = list("Raymond", "Françoise", "Paul"); // définit une
  liste linéaire
trace(listeCollaborateurs.max()); // affiche Françoise
trace(listeCollaborateurs.min()); // affiche Raymond
```

Les instructions suivantes utilisent `findPos` pour obtenir la position d'index d'une propriété spécifiée dans une liste de propriétés.

```
--Syntaxe Lingo
-- définit une liste de propriétés
listeNourriture = [#petitdéjeuner:"Gauffres", #déjeuner:"Hamburger"]
trace(listeNourriture.findPos(#déjeuner)) -- affiche 2

// Syntaxe JavaScript
// définit une liste de propriétés
var listeNourriture = propList("petitdéjeuner", "Gauffres", "déjeuner",
  "Hamburger");
trace(listeNourriture.findPos("petitdéjeuner")); // affiche 1
```

Ajout et suppression d'éléments de listes

Vous pouvez ajouter des éléments à une liste ou en supprimer à l'aide des méthodes suivantes.

- Pour ajouter un élément à la fin d'une liste, utilisez la méthode `append()`.
- Pour ajouter un élément à l'endroit correct dans une liste triée, utilisez la méthode `add()` ou `addProp()`.
- Pour ajouter un élément à un emplacement spécifique d'une liste linéaire, utilisez la méthode `addAt()`.
- Pour ajouter un élément à un emplacement spécifique d'une liste de propriétés, utilisez la méthode `addProp()`.
- Pour supprimer un élément d'une liste, utilisez la méthode `deleteAt()`, `deleteOne()`, ou `deleteProp`().
- Pour remplacer un élément d'une liste, utilisez le méthode `setAt()` ou `setaProp()`.

Les instructions suivante utilisent `append()` pour ajouter un élément à la fin d'une liste.

```
--Syntaxe Lingo
listeCollaborateurs = ["Raymond", "Françoise", "Paul"] - définit une liste
  linéaire
listeCollaborateurs.append("David")
trace(listeCollaborateurs) -- affiche ["Raymond", "Françoise", "Paul",
  "David"]

// Syntaxe JavaScript
var listeCollaborateurs = list("Raymond", "Françoise", "Paul"); // définit une
  liste linéaire
listeCollaborateurs.append("David");
trace(listeCollaborateurs); // affiche ["Raymond", "Françoise", "Paul",
  "David"]
```

Les instructions suivantes utilisent `addProp()` pour ajouter une propriété et une valeur associée à une liste de propriétés.

```
--Syntaxe Lingo
-- définit une liste de propriétés
listeNourriture = [#petitdéjeuner:"Gauffres", #déjeuner:"Hamburger"]
listeNourriture.addProp(#dîner, "Spaghetti") -- ajoute [#dîner: "Spaghetti"]

// Syntaxe JavaScript
// définit une liste de propriétés
var listeNourriture = propList("petitdéjeuner", "Gauffres", "déjeuner",
  "Hamburger");
listeNourriture.addProp("dîner", "Spaghetti"); // ajoute ["dîner":
  "Spaghetti"]
```

Il n'est pas nécessaire d'éliminer explicitement les listes. Elles sont automatiquement supprimées lorsque aucune variable n'y fait référence. Les autres types d'objets doivent être retirés de manière explicite, en donnant aux variables qui y font référence la valeur `VOID` (Lingo) ou `null` (syntaxe JavaScript).

Copie de listes

L'affectation d'une liste à une variable, puis l'affectation de cette variable à une autre variable ne crée pas automatiquement une copie de cette liste. Par exemple, la première instruction ci-dessous crée une liste contenant les noms de deux continents et affecte la liste à la variable `listeTerres`. La seconde instruction affecte la même liste à une nouvelle variable `listeContinents`. Dans la troisième instruction, l'ajout d' `Australie` à `listeTerres` ajoute automatiquement l'élément `Australie` à la liste `listeContinents`. La raison en est que les deux noms de variables font référence au même objet en mémoire. Le même comportement se produit lorsque vous utilisez un tableau dans la syntaxe JavaScript.

```
--Syntaxe Lingo
listeTerres = ["Asie", "Afrique"]
listeContinents = listeTerres
listeTerres.add("Australie") - ceci ajoute également "Australie" à
  listeContinents
```

Pour créer une copie d'une liste indépendante d'une autre liste :

- Utilisez la méthode `duplicate()`.

Par exemple, les instructions suivantes créent une liste puis font une copie indépendante de la liste.

```
--Syntaxe Lingo
ancienneListe = ["a", "b", "c"]
nouvelleListe = ancienneListe.duplicate() - fait une copie indépendante de
  ancienneListe

// Syntaxe JavaScript
var ancienneListe = list("a", "b", "c");
var nouvelleListe = ancienneListe.duplicate(); // crée une copie indépendante
  de ancienneListe
```

Une fois `nouvelleListe` créée, la modification de `ancienneListe` ou de `nouvelleListe` n'a aucun effet sur l'autre liste.

Tri des listes

Les listes sont triées dans l'ordre alphanumérique, les nombres étant triés avant les chaînes. Les chaînes sont triées en fonction de la première lettre, quel que soit le nombre de caractères qu'elles contiennent. Les listes triées sont traitées un peu plus rapidement que les listes non triées.

Une liste linéaire est triée en fonction des valeurs de la liste. Une liste de propriétés est triée en fonction des noms de propriétés de la liste ou du tableau.

Une fois les valeurs d'une liste linéaire ou d'une liste de propriétés triées, elles restent triées même si des valeurs sont ajoutées ou supprimées des listes.

Pour trier une liste :

- Utilisez la méthode `sort()`.

Par exemple, les instructions suivantes trient une liste alphabétique non triée.

```
--Syntaxe Lingo
ancienneListe = ["d", "a", "c", "b"]
ancienneListe.sort() -- renvoie ["a", "b", "c", "d"]

// Syntaxe JavaScript
var ancienneListe = list("d", "a", "c", "b");
ancienneListe.sort(); // renvoie ["a", "b", "c", "d"]
```

Création de listes multidimensionnelles

Vous pouvez également créer des listes multidimensionnelles qui vous permettront d'utiliser les valeurs de plusieurs listes à la fois.

Dans l'exemple suivant, les deux premières instructions créent les listes linéaires séparées `liste1` et `liste2`. La troisième instructions crée une liste multidimensionnelle et l'affecte à `listeMd`. Dans une liste multidimensionnelle, les instructions quatre et cinq utilisent des crochets pour accéder aux valeurs de la liste ; le premier crochet donne accès à une liste donnée, le second crochet donnant accès à la valeur se trouvant à la position d'index spécifiée dans la liste.

```
--Syntaxe Lingo
liste1 = list(5,10)
liste2 = list(15,20)
listeMd = list(liste1, liste2)
trace(listeMd[1][2]) -- affiche 10
trace(listeMd[2][1]) - affiche 15

// Syntaxe JavaScript
var liste1 = list(5,10);
var liste2 = list(15,20);
var listeMd = list(liste1, liste2);
trace(listeMd[1][2]); // affiche 10
trace(listeMd[2][1]); // affiche 15
```

Tableaux de la syntaxe JavaScript

Les tableaux de la syntaxe JavaScript sont similaires aux listes linéaires de Lingo, chaque élément d'un tableau étant une valeur unique. L'une des principales différences entre les tableaux de la syntaxe JavaScript et les listes linéaires de Lingo est que l'index d'un tableau commence toujours par 0.

Vous pouvez créer un tableau de la syntaxe JavaScript en utilisant l'objet Tableau. Vous pouvez utiliser soit des crochets (`[ ]`) soit le constructeur `Tableau` pour créer un tableau. Les deux instructions suivantes créent un tableau de deux valeurs.

```
// Syntaxe JavaScript
var monTableau = [10, 15]; // en utilisant des crochets
var monTableau = new Array(10, 15); // en utilisant le constructeur Tableau
```

Vous pouvez créer des tableaux vides. Les deux instructions suivantes créent un tableau vide.

```
// Syntaxe JavaScript
var monTableau = [];
var monTableau = new Array();
```

Remarque : La *Référence de scripting de Director* ne constitue pas une référence complète pour les objets Tableau de la syntaxe JavaScript. Pour plus d'informations concernant les objets Tableau, consultez les nombreuses autres ressources consacrées à ce sujet.

Vérification d'éléments dans les tableaux

Vous pouvez déterminer les caractéristiques d'un tableau et le nombre d'éléments qui y sont contenus en utilisant les méthodes suivantes.

- Pour afficher le contenu d'une liste, utilisez la fonction `put()` ou `trace()`, en passant la variable contenant la liste en tant que paramètre.
- Pour déterminer le nombre d'éléments contenus dans un tableau, utilisez la propriété `length` de l'objet Tableau.
- Pour déterminer le type d'un tableau, utilisez la propriété `constructor`.

  L'exemple suivant montre comment déterminer le nombre d'éléments contenus dans un tableau en utilisant la propriété `length` puis en renvoyant le type d'objet à l'aide de la propriété `constructor`.

```
// Syntaxe JavaScript
var x = ["1", "2", "3"];
trace(x.length); // affiche 3
trace(x.contructor == Array); // affiche true
```

Ajout et suppression d'éléments dans des tableaux

Vous pouvez ajouter des éléments à un tableau ou en supprimer à l'aide des méthodes suivantes.

- Pour ajouter un élément à la fin d'un tableau, utilisez la méthode `push()` de l'objet Tableau.
- Pour placer un élément à sa position correcte dans un tableau trié, utilisez la méthode `splice()` de l'objet Tableau.
- Pour placer un élément à un endroit précis d'un tableau, utilisez la méthode `splice()` de l'objet Tableau.
- Pour supprimer un élément d'un tableau, utilisez la méthode `splice()` de l'objet Tableau.
- Pour remplacer un élément d'un tableau, utilisez la méthode `splice()` de l'objet Tableau.

L'exemple suivant montre comment utiliser la méthode `splice()` de l'objet Tableau pour ajouter des éléments à un tableau, les supprimer du tableau ou les remplacer.

```
// Syntaxe JavaScript
var monTableau = new Array("1", "2");
trace(monTableau); affiche 1,2

myArray.push("5"); // ajoute la valeur "5" à la fin de monTableau
trace(monTableau); // affiche 1,2,5

myArray.splice(3, 0, "4"); // ajoute la valeur "4" après la valeur "5"
trace(monTableau); // affiche 1,2,5,4

myArray.sort(); // trie monTableau
trace(monTableau); // affiche 1,2,4,5

myArray.splice(2, 0, "3");
trace(monTableau); // affiche 1,2,3,4,5
```

```
myArray.splice(3, 2); // supprime deux valeurs aux positions d'index 3 et 4
trace(monTableau); // affiche 1,2,3

myArray.splice(2, 1, "7"); // remplace une valeur à la position d'index 2 par
  "7"
trace(monTableau); affiche 1,2,7
```

Copie de tableaux

L'affectation d'un tableau à une variable, puis l'affectation de cette variable à une autre variable ne crée pas une copie séparée de ce tableau.

Par exemple, la première instruction ci-dessous crée un tableau contenant les noms de deux continents puis affecte le tableau à la variable `listeTerres`. La seconde instruction affecte la même liste à une nouvelle variable `listeContinents`. Dans la troisième instruction, l'ajout d'`Australie` à `listeTerres` ajoute automatiquement l'élément `Australie` au tableau `listeContinents`. En effet, les deux noms de variables font référence au même objet Tableau en mémoire.

```
// Syntaxe JavaScript
var listeTerres = new Array("Asie", "Afrique");
var listeContinents = listeTerres;
listeTerres.push("Australie"); // ceci ajoute également "Australie" à
  listeContinents
```

Pour créer la copie d'un tableau indépendant d'un autre tableau :

- Utilisez la méthode `slice()` de l'objet Tableau.

Par exemple, les instructions suivantes créent un tableau puis utilisent `slice()` pour créer une copie indépendante du tableau.

```
// Syntaxe JavaScript
var ancienTableau = ["a", "b", "c"];
var nouveauTableau = ancienTableau.slice(); // crée une copie indépendante de
  ancienTableau
```

Une fois `nouveauTableau` créé, la modification de `ancienTableau` ou de `nouveauTableau` n'a aucun effet sur l'autre tableau.

Tri des tableaux

Les tableaux sont triés dans l'ordre alphanumérique, les nombres étant triés avant les chaînes. Les chaînes sont triées en fonction de la première lettre, quel que soit le nombre de caractères qu'elles contiennent.

Pour trier un tableau :

- Utilisez la méthode `sort()` de l'objet Tableau.

Les instructions suivantes trient un Tableau alphabétique non trié.

```
// Syntaxe JavaScript
var ancienTableau = ["d", "a", "c", "b"];
ancienTableau.sort(); // renvoie a, b, c, d
```

Les instructions suivantes trient un tableau alphanumérique non trié

```
// Syntaxe JavaScript
var ancienTableau = [6, "f", 3, "b"];
ancienTableau.sort(); // renvoie 3, 6, b, f
```

Le tri d'un tableau renvoie un nouveau tableau trié.

Création de tableaux multidimensionnels

Vous pouvez également créer des tableaux multidimensionnels qui vous permettront d'utiliser les valeurs de plusieurs tableaux à la fois.

Dans l'exemple suivant, les deux premières instructions créent les tableaux distincts `tableau1` et `tableau2`. La troisième instruction crée un tableau multidimensionnel et l'affecte à `tableauMd`. Pour accéder aux valeurs d'un tableau multidimensionnel, les instructions quatre et cinq utilisent des crochets ; le premier crochet donne accès à un tableau précis, le second donnant accès à une valeur située à une position d'index spécifiée dans le tableau.

```
// Syntaxe JavaScript
var tableau1 = nouveau Tableau(5,10);
var tableau2 = [15,20];
var tableauMd = nouveau Tableau(tableau1, tableau2);
trace(tableauMd[0][1]); // affiche 10
trace(tableauMd[1][0]); // affiche 15
```

CHAPITRE 3
Rédaction de scripts dans Director

Les scripts de Macromedia Director MX 2004 prennent en charge toutes sortes de fonctionnalités pour des animations qui seraient impossibles à réaliser sans eux. Au fur et à mesure de la rédaction des scripts, vous vous rendrez probablement compte que des scripts de plus en plus compliqués sont nécessaires pour prendre en charge l'interactivité complexe de vos animations Director. Nous vous présentons ici les concepts et techniques de scripting intermédiaires et avancés, notamment des informations concernant le scripting orienté objet dans Director.

Si vous êtes un novice en matière de scripting dans Director, assurez-vous de lire le Chapitre 2, *Principes de base du scripting dans Director*, page 9 en plus des rubriques suivantes.

Choisir entre Lingo et la syntaxe JavaScript

Lingo et JavaScript donnent tous deux accès aux mêmes objets, événements et API de scripting. Le langage dans lequel vous choisissez de rédiger vos scripts importe donc peu. Il vous suffit de décider du langage qui vous convient le mieux.

Voici quelques idées générales concernant le fonctionnement des langages de scripting avec un objet et modèle d'événement donnés dans Director :

- En général, un langage de scripting tel que Lingo ou JavaScript entoure un modèle d'objet et d'événement donné et donne accès à ces objets et événements.
- JavaScript est une implémentation de ECMAScript qui entoure le modèle d'objet et d'événement d'un navigateur web et donne accès aux objets et événements du navigateur.
- ActionScript est une implémentation de ECMAScript qui entoure le modèle d'objet et d'événement de Macromedia Flash et donne accès aux objets et événements de Flash.
- L'implémentation Director de JavaScript est une implémentation de ECMAScript qui entoure le modèle d'objet et d'événement de Director et donne accès aux objets et événements de Director.
- Lingo est une syntaxe personnalisée qui entoure le modèle d'objet et d'événement de Director et donne accès aux objets et événements de Director.

Lingo et JavaScript sont tout simplement les deux langages que vous pouvez utiliser pour accéder au même modèle d'objet etévénement dans Director. Les scripts rédigés dans l'un des langages ont les mêmes capacités que ceux rédigés dans l'autre.

Une fois que vous avez donc décidé de la manière d'accéder aux API de scripting dans un langage, vous saurez comment y accéder dans l'autre. Par exemple, le code de la syntaxe JavaScript peut accéder à des types de données Lingo tels que les symboles, les listes linéaires, les listes de propriétés et ainsi de suite, créer et invoquer des scripts et comportements parents de Lingo, créer et invoquer des Xtras et utiliser des expressions de sous-chaînes Lingo. De plus, les scripts de la syntaxe JavaScript et Lingo peuvent être utilisés au sein d'une animation ; toutefois, un acteur script ne peut contenir qu'un type de syntaxe.

Il existe deux différences principales entre Lingo et la syntaxe JavaScript :

- Chaque langage comprend des conventions de syntaxe et de terminologie qui leur sont propres. Par exemple, la syntaxe d'un gestionnaire d'événement dans Lingo est différente de celle de JavaScript.

```
-- Syntaxe Lingo
on mouseDown
  ...
end

// Syntaxe JavaScript
function mouseDown() {
  ...
}
```

  Pour plus d'informations concernant les conventions de syntaxe et de terminologie utilisées pour chaque langage, veuillez consulter *Terminologie de scripting*, page 10 et *Syntaxe de scripting*, page 13.

- L'accès à certains API de scripting varie légèrement d'un langage à l'autre. Par exemple, vous utiliseriez des constructions différentes pour accéder au deuxième mot du premier paragraphe d'un acteur texte :

```
-- Syntaxe Lingo
member("Nouvelles du jour").paragraph[1].word[2]

// Syntaxe JavaScript
member("Nouvelles du jour").getPropRef("paragraph", 1).getProp("word", 2);
```

Format de scripting à syntaxe à points

Que vous rédigiez des scripts avec Lingo ou en utilisant la syntaxe JavaScript, vous le faites à l'aide du format de scripting à syntaxe à points. Vous utilisez la syntaxe à points pour accéder aux propriétés ou méthodes liées à un objet. Une instruction utilisant la syntaxe à points commence par le nom de l'objet, suivi d'un point, puis du nom de la propriété, de la méthode ou de la sous-chaîne de texte que vous souhaitez spécifier. Chaque point d'une instruction représente un mouvement allant d'un niveau plus élevé et plus général de la hiérarchie des objets à un niveau inférieur et plus précis.

Par exemple, l'instruction suivante crée d'abord une référence à la bibliothèque de distribution appelée « Nouvelles du jour » puis utilise la syntaxe à points pour accéder au nombre d'acteurs compris dans cette bibliothèque.

```
-- Syntaxe Lingo
castLib("Nouvelles du jour").member.count

// Syntaxe JavaScript
castLib("Nouvelles du jour").member.count;
```

Pour identifier les sous-chaînes, les termes suivant le point servent à indiquer des éléments spécifiques du texte. Par exemple, la première instruction ci-dessous concerne le premier paragraphe de l'acteur texte appelé « Nouveaux Eléments ». La seconde instruction concerne le deuxième mot du premier paragraphe.

```
-- Syntaxe Lingo
member("Nouveaux Eléments").paragraph[1]
member("Nouveaux Eléments").paragraph[1].word[2]

// Syntaxe JavaScript
member("Nouveaux Eléments").getPropRef("paragraph", 1);
member("Nouveaux Eléments").getPropRef("paragraph", 1).getProp("word", 2);
```

Pour certains objets prenant en charge l'accès des propriétés en cascade à des données ou à un type d'acteur précis, comme indiqué dans les deux instructions précédentes, la syntaxe JavaScript normale ne supporte pas ce genre d'accès. Vous devez donc utiliser les méthodes `getPropRef()` et `getProp()` pour accéder aux propriétés de cascade dans la syntaxe JavaScript.

Voici quelques remarques concernant cette exception dans la syntaxe JavaScript :

- Vous devez appliquer cette technique aux objets 3D, aux acteurs texte, aux acteurs champ et aux Xtras XML Parser auxquels vous avez accédé à l'aide de la syntaxe JavaScript.
- Vous devez utiliser la méthode `getPropRef()` pour stocker une référence à l'un des objets précédemment mentionnés ou à ses propriétés à l'aide de la syntaxe JavaScript.
- Vous devez utiliser la méthode `getProp()` pour récupérer une valeur de propriété de l'un des objets précédemment mentionnés ou de ses propriétés à l'aide la syntaxe JavaScript.
- Vous devez accéder aux objets 3D et propriétés en utilisant leurs noms complets dans la syntaxe JavaScript. Dans Lingo par exemple, la propriété `shader` peut être utilisée comme un raccourci de la propriété `shaderList[1]`. Toutefois, dans la syntaxe JavaScript, la propriété `shaderList[1]` doit être utilisée tout le temps.

Introduction aux objets de Director

Les objets sont généralement des groupes logiques de données nommées pouvant également contenir des méthodes agissant sur ces données. Dans cette version de Director, les API de scripting ont été groupés en objets et sont accessibles à travers ces objets. Chaque objet donne accès à un ensemble précis de données nommées et de types de fonctionnalités. Par exemple, l'objet Image-objet donne accès aux données et fonctionnalités d'une image-objet, l'objet Animation donne accès aux données et fonctionnalités d'une animation, et ainsi de suite.

Les objets utilisés dans Director appartiennent aux quatre catégories suivantes. Selon la fonctionnalité que vous voulez ajouter et la partie d'une animation à laquelle vous l'ajoutez, vous utiliserez les objets d'une ou plusieurs de ces catégories :

- Objets principaux
- Types de médias
- Objets de scripting
- Objets 3D

Objets principaux

Cette catégorie d'objets donne accès aux principales fonctionnalités et fonctions de Director, telles que le moteur du lecteur de Director, les fenêtres des animations, les images-objets, les sons, etc. Ces objets représentent la couche de base à travers laquelle on accède à tous les API et autres catégories d'objets.

Il existe également un groupe de méthodes et propriétés de haut niveau qui vous permettent d'accéder directement à tous les objets principaux au lieu d'avoir à passer par la hiérarchie des objets.

Pour plus d'informations sur les objets principaux et leurs API, consultez le Chapitre 5, *Objets principaux de Director*, page 113.

Types de médias

Cette catégorie d'objets donne accès aux fonctionnalités des divers types de médias (RealMedia, DVD, GIF animé, etc.) ajoutés aux animations en tant qu'acteurs.

Les médias ne sont pas strictement des objets mais plutôt des acteurs qui se rapportent à un type de média précis. Lorsqu'un type de média est ajouté à une animation en tant qu'acteur, il hérite de la fonctionnalité de l'objet Acteur principal et étend l'objet Acteur en fournissant des fonctionnalités supplémentaires qui ne sont disponibles que pour le type de média spécifié. Par exemple, un acteur RealMedia a accès aux méthodes et propriétés de l'objet Acteur, et possède également d'autres méthodes et propriétés propres à RealMedia. Les autres types de médias affichent tous ce comportement.

Pour plus d'informations sur les types de médias disponibles et leurs API, consultez le Chapitre 6, *Types de médias*, page 131.

Objets de scripting

Cette catégorie d'objets, également connus sous le nom de Xtras, donne accès aux fonctionnalités des composants logiciels tels que XML Parser, Fileio et SpeechXtra qui sont installés dans Director et étendent les fonctionnalités principales de Director. Les Xtras existants fournissent certaines fonctions telles que l'importation de filtres et la connexion à Internet. Vous pouvez créer vos propres Xtras si vous savez programmer en langage C.

Pour plus d'informations sur les objets de scripting et leurs API, consultez le Chapitre 7, *Objets de scripting*, page 149.

Objets 3D

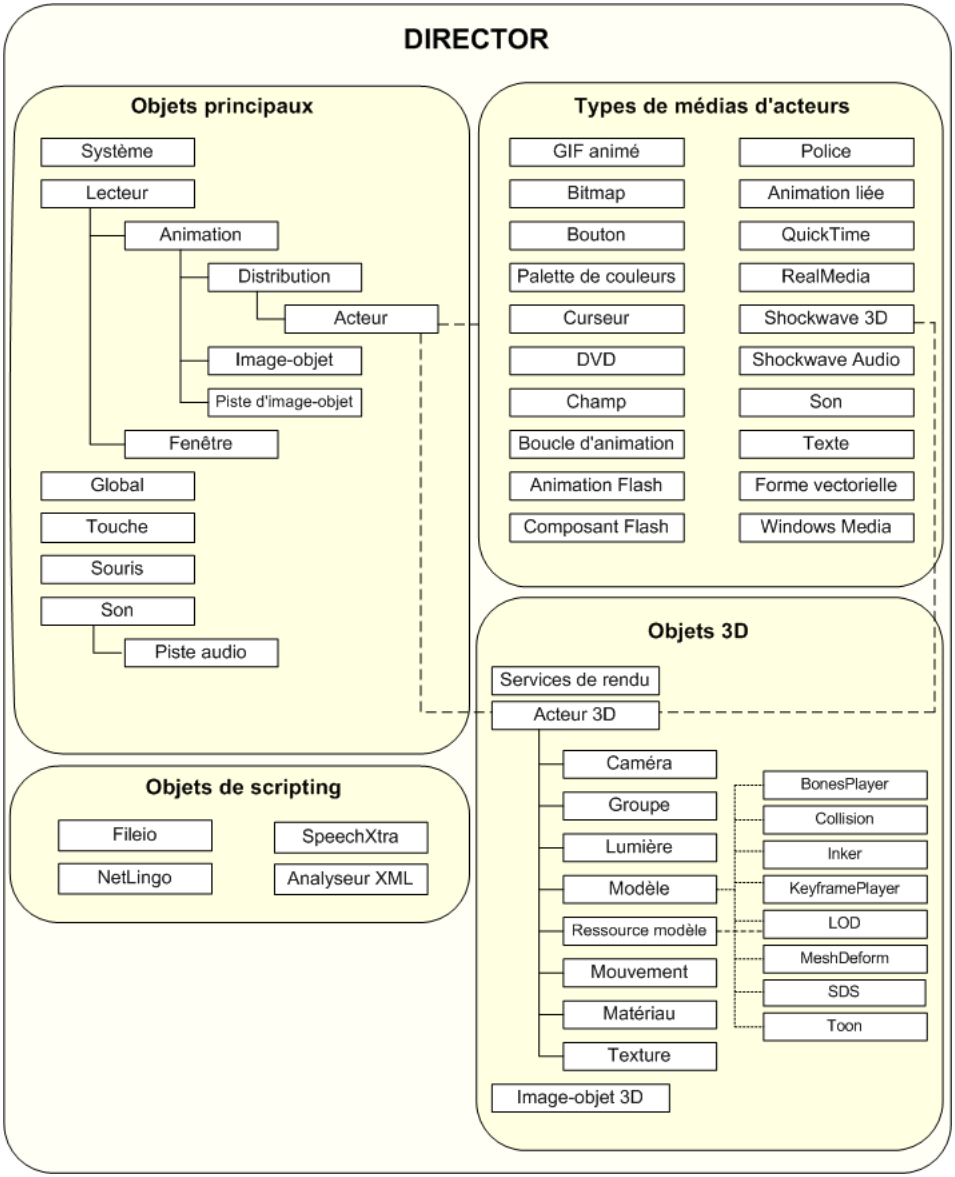
Cette catégorie d'objets donne accès aux fonctionnalités d'acteurs et de texte utilisées pour créer des animations 3D.

Pour plus d'informations sur les animations 3D, veuillez voir la rubrique Utilisation de Director dans le panneau d'aide de Director.

Pour plus d'informations sur les objets 3D et leurs API, consultez le Chapitre 8, *Objets 3D*, page 153.

Diagramme de modèles d'objets

Les diagrammes suivants illustrent les principales relations de haut niveau entre les groupes d'objets et leurs hiérarchies dans Director. Pour toute information sur la création d'objets, les propriétés et méthodes et les autres API, veuillez consulter les rubriques consacrées à chaque API.



Fonctions et propriétés de haut niveau

Il existe également un groupe de méthodes et propriétés de haut niveau qui vous permettent d'accéder directement à tous les objets principaux et fonctionnalités au sein de Director. Ces fonctions et propriétés sont très utiles pour créer des références aux objets principaux, aux nouvelles images, aux listes et ainsi de suite. Par exemple, la propriété `_movie` de haut niveau fait directement référence à l'objet principal Animation et la fonction `list()` de haut niveau crée une liste linéaire.

Les tableaux suivants dressent la liste des fonctions et propriétés de haut niveau.

Propriétés de haut niveau	
_global	_player
_key	_sound
_mouse	_system
_movie	

Fonctions de haut niveau	
castLib()	rect()
channel() (niveau supérieur)	script()
color()	showLocals()
date() (formats), date() (Système)	sound()
image()	isBusy()
list()	sprite()
member()	symbol()
point()	timeout()
propList()	trace()
put()	vector()
random()	window()
randomVector()	xtra()

Introduction à la programmation orientée objet dans Director

Vous pouvez appliquer les principes de la programmation orientée objet à vos scripts avec Lingo ou JavaScript. Cette opération facilite en général la programmation puisqu'elle vous permet de rédiger moins de codes et d'utiliser la logique pour effectuer des tâches, tout en améliorant la réutilisation et la modularité de vos codes.

En fonction du langage utilisé, vous appliquez ces principes à l'aide de deux types de paradigmes :

- Dans Lingo, vous utilisez les scripts parents, les scripts ancêtres et les objets enfants pour simuler la programmation orientée objet.
- Dans la syntaxe JavaScript, vous utilisez les techniques de programmation orientée objet de style JavaScript standard pour créer des classes et des sous-classes.

Chaque paradigme vous permet d'appliquer les avantages apportés par la programmation orientée objet à vos scripts. Ainsi, le langage de scripting que vous utilisez importe peu. Il vous suffit tout simplement d'appliquer les principes différemment.

Vu que chaque langage de scripting utilise un paradigme différent pour appliquer les principes orientés objet, les techniques valables pour un langage ne seront pas valables pour l'autre. Il vous suffit donc de consulter le contenu qui s'applique au langage de scripting que vous utilisez :

- Pour plus d'informations sur la simulation de la programmation orientée objet dans Lingo, consultez *Programmation orientée objet avec Lingo* sur cette page.
- Pour plus d'informations sur la simulation de la programmation orientée objet dans JavaScript, consultez *Programmation orientée objet avec la syntaxe JavaScript*, page 72.

Programmation orientée objet avec Lingo

Dans Lingo, les scripts parents offrent les avantages de la programmation orientée objet. Vous pouvez utiliser des scripts parents pour générer des objets scripts qui ont une réponse et un comportement similaires tout en fonctionnant indépendamment les uns des autres.

Lingo peut créer plusieurs copies (ou instances) d'un script parent. Chaque instance d'un script parent est un objet enfant. Vous pouvez créer des objets enfants sur demande au fur et à mesure de la lecture de l'animation. Director ne limite pas le nombre d'objets enfants que vous pouvez créer depuis un même script parent. Vous pouvez créer autant d'objets enfants que la mémoire de l'ordinateur peut en supporter.

Director peut créer plusieurs objets enfants depuis un même script parent de la même façon qu'il peut créer plusieurs instances d'un comportement pour différentes images-objets. Un script parent pourrait être assimilé à un modèle et un objet enfant à une implémentation du modèle parent.

Cette section présente les concepts de base concernant la rédaction des scripts parents, ainsi que la création et l'utilisation des objets enfants, et propose des exemples de script. Vous n'y trouverez pas les concepts de la programmation orientée objet, mais devrez cependant en comprendre les principes afin d'utiliser efficacement les scripts parents et les objets enfants. Vous trouverez dans le commerce de nombreux ouvrages présentant les notions de base de la programmation orientée objet.

Similarité avec les autres langages orientés objet

Si vous connaissez déjà un langage de programmation orienté objet tel que Java ou C++, vous comprenez probablement ces concepts, même s'il se peut que vous les connaissiez en d'autres termes.

Les termes par lesquels Director décrit les scripts parents et les objets enfants correspondent aux termes couramment utilisés dans la programmation orientée objet :

Les scripts parents de Director correspondent aux classes dans la programmation orientée objet.

Les objets enfants de Director correspondent aux instances dans la programmation orientée objet.

Les variables de propriétés de Director correspondent aux variables d'instances ou d'acteurs dans la programmation orientée objet.

Les gestionnaires de Director correspondent aux méthodes dans la programmation orientée objet.

Les scripts ancêtres de Director correspondent à la Super classe ou classe de base dans la programmation orientée objet.

Scripts parents et objets enfants : notions de base

Dans Lingo, un script parent contient un ensemble de gestionnaires et de propriétés définissant un objet enfant ; il ne constitue pas un objet enfant en lui-même. Un objet enfant est une instance autonome et indépendante d'un script parent. Les objets enfants du même parent possèdent des gestionnaires et des propriétés identiques, les objets enfants d'un même groupe pouvant donc avoir des réponses similaires aux événements et aux messages.

Les scripts parents sont généralement utilisés pour construire des objets enfants facilitant l'organisation de l'animation. Ces objets enfants sont particulièrement utiles lorsqu'une animation nécessite plusieurs exécutions simultanées d'une même logique avec des paramètres différents. Vous pouvez aussi ajouter un objet enfant à la propriété `scriptInstanceList` d'un objet Image-objet ou la propriété `actorList` de l'objet Animation pour contrôler les animations.

Tous les objets enfants d'un même script parent possédant des gestionnaires identiques, ces objets enfants répondent de façon semblable aux événements. Par contre, chaque objet enfant conserve ses propres valeurs pour les propriétés définies dans le script parent. Il s'en suit que chaque objet enfant peut avoir un comportement différent de celui des objets enfants apparentés tout en provenant du même script parent.

Par exemple, vous pouvez créer un script parent définissant des objets enfants de champs texte modifiables, possédant chacun leurs propres paramètres de propriété, texte et couleur, quels que soient les réglages des autres champs texte. En modifiant les valeurs des propriétés d'objets enfants précis, vous pouvez modifier ces caractéristiques pendant la lecture de l'animation, sans influer sur les autres objets enfants basés sur le même script parent.

De même, un objet enfant peut également contenir une propriété réglée sur `TRUE` ou `FALSE`, quel qu'en soit le réglage pour les objets enfants apparentés.

Un script parent se rapporte au nom d'un acteur script contenant les variables de propriétés et les gestionnaires. Un objet enfant créé à partir d'un script parent est essentiellement une nouvelle instance de l'acteur script.

Différences entre les objets enfants et les comportements

Bien que les objets enfants et les comportements soient similaires en ce sens qu'ils peuvent tous deux posséder plusieurs instances, ils présentent également plusieurs différences importantes. La principale différence entre les objets enfants et les comportements est que les comportements sont associés à des emplacements du scénario puisqu'ils sont affectés à des images-objets. Les objets comportement sont automatiquement créés à partir d'initialisateurs dans le scénario lorsque la tête de lecture passe d'une image à une autre et rencontre des images-objets possédant des comportements. Les objets enfants de scripts parents, eux, doivent être explicitement créés par un gestionnaire.

Les comportements et les objets enfants diffèrent dans la façon dont ils deviennent associés aux images-objets. Director associe automatiquement un comportement à l'image-objet à laquelle il est associé, alors que vous devez explicitement associer un objet enfant à une image-objet. Les objets enfants ne requièrent pas de références d'images-objets et n'existent que dans la mémoire.

Ancêtres : principes de base

Les scripts parents peuvent déclarer des ancêtres, des scripts supplémentaires dont un script enfant peut appeler et utiliser les gestionnaires et les propriétés.

La création de scripts ancêtres vous permet de créer un ensemble de gestionnaires et propriétés que vous pouvez utiliser et réutiliser pour plusieurs scripts parents.

Un script parent fait d'un autre script parent son ancêtre en affectant le script à sa propriété `ancestor`. Par exemple, l'instruction suivante transforme le script Ce_que_chacun_fait en ancêtre du script parent dans lequel l'instruction survient :

```
-- Syntaxe Lingo
ancestor = new(script "Ce_que_chacun_fait")
```

Lorsque les gestionnaires et les propriétés ne sont pas définis dans un objet enfant, Director recherche la propriété ou le gestionnaire requis dans les ancêtres de l'enfant, en commençant par son script parent. Si un gestionnaire est appelé ou qu'une propriété est testée et que le script parent ne contient aucune définition correspondante, Director recherche une définition dans le script ancêtre. Si ce script ancêtre contient une définition, celle-ci est utilisée.

Un objet enfant ne peut avoir qu'un ancêtre à la fois, mais ce script ancêtre peut posséder à son tour un ancêtre, qui peut également en avoir un, et ainsi de suite. Cela vous permet de créer des générations de scripts parents dont les gestionnaires sont accessibles à un objet enfant.

Rédaction d'un script parent

Un script parent contient le code nécessaire à la création d'objets enfants et en définit les actions et propriétés possibles. Vous devez d'abord décider du comportement envisagé des objets enfants. Rédigez ensuite un script parent qui effectue les opérations suivantes :

- Déclare les éventuelles variables de propriétés requises ; ces variables représentant des propriétés pour lesquelles chaque objet enfant peut contenir une valeur indépendamment des autres objets enfants.
- Définit les valeurs initiales des propriétés et des variables de l'objet enfant dans le gestionnaire `on new`.
- Contient des gestionnaires supplémentaires contrôlant les actions de l'objet enfant.

Déclaration de variables de propriétés

Chaque objet enfant créé à partir du même script parent contient, dans un premier temps, les mêmes valeurs pour ses variables de propriété. La valeur d'une variable de propriété n'appartient qu'à l'objet enfant auquel elle est associée. Chaque variable de propriété et sa valeur persistent tant que l'objet enfant existe. La valeur initiale d'une variable de propriété est généralement définie dans le gestionnaire `on new`. Si la propriété n'est pas initialement définie, sa valeur initiale est `VOID`.

Pour déclarer une variable de propriété :

- Utilisez le mot-clé `property` au début du script parent.

Pour définir et tester des variables de propriétés en dehors de l'objet enfant :

- Définissez et testez les variables de propriétés de la même façon que toute autre propriété dans vos scripts à l'aide de la syntaxe *`réfDObjet.nomDePropriété`*.

Par exemple, l'instruction suivante définit la propriété `vitesse` d'un objet `voiture1` :

```
voiture1.vitesse = 55
```

Création du nouveau gestionnaire

Chaque script parent utilise généralement un gestionnaire `on new`. Ce gestionnaire crée le nouvel objet enfant lorsqu'un autre script émet une commande `new(script nomDuScriptParent)` qui ordonne au script parent défini de créer un objet enfant basé sur lui-même. Si nécessaire, le gestionnaire `on new` du script parent peut aussi définir les valeurs initiales des propriétés de l'objet enfant.

Le gestionnaire `on new` commence toujours par l'expression `on new,`, suivie de la variable `me` et de tout paramètre communiqué au nouvel objet enfant.

Le gestionnaire `on new` suivant crée un nouvel objet enfant à partir du script parent et initialise la propriété `spriteNum` de l'enfant à l'aide de la valeur qui lui est associée dans le paramètre `aSpriteNum`. L'instruction `return me` renvoie l'objet enfant vers le gestionnaire qui a appelé initialement le gestionnaire `on new`.

```
-- Syntaxe Lingo
property spriteNum

on new me, aSpriteNum
  spriteNum = aSpriteNum
  return me
end
```

Pour plus d'informations sur l'appel des gestionnaires `on new`, veuillez consulter *Création d'un objet enfant*, page 66.

Ajout de gestionnaires supplémentaires

Le comportement de l'objet enfant est déterminé par l'inclusion des gestionnaires produisant le comportement escompté dans le script parent. Par exemple, vous pouvez ajouter un gestionnaire pour changer la couleur de l'image-objet.

Le script parent suivant définit une valeur pour la propriété `spriteNum` et contient un second gestionnaire qui modifie la propriété `foreColor` de l'image-objet.

```
-- Syntaxe Lingo
property spriteNum

on new me, aSpriteNum
  spriteNum = aSpriteNum
  return me
end

on changeColor me
  spriteNum.foreColor = random(255)
end
```

Référence à l'objet actuel

D'une manière générale, les objets enfants multiples sont créés à partir du même script parent et chacun d'eux utilise plus d'un gestionnaire. La variable de paramètre spéciale `me` indique aux gestionnaires de l'objet enfant qu'ils doivent agir sur les propriétés de cet objet, et non sur celles d'autres objets enfants. De cette façon, lorsqu'un gestionnaire dans un objet enfant se rapporte à des propriétés, il emploie les valeurs de son propre objet enfant pour ces propriétés.

Le terme `me` doit toujours être la première variable de paramètre indiquée dans chaque définition de gestionnaire d'un script parent. Il est toujours important de définir `me` comme premier paramètre pour les scripts parents et de passer le même paramètre si vous devez appeler d'autres gestionnaires dans le même script parent. Ceux-ci seront en effet les gestionnaires de chacun des objets enfants du script.

Lorsque vous faites référence à des propriétés définies dans des scripts ancêtres, vous devez utiliser le paramètre `me` comme source de la référence. En effet, la propriété, bien qu'elle soit définie dans le script ancêtre, n'en reste pas moins une propriété de l'objet enfant. Par exemple, l'instruction suivante utilise `me` pour désigner un objet et accéder aux propriétés définies dans un de ses ancêtres :

```
-- Syntaxe Lingo
x = me.y - accéder à la propriété y de l'ancêtre
```

La variable `me` étant présente dans chaque gestionnaire de l'objet enfant, elle indique que tous les gestionnaires contrôlent ce même objet enfant.

Création d'un objet enfant

Toute l'existence des objets enfants se déroule dans la mémoire ; ils ne sont pas enregistrés avec une animation. Seuls les scripts parents et ancêtres sont enregistrés sur disque.

Pour créer un nouvel objet enfant, utilisez la méthode `new()` et affectez à cet objet enfant un nom de variable ou une position dans une liste afin de pouvoir l'identifier et l'utiliser plus tard.

Pour créer un objet enfant et l'affecter à une variable, utilisez la syntaxe suivante :

```
-- Syntaxe Lingo
NomDeVariable = new(script "NomDeScript", paramètre1, paramètre2, ...)
```

Le paramètre *NomDeScript* est le nom du script parent, et *paramètre1*, *paramètre2*, ... représentent tout paramètre que vous passez au gestionnaire `on new` de l'objet enfant. La méthode `new()` crée un objet enfant dont l'ancêtre est *NomDeScript*. Elle appelle ensuite le gestionnaire `on new` de l'objet enfant, avec les paramètres indiqués.

L'instruction `new()` peut provenir de n'importe quel endroit de l'animation. Vous pouvez personnaliser les paramètres initiaux de l'objet enfant en modifiant les valeurs des paramètres transmis avec l'instruction `new()`.

Chaque objet enfant n'utilise que la mémoire nécessaire à l'enregistrement des valeurs courantes de ses propriétés et variables ainsi qu'une référence au script parent. Par conséquent, vous pouvez généralement créer et gérer autant d'objets enfants que vous le souhaitez.

Des instructions `new()` supplémentaires permettent de produire d'autres objets enfants à partir du même script parent.

Pour créer des objets enfants sans initialiser immédiatement leurs variables de propriété, utilisez la méthode `rawNew()`. La méthode `rawNew()` effectue cette opération en créant l'objet enfant sans appeler le gestionnaire `on new` du script parent. Au cas où de grandes quantités d'objets enfants seraient requises, `rawNew()` permet de créer les objets à l'avance et de reporter l'affectation des valeurs de propriété jusqu'à ce que chaque objet soit requis.

L'instruction suivante crée un objet enfant à partir du script parent Véhicule sans initialiser ses variables de propriétés et l'attribue à la variable `voiture1` :

```
-- Syntaxe Lingo
voiture1 = script("Voiture").rawNew()
```

Pour initialiser les propriétés de l'un de ces objets enfants, appelez son gestionnaire `voiture1` :

```
voiture1.new
```

Vérification des propriétés d'un objet enfant

Vous pouvez vérifier les valeurs de variables de propriétés spécifiques dans des objets enfants à l'aide de la syntaxe *NomObjet.NomDePropriété* syntax. Par exemple, l'instruction suivante affecte à la variable `x` la valeur de la propriété `vitesseVoiture` de l'objet enfant dans la variable `voiture1` :

```
-- Syntaxe Lingo
x = voiture1.vitesseVoiture
```

La consultation des propriétés d'un objet depuis l'extérieur de cet objet peut s'avérer utile pour obtenir des informations sur des groupes d'objets, comme la vitesse moyenne de tous les véhicules d'un jeu de course automobile. Vous pouvez également utiliser les propriétés d'un objet afin de déterminer le comportement d'autres objets qui en dépendent.

En plus de vérifier les propriétés que vous affectez, vous pouvez déterminer si un objet enfant contient un gestionnaire spécifique ou rechercher le script parent d'où provient un objet. Cette fonction est très pratique si des objets proviennent de scripts parents similaires mais présentant des différences très subtiles.

Par exemple, vous pouvez créer un scénario dans lequel un script parent parmi d'autres pourrait servir à créer un objet enfant. Vous pouvez ensuite déterminer de quel script parent un objet enfant déterminé provient à l'aide de la fonction `script()`, qui renvoie le nom du script parent d'un objet.

Les instructions suivantes vérifient si l'objet `voiture1` a été créé depuis le script parent appelé `Voiture` :

```
-- Syntaxe Lingo
if voiture1.script = script("Voiture") then
  _sound.beep()
end if
```

Vous pouvez également obtenir une liste de gestionnaires d'un objet enfant en utilisant la méthode `handlers()` ou vérifier si un gestionnaire particulier existe dans un objet enfant en utilisant la méthode `handler()`.

L'instruction suivante place une liste des gestionnaires de l'objet enfant `voiture1` dans la variable `maListeDeGestionnaire` :

```
-- Syntaxe Lingo
maListeDeGestionnaire = voiture1.handlers()
```

Cette liste pourrait se présenter comme suit :

```
[#démarrer, #accélérer, #arrêter]
```

Les instructions suivantes utilisent la méthode `handler()` pour vérifier si le gestionnaire `on accélérer` existe dans l'objet enfant `voiture1` :

```
-- Syntaxe Lingo
if voiture1.handler(#accélérer) then
  put("L'objet enfant voiture1 contient le gestionnaire indiqué sur
  accélérer.")
end if
```

Suppression d'un objet enfant

Vous pouvez supprimer un objet enfant d'une animation en modifiant la valeur de toutes les variables qui contiennent une référence à cet objet enfant. Si l'objet enfant a été affecté à une liste, comme `actorList`, vous devez également supprimer l'objet en question de la liste.

Pour supprimer un objet enfant et les variables qui y font référence :

- Affectez `VOID` à chaque variable.

Director supprime l'objet enfant lorsque plus rien ne lui fait référence. Dans l'exemple suivant, `balle1` contient la seule référence à un objet enfant particulier, et la définit comme étant `VOID` pour supprimer l'objet de la mémoire.

```
-- Syntaxe Lingo
balle1 = VOID
```

Pour supprimer un objet de actorList :

- Utilisez la méthode `delete()` pour supprimer l'élément de la liste.

Utilisation de scriptInstanceList

Vous pouvez utiliser la propriété `scriptInstanceList` pour ajouter dynamiquement de nouveaux comportements à une image-objet. Normalement, `scriptInstanceList` est la liste des instances de comportements créées à partir des initialisateurs de comportements définis dans le scénario. Si vous ajoutez des objets enfants créés à partir de scripts parents à la liste, les objets enfants reçoivent les messages envoyés à d'autres comportements.

Par exemple, l'instruction suivante ajoute un objet enfant à la propriété `scriptInstanceList` de l'image-objet 10 :

```
-- Syntaxe Lingo
add(sprite(10).scriptInstanceList, new(script "rotation", 10))
```

Le script suivant est le script parent éventuel auquel l'instruction précédente fait référence :

```
-- Syntax Lingo pour script parent "rotation"
property spriteNum

on new me, aSpriteNum
  spriteNum = aSpriteNum
  return me
end

on prepareFrame me
  sprite(spriteNum).rotation = sprite(spriteNum).rotation + 1
end
```

Lorsqu'un objet enfant est ajouté à `scriptInstanceList`, vous devez initialiser la propriété `spriteNum` de l'objet enfant. Cette opération est généralement effectuée à partir d'un paramètre passé au gestionnaire `on new`.

Remarque : Le message `beginSprite` n'est pas envoyé aux objets enfants ajoutés dynamiquement.

Pour obtenir des informations concernant `scriptInstanceList`, consultez *scriptInstanceList*, page 1030.

Utilisation de actorList

Vous pouvez établir une liste spéciale d'objets enfants (ou de tout autre objet) qui reçoivent un message personnel à chaque fois que la tête de lecture entre dans une image ou que la méthode `updateStage()` met la scène à jour.

La liste spéciale est `actorList`, qui ne contient que les objets ayant été explicitement ajoutés à la liste.

Le message est le message `stepFrame` qui n'est émis que lorsque la tête de lecture entre dans une image ou que la commande `updateStage()` est utilisée.

Les objets de `actorList` reçoivent un message `stepFrame` au lieu d'un message `enterFrame` à chaque image. Si les objets disposent d'un gestionnaire `on stepFrame`, le script du gestionnaire est exécuté à chaque fois que la tête de lecture entre dans une nouvelle image ou que la méthode `updateStage()` met la scène à jour.

Parmi les applications possibles de `actorList` et `stepFrame` figurent l'animation d'objets enfants utilisés en tant qu'images-objets ou la mise à jour d'un compteur qui suit le nombre de fois que la tête de lecture entre dans une image.

Un gestionnaire `on enterFrame` pourrait produire les mêmes résultats, mais la propriété `actorList` et le gestionnaire `stepFrame` sont conçus pour des performances optimales dans Director. Les objets de `actorList` répondent mieux aux messages `stepFrame` qu'aux messages `enterFrame` ou aux messages personnalisés émis après une méthode `updateStage()`.

Pour ajouter un objet dans actorList :

- Utilisez la propriété `actorList` comme suit, *`objetEnfant`* faisant référence à l'objet enfant à ajouter :

```
-- Syntaxe Lingo
_movie.actorList.add(objetEnfant)
```

Le gestionnaire `stepFrame` de l'objet, défini dans son script parent ou ancêtre est alors exécuté automatiquement à chaque fois que la tête de lecture avance. L'objet est transmis en tant que premier paramètre `me` au gestionnaire `on stepFrame`.

Director n'efface pas le contenu de `actorList` lorsqu'il passe à une autre animation, ce qui peut provoquer un comportement imprévisible dans cette dernière. Pour éviter le transfert des objets enfants de l'animation actuelle dans la nouvelle animation, insérez une instruction qui efface `actorList` dans le gestionnaire `on prepareMovie` de la nouvelle animation.

Pour effacer les objets enfants de actorList :

- Donnez à `actorList` la valeur `[ ]`, qui représente une liste vide.

Pour obtenir des informations concernant `actorList`, veuillez consulter *actorList*, page 667.

Création d'objets de temporisation

Un objet de temporisation est un objet script qui agit comme un minuteur et qui envoie un message à la fin du délai. Ce type d'objet est utile pour les scénarios qui nécessitent l'exécution de certains événements à intervalles réguliers ou après un certain délai.

Les objets de temporisation peuvent envoyer des messages appelant des gestionnaires dans des objets enfants ou des scripts d'animation. Vous pouvez créer un objet de temporisation en utilisant le mot de passe `new()`. Vous devez spécifier le nom de l'objet, le gestionnaire à appeler et la fréquence avec laquelle le gestionnaire doit être appelé. Lorsqu'un objet de temporisation est créé, Director maintient une liste des objets de temporisation actuellement actifs, nommée `timeOutList`.

Pour créer des objets de temporisation :

- Utilisez l'une des deux syntaxes ci-dessous :

```
--Syntaxe Lingo
NomDeVariable = timeOut(NomObjTemp).new(intMillisecondes, #NomGestionnaire \
  {, ObjetCible})

NomDeVariable = new timeOut(NomObjTemp, intMillisecondes, #NomGestionnaire \
  {, ObjetCible})
```

Cette instruction utilise les éléments suivants :

- *`NomDeVariable`* est la variable dans laquelle vous placez l'objet de temporisation.
- `timeOut` indique le type d'objet Lingo que vous créez.
- *`NomObjTemp`* est le nom que vous donnez à l'objet de temporisation. Ce nom figurera dans la liste `timeOutList`. C'est la propriété `#name` de l'objet.
- `new` crée un nouvel objet.
- *`intMillisecondes`* indique la fréquence avec laquelle l'objet de temporisation doit appeler le gestionnaire indiqué. C'est la propriété `#period` de l'objet. Une valeur de 2000, par exemple, appelle le gestionnaire indiqué toutes les 2 secondes.
- *`#NomGestionnaire`* est le nom du gestionnaire que l'objet doit appeler. C'est la propriété `#GestionnaireTemporisation` de l'objet. Vous la représentez sous la forme d'un symbole en précédant le nom du signe #. Par exemple, un gestionnaire nommé `on accélérer` serait noté `#accélérer`.
- *`ObjetCible`* indique le gestionnaire de l'objet enfant à appeler. C'est la propriété `#cible` de l'objet. Ce paramètre permet d'être spécifique si plusieurs objets enfants contiennent les mêmes gestionnaires. Si vous omettez ce paramètre, Director recherche le gestionnaire indiqué dans le script d'animation.

L'instruction suivante crée un objet de temporisation appelé `minuteur1` qui appellera un gestionnaire `on accélérer` dans l'objet enfant `voiture1` toutes les 2 secondes :

```
-- Syntaxe Lingo
monMinuteur = timeOut("minuteur1").new(2000, #accélérer, voiture1)
```

Pour déterminer le moment auquel le message de temporisation suivant est envoyé par un objet de temporisation déterminé, consultez sa propriété #time. La valeur renvoyée est le moment, en millièmes de secondes, auquel le message de temporisation suivant sera envoyé. Par exemple, l'instruction suivante détermine le moment auquel le prochain message de temporisation sera envoyé de l'objet de temporisation minuteur1 et l'affiche dans la fenêtre Message :

```
-- Syntaxe Lingo
put timeout("minuteur1").time)
```

Utilisation de timeOutList

Lorsque vous commencez à créer des objets de temporisation, vous pouvez utiliser timeOutList pour déterminer le nombre d'objets de temporisation actifs à un moment particulier.

L'instruction suivante attribue la variable x au nombre d'objets se trouvant dans timeOutList en utilisant la propriété count.

```
-- Syntaxe Lingo
x = _movie.timeoutList.count
```

Vous pouvez également faire référence à un objet de temporisation en employant son numéro dans la liste.

L'instruction suivante supprime le second objet de temporisation dans timeOutList en utilisant la méthode forget().

```
-- Syntaxe Lingo
timeout(2).forget()
```

Relais d'événements système au moyen d'objets de temporisation

Lorsque vous créez des objets de temporisation qui font référence à des objets enfants précis, vous permettez à ces derniers de recevoir des événements système. Les objets de temporisation relayent ces événements vers leurs objets enfants cibles. Les événements système qui peuvent être reçus par des objets enfants sont, par exemple, prepareMovie, startMovie, stopMovie, prepareFrame et exitFrame. En incluant des gestionnaires pour ces événements dans les objets enfants, vous pouvez ordonner aux objets enfants de leur répondre en fonction de vos besoins. Les événements système reçus par les objets enfants sont également reçus par les scripts d'animation, les scripts d'image et les autres scripts définis comme devant leur répondre.

Le script parent suivant contient un gestionnaire pour l'événement système exitFrame ainsi qu'un gestionnaire personnalisé slowDown.

```
-- Syntaxe Lingo
propriété vitesse

on new me
  vitesse = random(55)
end

on exitFrame
  vitesse = vitesse + 5
end

on slowDown mph
  vitesse = vitesse - mph
end
```

Association de propriétés personnalisées avec des objets de temporisation

Si vous souhaitez associer des propriétés personnalisées avec un objet de temporisation, vous devez créer un objet de temporisation dont la cible n'est pas une référence à un objet instance de script. Lorsque vous utilisez cette technique, les données cibles deviennent des données associées à l'objet de temporisation pouvant être utilisées dans votre gestionnaire de temporisation.

L'exemple suivant vous montre comment utiliser cette technique.

```
-- Syntaxe Lingo
-- initialisation d'un objet de temporisation et transmission dans une liste de
  propriété de données (tDonnées)
-- au lieu de faire référence à un objet d'instance de script.
tDonnées = [#beta: 0]
tTO = timeout("betaData").new(50,#gestionnaireCible,tDonnées)

-- au sein d'un script d'animation, créer le gestionnaire gestionnaireCible
on gestionnaireCible (aData)
  -- incrémentation et affichage de la propriété bêta
  tDonnées.beta = tDonnées.beta + 1
  put(tDonnées.beta)
end gestionnaireCible
```

Dans l'exemple précédent, la propriété `beta` continue d'être incrémentée. Cela signifie que vous pouvez initialiser plusieurs objets de temporisation qui appellent le même gestionnaire de script d'animation. Chaque objet de temporisation peut posséder sa propre liste de données.

En général, sachez que :

- Lors de l'utilisation d'instances de script en tant que cible, le gestionnaire cible de cette instance de script en particulier est appelé. Vous ne pouvez pas utiliser des propriétés personnalisées avec cette technique.
- Lors de l'utilisation d'une référence autre qu'une instance de script (une liste de propriété par exemple) en tant que cible, le gestionnaire cible dans un script d'animation est appelé. Vous pouvez utiliser des propriétés personnalisées avec cette technique.

Programmation orientée objet avec la syntaxe JavaScript

La programmation orientée objet dans la syntaxe JavaScript est quelque peu différente de celle d'autres langages orientés objet tels que Java et C++. Alors que certains langages orientés objet sont basés sur les classes, la syntaxe JavaScript est basée sur les prototypes.

Les deux sections qui suivent comparent à un niveau élevé les langages basés sur les classes aux langages basés sur les prototypes tels que la syntaxe JavaScript.

- Dans les langages basés sur les scripts, vous créez des définitions de classes se rapportant aux propriétés et méthodes initiales qui caractérisent toutes les instances créées dans ces classes. Une définition de classe contient des méthodes spéciales appelées méthodes de construction et utilisées pour créer des instances de cette classe. Une instance créée à l'aide de l'opérateur `new` en association avec une méthode de construction particulière hérite de toutes les propriétés de sa classe parent. Cette instance peut également effectuer toute autre tâche de traitement qui lui est propre en fonction du constructeur qui a été appelé.

  Dans une définition de classe, vous procédez à l'opération d'héritage en créant une sous-classe qui hérite de toutes les propriétés de sa classe parent, en plus de définir les nouvelles propriétés et de modifier les propriétés héritées. La classe parent à partir de laquelle une sous-classe a été créée est aussi appelée super classe.

- Dans les langages basés sur les prototypes tels que la syntaxe JavaScript, il n'existe pas de distinction entre les classes, les instances, les sous-classes, etc. Tous ces éléments sont considérés comme des objets. Dans la syntaxe JavaScript, au lieu d'utiliser des définitions de classes, vous utilisez des « objets prototype » en tant que modèle à partir duquel de nouveaux objets sont créés. Tout comme dans les langages basés sur les classes, dans la syntaxe JavaScript, vous créez un nouvel objet en utilisant l'opérateur `new` en association avec une fonction de construction.

  Au lieu d'utiliser les super classes et les sous-classes, vous associez les objets prototype à des fonctions de construction pour effectuer l'héritage dans la syntaxe JavaScript. Ce procédé est très similaire à celui qui consiste à utiliser les super classes et les sous-classes, à l'exception de la terminologie qui est différente.

  Contrairement aux langages basés sur les classes, la syntaxe JavaScript vous permet aussi d'ajouter et de supprimer des propriétés à un objet ou une série d'objets à l'exécution. Par exemple, si vous ajoutez une propriété à un objet prototype à l'exécution, les objets apparentés héritent également de cette propriété.

Terminologie orientée objet

Dans la syntaxe JavaScript, étant donné que tous les types correspondent à des objets, les termes basés sur les classes tels que *superclass (super-classe)*, *subclass (sous-classe)*, *class (classe)*, *instance*, etc. n'ont aucune signification technique littérale. Cependant, ces termes correspondent essentiellement à des objets dans la syntaxe JavaScript et peuvent être utilisés pour illustrer les différents concepts d'objets de syntaxe JavaScript. Ainsi, ces termes basés sur les classes sont synonymes d'*object* en ce qui concerne la programmation orientée objet dans la syntaxe JavaScript et signifient ceci :

- *superclass (super-classe)* ; Toute classe à partir de laquelle les sous-classes (objets) sont créées ; une classe parent.
- *subclass (sous-classe)* ; Toute classe ayant été créée à partir d'une super classe (objet) ; une classe enfant.
- *class (classe)* ; Terme générique signifiant super classe ou sous-classe ; une classe parent ou enfant.
- *instance* ou *object instance (object d'instance)* ; Objet unique ayant été créé à partir d'une super classe.

Classes personnalisées

L'un des principaux avantages de la programmation orientée objet est la possibilité de créer des classes personnalisées donnant accès à des fonctionnalités personnalisées à vos scripts. Les classes prédéfinies, telles que Objet, String et Math, de la syntaxe JavaScript sont utiles, mais les fonctionnalités qu'elles offrent peuvent ne pas s'appliquer à la tâche que vous souhaitez accomplir. Supposons, par exemple, que vous souhaitez que certains objets de votre animation représentent des types de transports, tels que des voitures, des bateaux, des avions, et que vous souhaitez que chaque catégorie affiche des caractéristiques et fonctionnalités uniques. Ni les classes de syntaxe JavaScript prédéfinies, ni les objets Director prédéfinis ne pourront directement vous fournir la fonctionnalité dont vous avez besoin. C'est pourquoi vous devez créer une nouvelle classe pour chaque type de transport afin de pouvoir définir des caractéristiques uniques pour chaque type.

Retenez que lorsque vous créez des classes personnalisées de la syntaxe JavaScript, vous avez toujours accès à toutes les fonctions et fonctionnalités des objets Director prédéfinis. Cela signifie que même si les objets Director prédéfinis ne vous fournissent pas directement les fonctionnalités dont vous avez besoin, vous pouvez toujours les utiliser dans vos classes personnalisées pour accéder à leurs valeurs et à leurs fonctionnalités prédéfinies.

Fonctions de construction

Dans la syntaxe JavaScript, une fonction de construction représente la classe qui contient le modèle à partir duquel les nouvelles instances d'objets sont créées. Les fonctions de construction créent et initialisent (définissent l'état par défaut) les propriétés des nouveaux objets.

Le format des fonctions de construction est fondamentalement identique à celui des fonctions de méthode standard de syntaxe JavaScript. Cependant, la fonction de construction utilise un mot de passe spécial `this` pour représenter une référence à un nouvel objet initialisé. En règle générale, une fonction de méthode effectue des actions sur un ensemble de données d'un objet.

L'exemple suivant vous montre une manière de créer une fonction de construction Rectangle qui pourrait être utilisée pour initialiser la hauteur et la largeur de nouveaux objets Rectangle.

```
function Rectangle(l, h) {
  this.width = l;
  this.height = h;
}
```

Vous pouvez également créer une fonction de construction en utilisant la syntaxe « fonction littérale ». La syntaxe fonction littérale fournit les mêmes fonctionnalités que la syntaxe précédente et correspond à une autre manière de rédiger une construction. L'exemple suivant vous montre comment utiliser la syntaxe fonction littérale pour créer une fonction de construction Rectangle identique à la précédente.

```
Rectangle = function(l, h) {
  this.width = l;
  this.height = h;
}
```

Remarque : Lorsque vous définissez des fonctions de construction à appliquer à une animation, assurez-vous de les placer dans un script d'animation afin qu'elles soient disponibles tout le temps.

Il est recommandé de donner des noms qui correspondent aux fonctionnalités des fonctions de construction et d'utiliser leur initiale dans le nom, tel que `Rectangle` ou `Cercle`.

En général, les fonctions de construction sont utilisées pour initialiser de nouveaux objets, mais peuvent également renvoyer le nom de l'objet le cas échéant. Si vous retournez l'objet initialisé, l'objet retourné devient la valeur de l'expression `new`.

Instances d'objet

La manière la plus commune de créer une nouvelle instance d'objet est d'utiliser l'opérateur `new` suivi du nom de la fonction de construction. Les exemples suivants créent de nouvelles instances d'objet.

```
var objAléatoire = new Object(); // affecte une référence à un objet Objet
var objChaîne = new String(); // affecte une référence à un objet Chaîne
```

Une fonction de construction peut parfois définir des paramètres qu'une nouvelle instance d'objet passe afin d'initialiser l'état de l'instance d'objet. Si une fonction de construction définit les paramètres utilisés lors de l'initialisation de nouvelles instances d'objet, les valeurs de propriétés sont initialisées ainsi :

- Si vous passez les valeurs à la fonction de construction lors de l'initialisation, les propriétés qui ont reçu les valeurs sont définies selon ces valeurs.
- Si vous ne passez pas ces valeurs à la fonction de construction lors de l'initialisation, les propriétés qui n'ont pas reçu les valeurs sont définies sur `undefined`.

Lorsque vous créez de nouvelles instances d'objet, le mot de passe `this` est utilisé dans le corps de la fonction de construction associée afin de faire référence à une nouvelle instance d'objet. Par conséquent, la nouvelle instance d'objet est initialisée avec toutes les propriétés définies à l'aide de la syntaxe `this.`*`NomDePropriété`*.

Dans l'exemple suivant, une fonction de construction `Cercle` utilise le mot de passe `this` pour spécifier les noms des trois propriétés qui seront associées aux nouvelles instances d'objet. L'instruction suivant la construction initialise une nouvelle instance d'objet en passant les valeurs à la construction. Ces valeurs sont utilisées en tant que valeurs initiales des propriétés spécifiées par le mot de passe `this`.

```
// Fonction de construction Cercle
function Cercle(x, y, r) {
  this.xCoord = x;
  this.yCoord = y;
  this.radius = r;
}

// xCoord = 10, yCoord = 15, radius = 5
var objCercle = new Cercle(10, 15, 5);
```

Une fois `objCercle` initialisé, vous pouvez accéder à ses propriétés. A l'aide de l'instance `objCercle` créée précédemment, vous pouvez définir certaines variables sur les valeurs de ses propriétés.

```
var laCoordX = objCercle.xCoord; // affecte la valeur 10 à laCoordX
var laCoordY = objCercle.yCoord; // affecte la valeur 15 à laCoordY
var leRayon = objCercle.radius; // affecte la valeur 5 à leRayon
```

Remarque : Pour plus d'informations concernant l'utilisation de la syntaxe à points pour accéder aux propriétés et méthodes d'un objet, veuillez consulter *Format de scripting à syntaxe à points*, page 56.

Il est recommandé de donner aux nouveaux objets des noms qui correspondent à leur fonctionnalité et d'utiliser des minuscules, tel que `objRectangle` ou `objCercle`.

Vous pouvez également créer une instance d'objet en utilisant la syntaxe « objet littéral ». Ainsi, vous n'avez pas besoin d'utiliser l'opérateur `new` et de fonction de construction. Utilisez cette technique uniquement lorsque vous n'avez besoin que d'une instance d'un objet n'ayant pas été défini dans une fonction de construction. Dans l'exemple suivant, une instance d'objet est créée avec `x = 1`, `y = 2` et `radius = 2`.

```
var ObjPetitCercle = { x:1, y:2, radius:2 };
```

Héritage d'objets

Non seulement la programmation orientée objet est capable de créer vos propres classes personnalisées, mais grâce à elle, les sous-classes peuvent également hériter des propriétés et méthodes des super classes à partir desquelles elles ont été créées. L'héritage vous facilite la création d'objets aux propriétés et fonctionnalités intégrées.

Dans la syntaxe JavaScript, une super classe correspond à la classe de base à partir de laquelle toutes les autres sous-classes sont créées : la super classe Objet. La super classe Objet comprend quelques propriétés et méthodes de base. Les sous-classes qui sont créées à l'aide du modèle Objet héritent toujours de ces propriétés et méthodes de base et définissent leurs propres propriétés et méthodes. Les sous-classes provenant de ces classes héritent de l'objet Objet, de leurs super classes et ainsi de suite. Tous les autres objets que vous créez prolongent cette chaîne d'héritage.

Par exemple, l'objet Objet contient la propriété `constructor` et la méthode `toString()`. Si vous créez une nouvelle classe nommée `SousObj1`, elle correspond à une sous-classe de l'objet Objet et hérite ainsi de la propriété `constructor` et de la méthode `toString()` de l'objet Objet. Ensuite, si vous créez une autre classe nommée `SousObj2` en utilisant `SousObj1` en tant que super classe, `SousObj2` héritera également de la propriété `constructor` et de la méthode `toString()` de l'objet Objet, en plus de toutes les autres propriétés et méthodes personnalisées que vous avez définies dans `SousObj1`.

Deux des propriétés importantes dont vos fonctions de constructions personnalisées héritent de la super classe Objet sont `prototype` et `constructor`. La propriété `prototype` représente l'objet prototype de la classe, qui vous permet d'ajouter des variables (propriétés) et des méthodes aux instances d'objet. C'est le moyen par lequel l'héritage est en général implémenté dans la syntaxe JavaScript. La propriété `constructor` représente la fonction de construction en elle-même. Dans les sections suivantes, on vous explique l'utilisation de ces propriétés.

Objets prototype

Lorsque vous créez une sous-classe, cette dernière hérite automatiquement des propriétés et méthodes de la super classe à laquelle elle se rapporte. Dans la syntaxe JavaScript, l'héritage est en général implémenté par les objets prototype. En réalité, une sous-classe hérite des propriétés et méthodes de l'objet prototype de sa super classe et non de la super classe en elle-même. Ce point important apporte une avantage sérieux : Toutes les propriétés et méthodes ne doivent pas être obligatoirement copiées d'une classe à une instance d'objet de cette classe, ce qui peut considérablement réduire la quantité de mémoire nécessaire à la création de nouvelles instances d'objet.

Dans la syntaxe JavaScript, chaque classe, y compris la classe prédéfinie Objet, contient uniquement un objet prototype. Chaque instance d'objet créée à partir d'une classe a accès aux propriétés et méthodes dans l'objet prototype de cette classe. Par conséquent, l'objet prototype d'une classe est en général le seul objet qui stocke les propriétés et méthodes pour cette classe ; une instance d'objet contient uniquement les propriétés nécessaires à l'initialisation de cette instance.

Dans votre code, il semble que chaque instance d'objet comprend réellement ces propriétés et méthodes parce que vous y avez accès directement à partir de l'instance d'objet. Mais en réalité, l'instance utilise l'objet prototype pour y accéder. L'objet prototype d'une classe est automatiquement créé lorsque vous créez la classe. Vous pouvez accéder à l'objet prototype à l'aide de la propriété `prototype` de la classe.

Etant donné que l'objet prototype d'une classe stocke les propriétés partagées par toutes les instances d'objet, il peut définir correctement les propriétés et méthodes donc les valeurs seront partagées parmi toutes les instances d'objet. Le fait de partager les propriétés et les méthodes parmi les instances d'objet vous facilite la création d'instances qui affichent un comportement défini par défaut et la personnalisation de toutes les instances associées au comportement par défaut.

En revanche, les objets prototype ne pourront pas définir correctement les propriétés et méthodes dont les valeurs peuvent varier selon les instances d'objet. Dans ces cas-là, vous pouvez définir ces propriétés et méthodes au sein de la classe elle-même.

Pour spécifier l'étendue d'une propriété ou méthode personnalisée, vous pouvez la définir selon l'un des quatre types suivants :

- Variables d'instances
- Méthodes d'instances
- Variables de classes
- Méthodes de classe

Variables d'instances

Les variables d'instances correspondent à toutes les variables (propriétés) définies dans une fonction de construction et copiées dans chaque instance d'objet de cette construction. Toutes les instances d'objet possèdent leurs propres copies de variables d'instances. Cela signifie que s'il existe cinq instances d'objet pour une classe `Cercle`, cinq copies de chaque variable d'instance sont définies dans la classe. Etant donné que chaque instance d'objet possède sa propre copie de variable d'instance, chaque instance d'objet peut affecter une valeur unique à une variable d'instance sans pour autant modifier les valeurs des autres copies de la variable d'instance. Vous pouvez accéder aux variables d'instance directement à partir des instances d'objet qui leur sont propres.

Dans l'exemple suivant, quatre variables d'instance –`marque`, `modèle`, `couleur` et `vitesse`– sont définies dans une fonction de construction. Ces quatre variables d'instances sont directement disponibles à partir de toutes les instances d'objet de la construction `Voiture`.

```
function Voiture(marque, modèle, couleur) { // définit une classe Voiture
  this.marque = marque;
  this.modèle = modèle;
  this.couleur = couleur;
  this.vitesse = 0;
}
```

L'instance d'objet suivante `objVoiture` contient les quatre variables d'instances. Bien que la valeur de la variable d'instance `vitesse` ne soit pas passée à la construction `Voiture`, `objVoiture` possède toujours une propriété `vitesse` dont la valeur initiale est 0 parce que la variable `vitesse` est définie dans la construction `Voiture`.

```
// objVoiture.marque="Subaru", objVoiture.modèle="Forester",
// objVoiture.couleur="gris", objVoiture.vitesse = 0
var objVoiture = new Voiture("Subaru", "Forester", "gris");
```

Méthodes d'instances

Les méthodes d'instances correspondent à toute méthode accessible via une instance d'objet. Les instances d'objet ne possèdent pas leurs propres copies de méthodes d'instances. En revanche, les méthodes d'instances sont les premières définies en tant que fonctions, et ensuite les propriétés de l'objet prototype de la fonction de construction sont définies sur les valeurs de la fonction. Les méthodes d'instances utilisent le mot de passe `this` dans le corps de la fonction de construction définie afin de faire référence à l'instance d'objet sur laquelle elles opèrent. Bien qu'une instance d'objet donné ne possède pas une copie d'une méthode d'instance, vous pouvez toujours accéder aux méthodes d'instances directement à partir des instances d'objet qui y sont associées.

Dans l'exemple suivant, une fonction nommée `Voiture_augmenterVitesse()` est définie. Le nom de la fonction est ensuite affecté à la propriété `augmenterVitesse` de l'objet prototype de la classe `Voiture`.

```
// augmente la vitesse de la voiture
function Voiture_augmenterVitesse(x) {
  this.vitesse += x;
  return this.vitesse;
}
Voiture.prototype.augmenterVitesse = Voiture_augmenterVitesse;
```

Une instance d'objet de `Voiture` peut ensuite accéder à la méthode `augmenterVitesse()` et affecter sa valeur à la variable à l'aide de la syntaxe suivante.

```
var objVoiture = new Voiture("Subaru", "Forester", "gris");
var nouvelleVitesse = objVoiture.augmenterVitesse(30);
```

Vous pouvez également créer une méthode d'instance à l'aide de la syntaxe de fonction littérale. En utilisant la syntaxe de fonction littérale, il n'est plus nécessaire ni de définir une fonction, ni d'affecter un nom de propriété au nom de fonction.

Dans l'exemple suivant, la syntaxe de fonction littérale est utilisée pour définir une méthode `augmenterVitesse()` qui contient les mêmes fonctionnalités que la fonction `augmenterVitesse()` définie précédemment.

```
// augmente la vitesse d'une voiture
Voiture.prototype.augmenterVitesse = function(x) {
  this.vitesse += x;
  return this.vitesse;
}
```

Variables de classes

Egalement appelées variables *static*, elles correspondent à toutes les variables (propriétés) associées à une classe et non à une instance d'objet. Il n'existe qu'une copie de variable de classe, quel que soit le nombre d'instances d'objet créées dans cette classe. Les variables de classe n'utilisent pas l'objet prototype pour implémenter l'héritage. Vous pouvez accéder à une variable de classe directement via la classe et non via une instance d'objet ; vous devez définir une classe dans une fonction de construction avant de pouvoir définir les variables de classe.

Dans l'exemple suivant, deux variables de classe sont définies–`VITESSE_MAX` et `VITESSE_MIN`.

```
function Voiture() { // définit une classe de voiture
  ...
}

Voiture.VITESSE_ = 165;
Voiture.VITESSE_MIN = 45;
```

Vous avez accès aux variables de classe `MAX_SPEED` et `MIN_SPEED` directement à partir de la classe `Voiture`.

```
var VitesseMaxVoiture = Voiture.VITESSE_MAX; // VitesseMaxVoiture = 165
var VitesseMinVoiture = Voiture.VITESSE_MIN; // VitesseMinVoiture = 45
```

Méthodes de classe

Egalement appelées méthodes *static*, elles correspondent à toutes les méthodes associées à une classe et non à une instance d'objet. Certaines des méthodes ou propriétés suivantes s'appliquent uniquement aux images-objets ayant été créées à partir d'un acteur RealMedia. Les méthodes de classe n'utilisent pas l'objet prototype pour implémenter l'héritage. Vous avez accès à une méthode directement via la classe et non via une instance d'objet ; vous devez définir une classe dans une fonction de construction avant de pouvoir définir les méthodes de classe.

Dans l'exemple suivant, on définit une fonction nommée `définirVitesseInitiale()` qui peut modifier la vitesse par défaut des nouvelles instances de voiture. Le nom de la fonction est affecté à la propriété `définirVitesseInitiale` de la classe `Voiture`.

```
function Voiture(marque, modèle, couleur) { // définit une classe de voiture
  this.marque = marque;
  this.modèle = modèle;
  this.couleur = coleur;
  this.vitesse = Voiture.vitesseParDéfaut;
}
Voiture.vitesseParDéfaut = 10; // vitesse initiale des nouvelles instances de
  voiture
// augmente la vitesse d'une voiture
function Voiture_définirVitesseInitiale(x) {
  Voiture.VitesseParDéfaut = x;
}
Voiture.définirVitesseInitiale = Voiture_définirVitesseInitiale;
```

Vous pouvez accéder à la méthode de classe `définirVitesseInitiale()` directement à partir de la classe `Voiture`.

```
var nouvelleVitesse = Voiture.définirVitesseInitiale(30);
```

Vous pouvez également créer une méthode de classe à l'aide de la syntaxe de fonction littérale. Dans l'exemple suivant, la syntaxe de fonction littérale est utilisée pour définir une méthode `définirVitesseInitiale()` contenant les mêmes fonctionnalités que la fonction `définirVitesseInitiale()` définie précédemment.

```
// augmente la vitesse d'une voiture
Voiture.définirVitesseInitiale = function(x) {
  Voiture.VitesseParDéfaut = x;
}
```

Etapes recommandées de définition d'une classe

La liste suivante décrit les étapes que nous vous recommandons de suivre lorsque vous définissez une classe.

1 Définissez une fonction de construction utilisée en tant que modèle à partir duquel les instances d'objets sont initialisées. Vous pouvez également définir toute variable d'instance de la fonction de construction en utilisant le mot de passe `this` pour faire référence à une instance d'objet.

2 Définissez toute méthode d'instance, ainsi que toute variable d'instance, stockées dans l'objet prototype d'une classe. Ces méthodes et variables d'instances sont disponibles pour toutes les instances d'objet et vous pouvez y accéder via l'objet prototype de la classe.

3 Définissez toute méthode de classe, variable de classe et constante stockées dans la classe elle-même. Vous pouvez accéder à ces méthodes et variables de classe via la classe elle-même.

Dans votre code, lorsque vous accédez à une propriété d'une instance d'objet, la syntaxe JavaScript recherche l'instance d'objet pour cette propriété. Si l'instance ne contient pas de propriété, la syntaxe JavaScript recherche alors l'objet prototype de la super classe à partir de laquelle l'instance a été créée. Etant donné qu'une instance d'objet est recherchée avant l'objet prototype de la classe à partir de laquelle il a été créé, l'instance d'objet masque les propriétés à l'objet prototype de leurs super classes. Cela signifie qu'une instance d'objet et sa super classe peuvent réellement définir une propriété si elles portent le même nom mais avec des valeurs différentes.

Suppression de variables

Vous pouvez supprimer une variable de classe ou une variable d'instance à l'aide de l'opérateur `delete`. L'exemple suivant illustre ce processus.

```
function Voiture() { // définit la fonction de construction d'une voiture
  ...
}
Voiture.couleur = "bleu"; // définit la propriété couleur de la classe de
  voiture
Voiture.prototype.moteur = "V8"; // définit une propriété de moteur du
  prototype

var objVoiture = new Voiture();

trace(Voiture.couleur); // affiche « bleu »
trace(objVoiture.moteur); // affiche « V8 »

delete Voiture.couleur;
delete Voiture.prototype.moteur;

trace(Voiture.couleur); // affichage non défini
trace(objVoiture.moteur); // affichage non défini
```

Pour accéder à la propriété de construction d'un objet prototype

Lorsque vous définissez une classe en créant une fonction de construction, la syntaxe JavaScript crée un objet prototype pour cette classe. Lorsque l'objet prototype est créé, il comprend au départ une propriété `constructor` qui se rapporte à la fonction de construction elle-même. Vous pouvez utiliser la propriété `constructor` d'un objet prototype pour déterminer le type de tout objet.

Dans l'exemple suivant, la propriété `constructor` contient une référence à la fonction de construction utilisée pour créer l'instance d'objet. La valeur de la propriété `constructor` est en réalité une référence à la construction elle-même et n'est pas une chaîne qui contient le nom de la construction.

```
function Voiture() { // définit la classe d'une voiture
  // code d'initialisation ici
}

var maVoiture = new Voiture(); // maVoiture.constructor == function Voiture()
  {}
```

Création dynamique de propriétés

Un des autres avantages dont vous pouvez bénéficier en utilisant un objet prototype pour implémenter l'héritage est que les propriétés et méthodes ajoutées à un objet prototype sont automatiquement disponibles pour les instances d'objets. Cela est vrai même si une instance d'objet a été créée avant que les propriétés ou les méthodes ne soient ajoutées.

Dans l'exemple suivant, la propriété `couleur` est ajoutée à l'objet prototype d'une classe `Voiture` après qu'une instance d'objet de `Voiture` ait déjà été créée.

```
function Voiture(marque, modèle) { // définit la classe d'une voiture
  this.marque = marque;
  this.modèle = modèle;
}

var maVoiture = new Voiture("Subaru", "Forester"); // crée une instance d'objet

trace(maVoiture.couleur); // renvoie undefined

// ajoute la propriété de couleur de la classe voiture après que maVoiture ait
  été initialisé
Voiture.prototype.couleur = "blue";

trace(maVoiture.couleur); // renvoie « bleu »
```

Vous pouvez également ajouter des propriétés aux instances d'objet après que les instances ont été créées. Lorsque vous ajoutez une propriété à une instance d'objet spécifique, cette propriété est disponible uniquement pour cette instance d'objet spécifique. A l'aide de l'instance d'objet `maVoiture` précédemment créée, les instructions suivantes ajoutent la propriété `couleur` à `maVoiture` après sa création.

```
trace(maVoiture.couleur); // renvoie undefined

maVoiture.couleur = "bleu"; // ajoute la propriété de couleur à l'instance
  maVoiture

trace(maVoiture.couleur); // renvoie « bleu »
```

```
var deuxièmeVoiture = new Voiture("Honda", "Accord"); // crée une deuxième
  instance d'objet

trace(deuxièmeVoiture.couleur); // renvoie undefined
```

Rédaction de scripts dans la fenêtre Script

Lorsque vous rédigez des scripts pour une animation, ces scripts peuvent considérablement augmenter en quantité et en complexité. La décision des méthodes et propriétés à utiliser, de la structuration efficace des scripts et de leur position judicieuse exige un plan d'action rigoureux et de nombreux tests au fur et à mesure que la complexité de l'animation augmente.

Avant de commencer à rédiger des scripts, précisez l'objectif que vous voulez atteindre. Ceci est en fait aussi important et en général aussi laborieux que la création des storyboards de l'animation.

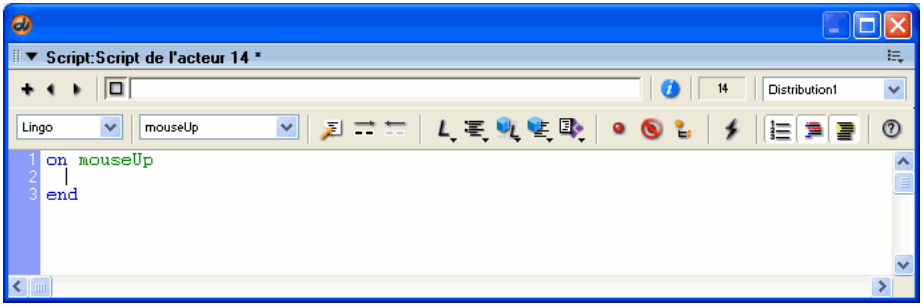
Une fois le plan d'ensemble de votre animation conçu, vous pouvez commencer à rédiger puis à tester les scripts. Attendez-vous à y passer du temps. Il est très rare qu'un script produise le résultat attendu dès la première rédaction, les premiers tests ou les premières opérations de débogage.

Il est donc conseillé de commencer de manière progressive et de tester vos scripts fréquemment. Dès qu'une partie d'un script produit l'effet escompté, rédigez la partie suivante, et ainsi de suite. Cette méthode vous permet d'identifier rapidement les bogues et garantit la précision de vos scripts au fur et à mesure que leur complexité augmente.

Lorsque vous rédigez vos scripts, vous le faites dans la fenêtre Script de l'interface utilisateur de Director. La fenêtre Script propose un certain nombre de fonctionnalités qui vous permettent de créer et modifier vos scripts.

Pour ouvrir la fenêtre Script, effectuez l'une des opérations suivantes :

- Choisissez Fenêtre > Script.
- Double-cliquez sur l'acteur script dans la fenêtre Distribution.



Vous trouverez d'autres façons de créer et d'ouvrir des scripts dans la section *Exécution d'opérations élémentaires*, page 88.

Définition des préférences de la fenêtre Script

Vous pouvez modifier la police du texte de la fenêtre Script et les couleurs des différents éléments du code. Pour modifier la police par défaut du texte de la fenêtre Script et les couleurs des différents éléments du code, vous utilisez les préférences de la fenêtre Script. Director affecte automatiquement une couleur distincte aux différents éléments du code, sauf si vous désactivez Mise en couleur automatique.

Pour définir des préférences dans la fenêtre Script :

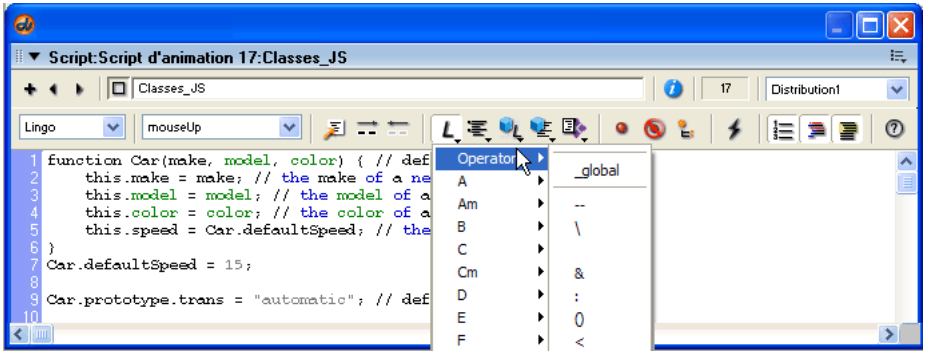
1. Choisissez Edition > Préférences > Script.
2. Pour choisir la police par défaut, cliquez sur le bouton Police et sélectionnez les attributs de la police dans la boîte de dialogue Police.
3. Pour choisir la couleur par défaut du texte affiché dans la fenêtre Script, choisissez une couleur dans la puce Couleur.
4. Pour choisir la couleur d'arrière-plan de la fenêtre Script, choisissez une couleur dans la puce Arrière-plan.
5. Pour que la fenêtre Script colorie automatiquement certains éléments du code, sélectionnez Mise en couleur automatique. Cette option est activée par défaut. Lorsque l'option Mise en couleur automatique est désactivée, le texte a la couleur par défaut.
6. Pour que la nouvelle fenêtre Script formate automatiquement vos scripts avec la mise en retrait, sélectionnez l'option Format automatique. Cette option est activée par défaut.

   Remarque : Les fonctions de coloration automatique et de formatage automatique ne s'appliquent pas aux codes JavaScript. Par conséquent, si vous créez des scripts à l'aide de la syntaxe JavaScript, les boutons Mise en couleur automatique et Formatage automatique sont désactivés dans la fenêtre Script et les termes tels que `function`, `var` et `this` ont la couleur par défaut.

7. Pour que la nouvelle fenêtre Script affiche les numéros de ligne associés à vos scripts, sélectionnez l'option Numérotation des lignes. Cette option est activée par défaut.
8. Si l'option Mise en couleur automatique est activée, choisissez les couleurs des éléments de code suivants dans les menus couleurs correspondants :
   - Mots-clés
   - Commentaires
   - Constantes
   - Personnalisé (termes que vous définissez dans votre propre code)
9. Pour changer la couleur de fond de la colonne des numéros de ligne, cliquez sur Numérotation des lignes et choisissez une nouvelle couleur.
10. Pour changer l'emplacement des volets Pile d'appels, Variables et Surveillance dans la fenêtre Débogueur, sélectionnez Gauche, Droit, Haut ou Bas dans le menu Panneaux de débogage.

Insertion de termes de scripting communs

La fenêtre Script offre des menus locaux contenant les termes de scripting communs que vous pouvez utiliser pour insérer des instructions dans un script. Les mêmes menus sont également disponibles dans la fenêtre Messages.



Dans les fenêtres Script et Messages, vous pouvez sélectionner la syntaxe de scripting à utiliser pour un script donné.

Pour sélectionner la syntaxe de scripting :

- Dans le menu local Syntaxe de script, sélectionnez Lingo ou JavaScript.

Après avoir sélectionné une syntaxe de scripting, entrez le code dans la syntaxe que vous avez choisie. Si vous essayez de compiler un script dans une syntaxe autre que celle que vous avez choisie, vous obtenez une erreur de script.

Lorsque vous entrez des scripts dans la fenêtre Script, vous pouvez insérer ou supprimer des marques de commentaire sur une seule ou plusieurs lignes de code à l'aide des boutons Insérer une marque de commentaire et Supprimer la marque de commentaire. Selon la syntaxe choisie, les boutons Insérer une marque de commentaires et Supprimer la marque de commentaire affichent les bons repères de commentaire pour cette syntaxe ; Lingo utilisant des tirets doubles (--), et la syntaxe JavaScript deux barres obliques (//).

Pour insérer un commentaire dans le code :

- Mettez en surbrillance la ou les lignes auxquelles vous voulez ajouter un commentaire et cliquez sur Insérer une marque de commentaire.

Remarque : Lorsque vous utilisez le bouton Insérer une marque de commentaire pour ajouter des commentaires sur plusieurs lignes de code en JavaScript, Director place deux barres obliques avant chaque ligne. Vous pouvez également insérer des commentaires sur plusieurs lignes de code en tapant le signe /* avant la première ligne de code et le signe */ après la dernière ligne ; mais vous devez le faire manuellement.

Pour supprimer le commentaire d'un code :

- Mettez en surbrillance la ou les lignes dont vous voulez supprimer les commentaires et cliquez sur Supprimer la marque de commentaire.

Les fenêtres Script et Messages contiennent toutes deux les menus suivants :

- Le menu alphabétique Lingo est une liste alphabétique de tous les éléments, à l'exception du Lingo 3D.

- Le menu par catégorie est une liste des éléments Lingo répertoriés selon leurs fonctions. Il n'inclut pas les éléments Lingo 3D.
- Le menu alphabétique 3D de Lingo est une liste alphabétique de tous les éléments Lingo 3D.
- Le menu par catégorie 3D de Lingo est une liste de tous les éléments Lingo 3D répertoriés selon leurs fonctions.
- Le menu local des Xtras de programmation incluent les méthodes et propriétés de tous les Xtras de programmation trouvés, qu'il s'agisse d'Xtras Macromedia ou autres.

Remarque : Les Xtras de programmation figurant dans le menu local sont limités à ceux qui supportent la méthode `interface()` et dont les noms apparaissent dans le menu local. Bien que certains types de médias d'acteurs tels 3D et DVD supportent également la méthode `interface()`, ils ne figurent pas dans le menu local Xtras de programmation parce qu'ils ne sont pas implémentés en tant que Xtras de programmation dans Director.

L'élément sélectionné dans les menus locaux est inséré par Director à l'emplacement du curseur dans la fenêtre Script.

Si un élément nécessite des paramètres supplémentaires, des repères de noms indiquant les informations supplémentaires requises sont insérés. Lorsque plusieurs arguments ou paramètres sont nécessaires, le premier est mis en surbrillance pour vous inviter à le saisir et le remplacer. Vous devez sélectionner et remplacer les autres paramètres vous-même.

Certains types d'acteurs et d'Xtras de programmation offrent des termes qui ne figurent pas dans les menus locaux. Ces types d'acteurs et d'Xtras possèdent généralement leur propre documentation ; vous pouvez également trouver des informations à partir de Director.

Pour afficher la liste des Xtras disponibles :

- Emettez soit `put(_player.xtraList)` soit `trace(_player.xtraList)` dans la fenêtre Message.

Pour afficher la liste des Xtras de programmation disponibles :

- Emettez soit `put(_player.scriptingXtraList)` soit `trace(_player.scriptingXtraList)` dans la fenêtre Message.

Pour afficher la liste des méthodes et propriétés d'un Xtra :

- Dans le menu local Xtras de programmation, placez-vous sur un Xtra et, dans le menu secondaire, cliquez sur put interface. Les méthodes et propriétés de cet Xtra apparaissent dans la fenêtre Messages.

Saisie et modification de texte

La saisie et la modification de texte dans une fenêtre Script se font de la même manière que dans n'importe quel champ.

Les opérations d'édition les plus communes effectuées dans une fenêtre Script sont les suivantes :

- Pour sélectionner un mot, double-cliquez dessus.
- Pour sélectionner un script entier, choisissez Edition > Tout sélectionner.
- Pour commencer une nouvelle ligne, entrez un retour chariot.

- Dans Lingo, pour renvoyer une longue ligne de code à la ligne en insérant un symbole de continuation, appuyez sur Alt+Entrée (Windows) ou sur Option+Retour (Macintosh) à l'endroit où vous voulez insérer un retour à la ligne. Le symbole de continuation (\) qui apparaît indique que l'instruction continue sur la ligne suivante.

  Pour renvoyer une longue ligne de code à la ligne dans la syntaxe JavaScript, insérez un saut de ligne en appuyant sur Entrée (Windows) ou Retour (Macintosh). Le symbole de continuation Lingo cause une erreur de script dans les scripts de la syntaxe JavaScript.
- Pour trouver une gestionnaire dans le script actuel, choisissez son nom dans le menu local Passer au gestionnaire de la fenêtre Script.
- Pour compiler les scripts que vous venez de rédiger, cliquez sur le bouton Recompiler tous les scripts modifiés de la fenêtre Script ou fermez cette dernière. Lorsque vous éditez un script, un astérisque apparaît dans la barre de titre de la fenêtre Script, indiquant que le script doit être recompilé.
- Pour compiler tous les scripts d'une animation, sélectionnez Recompiler tous les scripts dans le menu Contrôle.
- Pour reformater un script avec la bonne mise en retrait, appuyez sur Tab dans la fenêtre Script.

  Director place automatiquement les instructions en retrait lorsque leur syntaxe est correcte. Si une ligne n'est pas correctement mise en retrait, la syntaxe de cette ligne est incorrecte.
- Pour ouvrir une seconde fenêtre Script, appuyez sur la touche Alt (Windows) ou Option (Macintosh), tout en cliquant sur le bouton Nouvel acteur dans la fenêtre Script. Cette opération peut s'avérer utile, par exemple lorsque vous éditez simultanément deux sections différentes d'un long script.
- Pour activer ou désactiver la numérotation des lignes, cliquez sur le bouton Numérotation des lignes.
- Pour activer ou désactiver la mise en couleur automatique, cliquez sur le bouton Mise en couleur automatique. La mise en couleur automatique affiche chaque type d'élément Lingo (propriétés, commandes, etc.) dans une couleur différente.
- Pour activer ou désactiver le formatage automatique, cliquez sur le bouton Format automatique. L'option Format automatique applique une mise en retrait correcte à vos scripts chaque fois que vous ajoutez un retour chariot ou que vous appuyez sur la touche Tab.

  Remarque : Les fonctions de coloration automatique et de formatage automatique ne s'appliquent pas aux codes JavaScript. Par conséquent, si vous créez des scripts à l'aide de la syntaxe JavaScript, les boutons Mise en couleur automatique et Formatage automatique sont désactivés dans la fenêtre Script et les termes tels que `function`, `var` et `this` ont la couleur par défaut.

Recherche de gestionnaires et de texte dans les scripts

La commande Rechercher du menu Edition permet de rechercher des gestionnaires ainsi que de rechercher et modifier du texte ou des gestionnaires.

Pour rechercher des gestionnaires dans les scripts :

1 Choisissez Edition > Rechercher > Gestionnaire.

  La boîte de dialogue Rechercher un gestionnaire apparaît.

  La colonne la plus à gauche de la boîte de dialogue Rechercher un gestionnaire affiche les noms de tous les gestionnaires de l'animation. La colonne du milieu affiche le numéro de l'acteur associé au script du gestionnaire, ainsi que son nom. La colonne la plus à droite affiche la distribution dans laquelle se trouve l'acteur.

2 Sélectionnez le gestionnaire à rechercher.

3 Cliquez sur Rechercher.

Le gestionnaire apparaît dans la fenêtre Script.

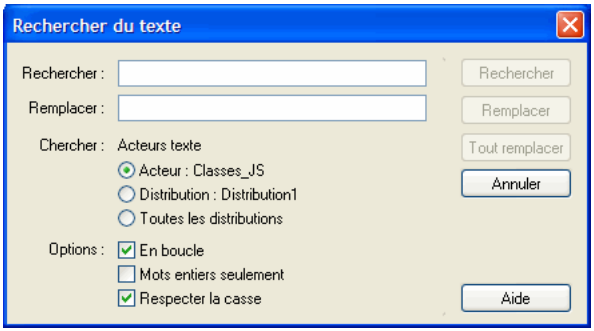
La barre de titre en haut de la fenêtre Script indique le type du script.

Pour rechercher du texte dans les scripts :

1 Activez la fenêtre Script.

2 Choisissez Edition > Rechercher > Texte.

La boîte de dialogue Rechercher du texte apparaît.



3 Saisissez le texte à rechercher dans le champ Rechercher, puis cliquez sur Rechercher.

La recherche ne fait pas de distinction entre les majuscules et les minuscules : `ThisHandler`, `thisHandler` et `THISHANDLER` ne sont pas différenciés lors de la recherche. Cliquez sur la case Respecter la casse pour que la recherche prenne en compte les majuscules et les minuscules.

Pour spécifier les acteurs dans lesquels effectuer la recherche :

- Sélectionnez l'option voulue sous Rechercher.

Pour reprendre la recherche au début une fois qu'elle atteint la fin :

- Sélectionnez l'option En boucle.

Pour ne rechercher que des mots entiers et non des fragments de mots correspondant au mot recherché :

- Sélectionnez l'option Mots entiers seulement.

Pour rechercher l'occurrence suivante du texte spécifié dans le champ Rechercher :

- Choisissez Edition > Poursuivre la recherche.

Pour trouver toutes les occurrences du texte sélectionné :

1 Sélectionnez le texte.

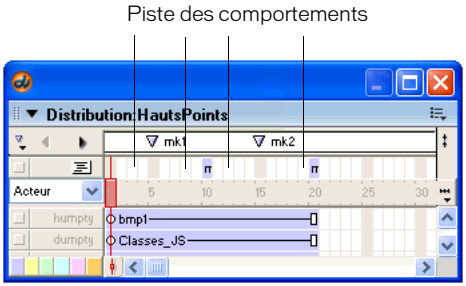
2 Choisissez Edition > Rechercher > Sélection.

Exécution d'opérations élémentaires

Vous trouverez ci-dessous les opérations élémentaires permettant de créer, associer et ouvrir des scripts.

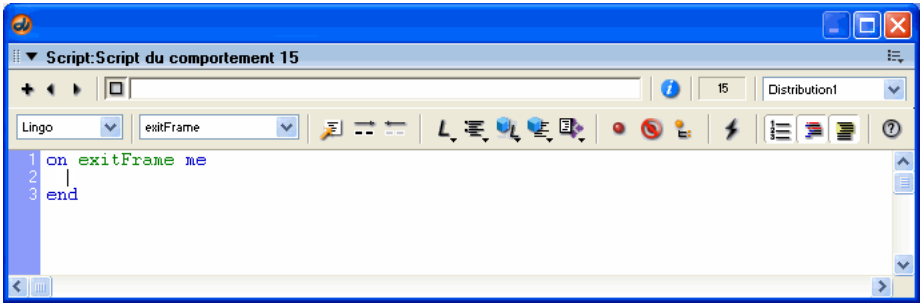
Pour créer un comportement d'image (script associé à une image) :

- Double-cliquez sur la piste des comportements dans l'image à laquelle vous souhaitez affecter le comportement.



Lorsque vous créez un nouveau comportement, celui-ci reçoit le premier numéro de distribution disponible dans la fenêtre Distribution active.

Lorsque vous créez un nouveau comportement d'image, la fenêtre Script apparaît et contient automatiquement le gestionnaire Lingo `on exitFrame`. La première ligne contient l'expression `on exitFrame`, suivie d'une ligne dans laquelle le curseur clignote puis d'une autre comprenant le mot `end`. Cela vous permet de facilement et rapidement associer un comportement Lingo commun à l'image. Pour que ce gestionnaire puisse fonctionner avec la syntaxe JavaScript, remplacez `on exitFrame` par `function exitFrame() {` puis `end` par `}`.



Un des comportements d'image les plus fréquents est celui qui maintient la tête de lecture en boucle dans une même image. Il peut s'avérer utile lorsque vous souhaitez que votre animation maintienne la lecture sur une même image en attendant que l'utilisateur clique sur un bouton ou que la lecture d'une vidéo numérique ou d'un fichier audio s'achève.

Pour maintenir la tête de lecture sur une seule image :

- Dans un comportement d'image, tapez l'instruction suivante sur la ligne qui suit directement l'instruction `on exitFrame` (Lingo) ou `function exitFrame()`(syntaxe JavaScript seulement) :

```
-- Syntaxe Lingo
_movie.go(_movie.frame)

// Syntaxe JavaScript
_movie.go(_movie.frame);
```

La propriété `frame` de l'objet Animation se rapporte à l'image actuellement occupée par la tête de lecture. Cette instruction demande essentiellement à la tête de lecture de « revenir au niveau de l'image active ».

Pour créer un comportement d'image-objet (script associé à une image-objet) :

- Dans le scénario ou sur la scène, sélectionnez l'image-objet à laquelle vous souhaitez associer le comportement. Choisissez ensuite Fenêtre > Inspecteur de comportement et choisissez Nouveau comportement dans le menu local Comportements.

Lorsque vous créez un nouveau comportement d'image-objet, la fenêtre Script apparaît et contient automatiquement le gestionnaire Lingo `on mouseUp`. La première ligne contient l'expression `on mouseUp`, suivie d'une ligne dans laquelle le curseur clignote puis d'une autre comprenant le mot `end`. Cela vous permet de facilement et rapidement associer un comportement commun à l'image-objet. Pour que ce gestionnaire puisse fonctionner avec la syntaxe JavaScript, remplacez `on mouseUp` par `function mouseUp() {` puis `end` par `}`.

Pour ouvrir un comportement afin de le modifier :

1. Double-cliquez sur le comportement dans la fenêtre Distribution.

   L'inspecteur de comportement apparaît.
2. Cliquez sur l'icône Script de l'acteur dans l'inspecteur de comportement.

   La fenêtre Script de l'acteur affiche le comportement.

Vous pouvez également ouvrir la fenêtre Script et faire défiler les scripts jusqu'à ce que le comportement recherché apparaisse.

Pour supprimer un comportement d'un emplacement dans le scénario :

- Sélectionnez l'emplacement, puis supprimez le script de la liste affichée dans l'inspecteur des propriétés (volet Comportement).

Pour associer des comportements existants à des images-objets ou à des images, effectuez l'une des opérations suivantes :

- Faites glisser un comportement d'une distribution vers une image-objet ou une image du scénario ou (pour les images-objets) vers une image-objet de la scène.
- Dans le scénario, sélectionnez les images-objets ou les images auxquelles vous souhaitez associer le comportement. Choisissez ensuite Fenêtre > Inspecteur de comportement et choisissez le comportement existant dans le menu local Comportements.

Pour créer un script d'animation (un script associé à une animation), effectuez l'une des opérations suivantes :

- Si le script actuel de la fenêtre Script est un script d'animation, cliquez sur le bouton Nouveau script de cette fenêtre. (Le bouton Nouveau script crée toujours un script du même type que le script actuel.)
- Si le script actuel de la fenêtre Script n'est pas un script d'animation, cliquez sur le bouton Nouveau script puis remplacez le type du nouveau script à l'aide du menu local Type du volet Script de l'inspecteur des propriétés.
- Si aucune image-objet ni aucun script n'est sélectionné dans la distribution, dans le scénario ou sur la scène, ouvrez alors une nouvelle fenêtre Script. Par défaut, un nouveau script d'animation sera créé.

Pour ouvrir un script d'animation ou un script parent afin de le modifier :

- Double-cliquez sur le script dans la fenêtre Distribution.

Pour modifier le type d'un script :

1. Sélectionnez le script dans la fenêtre Distribution ou ouvrez-le dans la fenêtre Script.
2. Cliquez sur l'onglet Script de l'inspecteur des propriétés et choisissez un type de script dans le menu local Type.

Pour faire défiler les scripts dans la fenêtre Script :

- Utilisez les flèches Acteur précédent et Acteur suivant situées en haut de la fenêtre Script pour vous déplacer vers l'avant ou l'arrière dans les scripts.

Pour dupliquer un script :

- Sélectionnez le script dans la fenêtre Distribution, puis choisissez Dupliquer dans le menu Edition.

Pour créer un script automatiquement associé à chaque image-objet créée à partir d'un acteur spécifique, associez le script à l'acteur proprement dit.

Pour créer un script associé à un acteur ou pour en ouvrir un, effectuez l'une des opérations suivantes :

- Cliquez sur le bouton droit de la souris (Windows) ou cliquez en maintenant la touche Ctrl enfoncée (Macintosh) sur un acteur dans la fenêtre Distribution, puis choisissez Script dans le menu contextuel.
- Sélectionnez un acteur dans la fenêtre Distribution et cliquez ensuite sur le bouton Script de l'acteur dans la fenêtre Distribution.

Utilisation de scripts liés

En plus des scripts stockés sous la forme d'acteurs internes, vous pouvez placer des scripts dans des fichiers texte externes et les lier à votre animation Director. Ces scripts liés sont similaires aux fichiers d'images ou de vidéo numérique que vous pouvez importer dans une animation Director.

Parmi les avantages de l'emploi de scripts liés, citons les suivants :

- Une personne peut travailler sur le fichier Director alors qu'une autre travaille sur le script.
- Il est facile d'échanger des scripts avec d'autres personnes.
- Vous pouvez contrôler les scripts séparément du fichier Director, dans une application de contrôle de code source telle que Microsoft Visual SourceSafe ou Perforce de Perforce Software. Ce type d'application évite que les différents programmeurs qui travaillent ensemble sur un projet Director écrasent le travail des autres.

Les scripts liés ne sont utilisés par Director qu'en cours de création. A l'exécution, les projections Director et le lecteur Macromedia Shockwave utilisent une copie interne spéciale des données du script stockée dans l'animation. De cette façon, il n'est pas nécessaire de distribuer vos scripts liés avec vos animations, et il est impossible à l'utilisateur final de les copier.

Pour importer un script sous la forme d'un fichier texte lié :

1 Choisissez Fichier > Importer.

2 Choisissez Script comme type de fichier à importer.

3 Sélectionnez le ou les fichiers de script que vous souhaitez importer.

Vous pouvez importer des fichiers possédant les extensions .txt, .ls ou .js. L'extension .ls est l'extension désignant les scripts liés de Director.

Pour créer une liste des fichiers à importer, vous pouvez utiliser les boutons Ajouter et Tout ajouter. Une telle liste est notamment utile si vous souhaitez importer des scripts de plusieurs endroits différents.

4 Choisissez Lier au fichier externe dans le menu local Médias.

5 Cliquez sur Importer.

Vous pouvez modifier les scripts liés de manière normale dans la fenêtre Script de Director. Les modifications que vous apportez sont écrites dans les fichiers externes à chaque fois que vous enregistrez l'animation Director. (Si vous avez importé le script lié depuis un serveur UNIX, les fins de ligne UNIX sont préservées.) Si vous importez un script dont le fichier texte est verrouillé, il vous sera impossible de le modifier dans Director.

Il est impossible d'appliquer des couleurs de texte personnalisées aux scripts liés dans la fenêtre Script. Par contre, la fonction de coloration automatique des scripts est activée pour les scripts liés.

Pour transformer un acteur script interne en acteur script lié externe :

1 Sélectionnez l'acteur interne et cliquez sur l'onglet Script de l'inspecteur des propriétés.

2 Cliquez sur Lier le script sous.

3 Entrez le nom du fichier dans la boîte de dialogue Enregistrer le script sous.

4 Cliquez sur Enregistrer.

Pour recharger un script lié après sa modification :

- Utilisez la propriété `unload()` de l'objet Image-objet.

Si un script lié est modifié en dehors de Director, vous pouvez le recharger avec la méthode `unload()` dans la fenêtre Messages. L'instruction suivante déchargera puis rechargera l'acteur script `monScript` :

```
-- Syntaxe Lingo
member("monScript").unload()

// Syntaxe JavaScript
member("monScript").unload();
```

CHAPITRE 4
Débogage de scripts dans Director

Les scripts ne répondent pas toujours immédiatement aux instructions. Le script présente souvent une erreur de syntaxe : il s'agit généralement d'un mot mal écrit ou d'une partie du script absente. Il arrive aussi que le script fonctionne mais ne produise pas le résultat escompté. Des erreurs ou des bogues survenant presque toujours lors de la rédaction de scripts, il est recommandé de prévoir le temps nécessaire au débogage lors du développement des projets multimédia.

Au fur et à mesure de votre apprentissage, vous rencontrerez probablement d'autres types de problèmes, car lorsque vous maîtriserez un sujet, vous commencerez seulement à découvrir les autres. Toutefois, les principales techniques de dépannage présentées dans cette section sont destinées à la fois aux utilisateurs débutants et expérimentés.

Le meilleur moyen de corriger une erreur dans vos scripts varie d'une situation à l'autre. Il n'existe pas de procédure standard permettant de résoudre un problème. Vous devrez utiliser plusieurs des outils et techniques présentés plus bas :

- Présentation générale et détaillée de l'interaction des scripts dans une animation
- Expérimentation et pratique des principales méthodes de débogage

Les outils suivants sont destinés à vous aider à identifier les problèmes dans les scripts :

- Lorsque la fonction de suivi est activée, la fenêtre Messages affiche un enregistrement des images lues et des gestionnaires en cours d'exécution dans l'animation.
- La fenêtre Débogueur affiche les valeurs des variables globales, les propriétés du script actuellement en cours d'exécution, la séquence de gestionnaires exécutée pour parvenir au niveau actuel, ainsi que la valeur des variables et des expressions que vous avez sélectionnées.
- La fenêtre Script vous permet de saisir des commentaires, d'insérer des points d'arrêt dans le script et de sélectionner des variables dont la valeur apparaît dans l'inspecteur d'objet.
- L'inspecteur d'objet vous permet d'afficher et de définir les valeurs des objets et des propriétés que vous avez sélectionnés.

Bonnes habitudes de rédaction de scripts

De bonnes habitudes vous permettront d'éviter, dès le départ, bon nombre de problèmes dans la rédaction de scripts.

- Veillez à rédiger les scripts par petits lots et testez directement chacun de vos scripts, au fur et à mesure que vous les créez. Cette procédure permettra d'isoler les éventuels problèmes afin de les identifier plus facilement.
- Insérez des commentaires expliquant l'objectif des instructions et des valeurs du script. Le script sera alors plus facile à comprendre, lorsque vous y reviendrez ultérieurement ou lorsqu'un autre utilisateur devra l'utiliser. Par exemple, les commentaires des instructions suivantes indiquent l'objectif de la structure `if…then` et clarifient la répétition de la boucle :

```
-- Syntaxe Lingo
-- Boucle jusqu'à ce que la touche "s" soit enfoncée
repeat while not(_key.keyPressed("s"))
  _sound.beep()
end repeat

// Syntaxe JavaScript
// Boucle jusqu'à ce que la touche "s" soit enfoncée
while(!_key.keyPressed("s")) {
  _sound.beep()
}
```

- Vérifiez si la syntaxe du script est correcte. Utilisez les menus locaux de la fenêtre Script pour insérer des versions préformatées des éléments de scripting. Consultez les rubriques des API de ce référentiel pour vérifier si les instructions sont rédigées correctement.
- Utilisez des noms de variable qui indiquent l'objectif d'une variable. Par exemple, une variable contenant un nombre devrait porter un nom tel que `nouveauNombre` plutôt que `ABC`.

Opérations de débogage de base

Le processus de débogage implique des étapes de stratégie et d'analyse, et non une procédure standard rigoureuse. Cette section décrit les opérations de débogage fondamentales servant à déboguer tous les types de code, et pas uniquement le code Lingo ou JavaScript.

Avant d'apporter une modification majeure à une animation, veillez à toujours en faire une copie de sauvegarde. Il est recommandé de nommer les copies par incréments (par exemple nomFichier_01.dir, nomFichier_02.dir, nomFichier_03.dir, et ainsi de suite) afin de pouvoir suivre les diverses étapes d'une animation.

Identification du problème

Cela peut paraître évident, mais rappelons que la première chose à faire, lors d'une procédure de débogage, est d'identifier le problème. La fonction d'un bouton est-elle faussée ? L'animation accède-t-elle à une autre image que celle prévue ? L'édition d'un champ s'avère-t-elle impossible ?

Essayez aussi de comparer la fonction escomptée d'un script donné à sa fonction réelle. Cette opération vous aidera à déterminer clairement votre objectif et les parties de cet objectif qui n'ont pas été réalisées.

Si vous avez copié un script ou une partie de ce script à partir d'une autre animation ou d'un exemple écrit, vérifiez si ce script a été conçu pour des conditions bien spécifiques. Il était peut-être nécessaire de programmer une piste d'image-objet. Il se peut également que les noms d'acteur doivent suivre une convention stylistique spécifique.

Localisation du problème

Pour localiser un problème, procédez comme suit :

- Remontez la chaîne et essayez de localiser l'emplacement auquel le problème semble avoir commencé.
- Utilisez la fenêtre Messages pour suivre les images parcourues par l'animation et identifier les gestionnaires exécutés par vos scripts.
- Déterminez le comportement présumé des scripts et ce qui, dans ces instructions, est associé au problème. Par exemple, si un acteur texte ne peut pas être édité alors qu'il devrait l'être, localisez l'emplacement de la propriété `editable` de l'acteur dans votre script.
- Si vous ne parvenez pas à modifier comme vous le souhaitiez une image-objet sur la scène, vérifiez si la méthode `updateStage` n'est pas requise à un emplacement précis.
- Vérifiez si le problème survient sur tous les ordinateurs ou sur un seul. Tâchez de définir si le problème survient uniquement lorsque l'affichage est réglé sur millions de couleurs. Il se peut qu'un élément de l'ordinateur interfère avec l'application.

Concentrez-vous sur des lignes de script précises en insérant un point d'arrêt, c'est-à-dire un point où le script interrompt son exécution et appelle la fenêtre Débogueur, dans une ligne. Ceci vous permettra d'analyser les conditions à ce point précis, avant de poursuivre l'opération du script. Pour plus d'informations sur l'insertion de points d'arrêt dans un script, consultez *Débogage dans la fenêtre Débogueur*, page 105.

Résolutions de problèmes simples

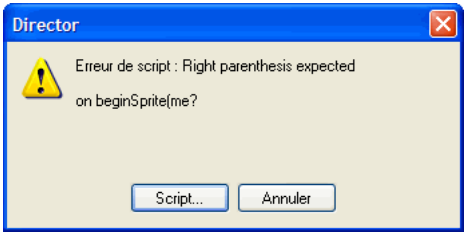
Lorsque vous découvrez un bogue, consultez tout d'abord les résolutions de problèmes simples.

Le premier test de débogage s'effectue lorsque vous compilez votre script. Pour compiler votre script, utilisez l'un des méthodes suivantes :

- Dans la fenêtre Script, cliquez sur Recompiler tous les scripts modifiés.
- Cliquez sur le bouton Recompiler tous les scripts dans le menu Contrôle.
- Appuyez sur Maj+F8.
- Fermez la fenêtre Script.

Il est généralement recommandé de compiler des scripts en utilisant l'une des trois premières méthodes. Pour pouvoir utiliser la quatrième option, vous devez fermer la fenêtre Script à chaque fois que vous voulez compiler un script.

Lorsque vous compilez votre script, Macromedia Director MX 2004 présente un message d'erreur si le script contient une syntaxe incorrecte. Le message affiche généralement la ligne dans laquelle le problème a été détecté initialement. Un point d'interrogation apparaît au point précis où Director a initialement détecté le problème.



Par exemple, la première ligne du message précédent vous indique que l'erreur en question est une erreur de syntaxe et vous donne une explication. La deuxième ligne du message d'erreur affiche la ligne de code contenant l'erreur de syntaxe.

Recherche d'erreurs de syntaxe

Les erreurs de syntaxe sont certainement à l'origine des bogues les plus courants dans le scripting. Lorsqu'un script échoue, il est recommandé de vérifier immédiatement les points suivants :

- Les termes sont écrits correctement, les espaces sont placés aux endroits appropriés et la ponctuation correcte est utilisée. Director ne peut pas interpréter une syntaxe incorrecte.
- Des guillemets sont placés de part et d'autre des noms d'acteurs, des libellés et des chaînes dans l'instruction.
- Tous les paramètres requis sont présents. A chaque élément doivent être associés des paramètres spécifiques. Consultez les informations sur les API de ce référentiel pour savoir si un élément nécessite d'autres paramètres.

Recherche de bogues simples

Si votre script se compile sans afficher de message d'erreur, il risque de contenir un bogue. Si votre script ne produit pas les résultats escomptés, vérifiez les points suivants :

- Les valeurs des paramètres sont-elles correctes ? Par exemple, l'utilisation d'une valeur incorrecte pour le nombre de bips sonores que doit générer la méthode `beep` produit un autre nombre de bips.
- Les valeurs sujettes à modifications, telles que les variables et le contenu d'acteurs texte, ont-elles les valeurs escomptées ? Vous pouvez afficher leurs valeurs dans l'inspecteur d'objet en sélectionnant le nom de l'objet et en cliquant sur Inspecteur d'objet dans la fenêtre Script, ou dans la fenêtre Messages en utilisant les fonctions `put()` ou `trace()`.
- Les éléments de scripting se comportent-ils normalement ? Vous pouvez vérifier leur comportement dans les rubriques consacrées aux API, dans ce référentiel.
- Si le script est rédigé dans la syntaxe JavaScript, il risque de contenir une erreur de casse. La syntaxe JavaScript est sensible à la casse, ce qui signifie que les méthodes, fonctions, propriétés et variables doivent être saisies en respectant l'emploi des majuscules et minuscules.

  Si vous appelez une méthode ou fonction en ne respectant pas la casse, vous obtenez une erreur de script.

Si vous essayez d'accéder à une variable ou une propriété en utilisant une casse incorrecte, il est possible que vous ne receviez pas d'erreur, mais votre script risque de ne pas avoir le comportement voulu. Par exemple, le gestionnaire `mouseUp` suivant contient une instruction qui essaie d'accéder à la propriété `étiquetteElement` en utilisant une casse incorrecte. Ce script ne renverra pas d'erreur, mais créera automatiquement une nouvelle variable contenant une casse incorrecte. La valeur de la nouvelle variable est `undefined`.

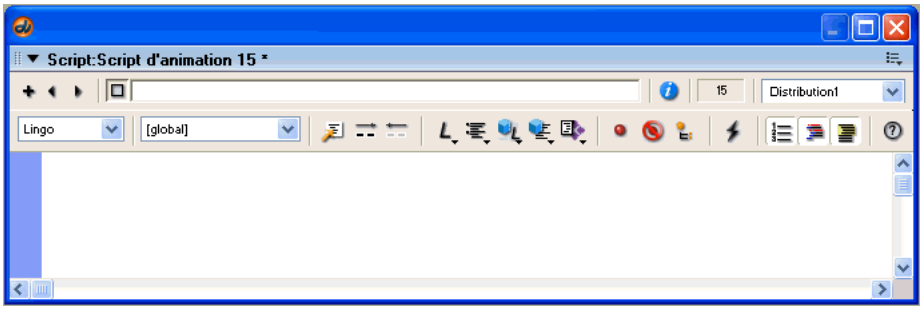
```
// Syntaxe JavaScript
fonction beginSprite() {
  this.étiquetteElement = "Plans";
}

function mouseUp() {
  trace(this.étiquetteElement); // crée la propriété étiquetteElement
}
```

Débogage dans la fenêtre Script

La fenêtre Script propose un certain nombre de fonctionnalités qui vous permettent de déboguer vos scripts.

Pour ouvrir la fenêtre Messages :

- Choisissez Fenêtre > Script.



Pour rédiger un commentaire associé à la ligne de code courante :

- Cliquez sur Insérer une marque de commentaire.

Pour supprimer le commentaire de la ligne de code courante :

- Cliquez sur Supprimer la marque de commentaire.

Pour activer ou désactiver les points d'arrêt dans la ligne de code courante :

- Cliquez sur Activer/désactiver le point d'arrêt.

Pour désactiver tous les points d'arrêt :

- Cliquez sur Ignorer les points d'arrêt.

Pour ajouter l'expression ou la variable sélectionnée à l'inspecteur d'objet :

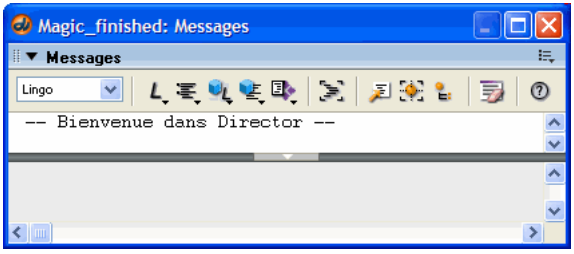
- Cliquez sur Inspecteur d'objet.

Débogage dans la fenêtre Messages

La fenêtre Messages vous permet de tester les commandes de scripting et d'en contrôler le processus lors de la lecture d'une animation.

Pour ouvrir la fenêtre Messages :

- Choisissez Fenêtre > Messages.



Gestion de la fenêtre Messages

La fenêtre Messages contient un volet de saisie et un volet de résultat. Le contenu du volet de saisie est modifiable. Le contenu du volet de résultat est en lecture seule. Le seul moyen d'afficher le texte dans le volet de résultat est d'appeler la fonction `put()` ou `trace()`.

Vous pouvez ajuster la taille des volets de saisie et de résultat en faisant glisser le séparateur horizontal situé entre les deux volets.

Pour redimensionner le volet de résultat :

- Faites glisser le séparateur horizontal vers un nouvel emplacement.

Pour masquer complètement le volet de résultat :

- Cliquez sur le bouton Réduire/Agrandir, au centre du séparateur horizontal.

  Lorsque le volet de résultat est masqué, les sorties des scripts en cours d'exécution sont affichées dans le volet de saisie.

Pour afficher le volet de résultat lorsqu'il est masqué :

- Cliquez de nouveau sur le bouton Réduire/Agrandir.

Pour effacer le contenu de la fenêtre Messages :

- Cliquez sur le bouton Effacer.

  Si le volet de résultat est visible, son contenu est effacé.

  Si le volet de résultat n'est pas visible, le contenu du volet de saisie est effacé.

Pour effacer une partie du contenu du volet de résultat :

1. Sélectionnez le texte à effacer.
2. Appuyez sur la touche Retour arrière ou Suppr.

Pour copier du texte dans le volet de saisie ou de résultat :

1. Sélectionnez le texte.
2. Choisissez Edition > Copier.

Test de scripts dans la fenêtre Messages

Vous pouvez tester les instructions Lingo et JavaScript pour vérifier leur fonctionnement en les saisissant dans la fenêtre Messages et en observant les résultats. Lorsque vous saisissez une commande dans la fenêtre Messages, Director l'exécute immédiatement, qu'une animation soit ou non en cours d'exécution.

Avant de saisir les instructions que vous voulez tester, vous devez d'abord sélectionner la syntaxe de scripting (Lingo ou JavaScript) à tester.

Pour sélectionner la syntaxe de scripting :

1 Dans le menu déroulant Syntaxe de script, sélectionnez Lingo ou JavaScript.

Pour tester une instruction d'une ligne :

1 Saisissez directement l'instruction dans la fenêtre Messages.

2 Appuyez sur Entrée (Windows) ou sur Retour (Macintosh). Director exécute l'instruction.

Si l'instruction est valide, la fenêtre Messages affiche le résultat de l'instruction dans le volet de résultat, en bas de l'écran. Si le script n'est pas valide, un message d'erreur apparaît.

Par exemple, si vous saisissez l'instruction suivante dans la fenêtre Messages :

```
-- Syntaxe Lingo
put 50+50

// Syntaxe JavaScript
put 50+50
```

puis que vous appuyez sur la touche Entrée (Windows) ou Retour (Macintosh), le résultat apparaît dans le volet de résultat :

```
-- Syntaxe Lingo
-- 100

// Syntaxe JavaScript
// 100
```

Si vous saisissez l'instruction suivante dans la fenêtre Messages :

```
-- Syntaxe Lingo
_movie.stage.bgColor = 255

// Syntaxe JavaScript
_movie.stage.bgColor = 255;
```

puis que vous appuyez sur la touche Entrée (Windows) ou Retour (Macintosh), la scène apparaît en noir.

Vous pouvez tester plusieurs lignes de code en une seule opération en copiant et collant des instructions dans la fenêtre Messages ou en appuyant simultanément sur les touches Maj et Retour (Entrée) après chaque ligne de code.

Pour exécuter plusieurs lignes de code par copier/coller :

1 Copiez les lignes de code dans le Presse-papiers.

2 Entrez une ligne vierge dans la fenêtre Messages.

3 Collez le code dans le volet de saisie de la fenêtre Messages.

4 Placez le point d'insertion à la fin de la dernière ligne de code.

5 Appuyez sur Ctrl+Entrée (Windows) ou Ctrl+Retour (Macintosh). Director trouve la première ligne vierge au-dessus du point d'insertion et exécute successivement chaque ligne de code après la ligne vierge.

Pour saisir plusieurs lignes de code manuellement :

1 Entrez une ligne vierge dans la fenêtre Messages.

2 Entrez la première ligne de code.

3 Appuyez sur Maj+Retour (Entrée) à la fin de la ligne.

4 Répétez les étapes 2 et 3 jusqu'à la dernière ligne de code.

5 Appuyez sur Ctrl+Entrée (Windows) ou Ctrl+Retour (Macintosh). Director trouve la première ligne vierge au-dessus du point d'insertion et exécute successivement chaque ligne de code après la ligne vierge.

Vous pouvez tester un gestionnaire sans exécuter l'animation, en écrivant le gestionnaire dans une fenêtre de script d'animation ou de script de comportement, puis en l'appelant depuis la fenêtre Messages.

Pour tester un gestionnaire :

1 Copier et collez ou saisissez manuellement un gestionnaire à plusieurs lignes dans la fenêtre Messages, comme indiqué dans les deux procédures précédentes.

2 Placez le point d'insertion à la fin de la dernière ligne de code.

3 Appuyez sur Entrée (Windows) ou sur Retour (Macintosh). Le gestionnaire est exécuté.

Toutes les sorties provenant de l'instruction `put()` ou `trace()`dans le gestionnaire sont affichées dans la fenêtre Messages.

Tout comme la fenêtre Script, la fenêtre Messages contient des menus locaux des commandes de scripting. Lorsque vous sélectionnez une commande dans l'un de ces menus locaux, la commande apparaît automatiquement dans la fenêtre Messages, en présentant le premier argument fourni. Plusieurs menus sont disponibles et permettent un accès rapide au catalogue complet des termes de scripting.

Les menus locaux comprennent les options suivantes :

- Lingo par ordre alphabétique : toutes les commandes, à l'exception de Lingo 3D, présentées par ordre alphabétique.
- Lingo par catégorie : toutes les commandes, à l'exception de Lingo 3D, présentées par catégorie.
- Lingo 3D par ordre alphabétique : tous les termes Lingo 3D, présentés par ordre alphabétique.
- Lingo 3D par catégorie : tous les termes Lingo 3D, présentés par catégorie.
- Les Xtras de programmation incluent les méthodes et propriétés de tous les Xtras de programmation trouvés, qu'il s'agisse d'Xtras Macromedia ou autres.

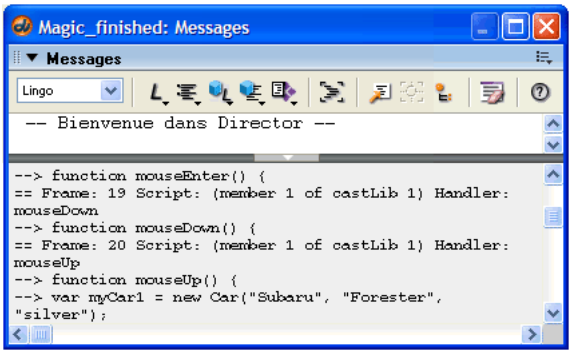
Remarque : Les Xtras de programmation figurant dans le menu local sont limités à ceux qui supportent la méthode `interface()` et dont les noms apparaissent effectivement dans le menu local. Bien que certains types de médias d'acteurs tels 3D et DVD prennent également en charge la méthode `interface()`, ils ne figurent pas dans le menu local Xtras de programmation parce qu'ils ne sont pas implémentés en tant que Xtras de programmation dans Director.

Gestion de scripts dans la fenêtre Messages

Vous pouvez régler le volet de résultat de la fenêtre Messages de manière à afficher un enregistrement des instructions qu'une animation exécute lors de sa lecture. Ceci s'avère utile pour faire le suivi du flux de votre code et examiner le résultat d'instructions spécifiques. Il existe deux manières d'effectuer cette opération.

Pour afficher des instructions dans le volet de résultat, procédez comme suit :

- Dans la fenêtre Messages, cliquez sur Trace.
- Donnez à la propriété `traceScript` de l'objet Animation la valeur `TRUE`.



Les entrées placées après un double signe égal (==) indiquent ce qui s'est produit dans l'animation, par exemple la dernière image ouverte, le script en cours d'exécution ou le résultat d'une méthode ou de la définition d'une valeur.

Par exemple, la ligne suivante contient plusieurs renseignements :

```
== Frame: 39 Script: 1 Handler: mouseUp
```

- L'animation a accédé à l'image 39.
- L'animation a exécuté le script 1, le premier script associé à l'image.
- L'animation a exécuté le gestionnaire `mouseUp` dans le script 1 après que l'animation a accédé à l'image.

Les entrées situées après une flèche constituée d'un double tiret et d'un signe supérieur à (`-->`) indiquent les lignes de votre code qui ont été exécutées. Par exemple, les lignes Lingo suivantes :

```
--
--> if leftSide < 10 then
--> if leftSide < 200 then
--> _movie.go("Début du jeu")
```

indiquent que ces instructions Lingo ont été exécutées. Supposons que vous souhaitiez déterminer la raison pour laquelle la tête de lecture n'a pas accédé à l'image appelée « Début du jeu ». Si la ligne `--> _movie.go("Début du jeu")` ne s'est pas affichée dans la fenêtre Messages, il se peut que la condition de l'instruction précédente ne soit pas celle escomptée.

Le volet de résultat de la fenêtre Messages peut contenir une grande quantité de texte lorsque la fonction de suivie est activée. Pour supprimer le contenu du volet de résultat, cliquez sur le bouton Effacer. Si le volet de résultat n'est pas visible, le contenu du volet de saisie est effacé.

Vous pouvez retracer le suivi des valeurs de variables et d'autres objets en sélectionnant le nom de l'objet dans la fenêtre Messages et en cliquant sur le bouton Inspecteur d'objet. L'objet est ajouté à l'inspecteur d'objet, où sa valeur sera affichée et actualisée lors de la lecture de l'animation. Pour plus d'informations sur l'inspecteur d'objet, veuillez consulter *Débogage de l'inspecteur d'objet*, page 102.

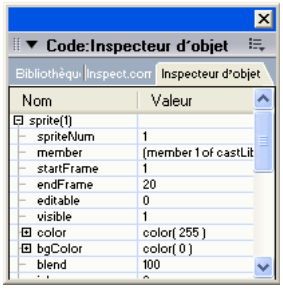
Lorsque vous êtes en mode de débogage, vous pouvez retracer le suivi des modifications d'une variable en la sélectionnant dans la fenêtre Messages et en cliquant sur le bouton Surveiller l'expression. Director ajoute ensuite la variable au volet Surveillance dans la fenêtre Débogueur, dans laquelle sa valeur est affichée et actualisée pendant que vous travaillez dans la fenêtre Débogueur. Pour plus d'informations sur le volet Surveillance, veuillez consulter *Débogage dans la fenêtre Débogueur*, page 105.

Débogage de l'inspecteur d'objet

L'inspecteur d'objet permet d'afficher et de définir les propriétés d'un grand nombre d'objets ne s'affichant pas dans l'inspecteur des propriétés. Il s'agit notamment des objets de scripting tels que les variables globales, les listes, les objets enfants de scripts parents, toutes les propriétés d'acteur 3D, les propriétés d'images-objets, les expressions de script, etc. En outre, l'inspecteur d'objet affiche les modifications apportées aux propriétés d'objet lors de la lecture de l'animation, par exemple les modifications dues aux scripts ou apportées aux propriétés de scénario de l'image-objet. Ces types de modifications ne sont pas affichés dans l'inspecteur des propriétés lors de la lecture de l'animation.

Pour ouvrir l'Inspecteur d'objet :

- Choisissez Fenêtre > Inspecteur d'objet.



Présentation détaillée des structures d'objets

L'inspecteur d'objet est très utile pour comprendre la structure d'objets complexes. Par exemple, les acteurs 3D contiennent un grand nombre de couches de propriétés. L'inspecteur d'objet affichant une représentation visuelle de la structure imbriquée de ces propriétés, il vous aide à comprendre l'organisation de ces propriétés, ainsi que leurs interactions. Il est important de comprendre la structure des propriétés des objets dans Director lors de la rédaction des scripts.

La possibilité d'examiner le changement de valeur des propriétés lors de la lecture d'une animation est pratique pour comprendre le fonctionnement de l'animation. Cela s'avère particulièrement utile lors des procédures de test et de débogage des scripts, car vous pouvez constater les changements de valeurs en fonction des scripts que vous avez rédigés.

La fenêtre Débogueur de Director affiche également ces informations, mais uniquement en mode de débogage. Pour plus d'informations sur le débogage, veuillez consulter *Débogage avancé*, page 111.

Objets visibles

Voici quelques exemples d'objets que vous pouvez entrer dans l'inspecteur d'objet :

- Images-objets, telles que `sprite(3)`
- Acteurs, tels que `member("3d")`
- Variables globales, telles que `gMaListe`
- Objets enfants, tels que `gMonEnfant`
- Objets Macromedia Flash, tels que `gMonObjetFlash` ; pour plus d'informations sur l'utilisation d'objets Flash dans Director, veuillez consulter la rubrique Utilisation de Director dans le panneau d'aide de Director.
- Expressions de script, telles que `sprite(7).blend`

Affichage d'objets

Vous pouvez afficher un objet dans l'inspecteur d'objet de trois manières. Vous pouvez faire glisser les éléments directement dans l'inspecteur d'objet, saisir manuellement le nom d'un de ses éléments ou utiliser le bouton Inspecteur d'objet dans les fenêtres Messages et Script.

Pour faire glisser un élément dans l'inspecteur d'objet, effectuez l'une des opérations suivantes :

- Sélectionnez une image-objet dans la fenêtre Scénario et faites-la glisser dans l'inspecteur d'objet.
- Sélectionnez un acteur dans la fenêtre Acteur et faites-le glisser dans l'inspecteur d'objet.
- Sélectionnez le nom d'un objet dans les fenêtres Script, Messages ou Texte et faites-le glisser dans l'inspecteur d'objet.

Pour entrer manuellement un objet dans l'inspecteur d'objet :

1. Double-cliquez dans la première cellule vide de la colonne Objet de l'inspecteur d'objet.
2. Tapez le nom de l'objet dans la cellule. Utilisez le même nom que celui utilisé pour cet objet dans vos scripts.
3. Appuyez sur Entrée (Windows) ou sur Retour (Macintosh). Si l'objet possède des sous-propriétés, un signe plus (+) est affiché sur sa gauche.

4 Cliquez sur le signe plus. Les propriétés de l'objet s'affichent en dessous de celui-ci. Les propriétés contenant des sous-propriétés sont affichées avec un signe plus sur leur gauche. Cliquez sur chaque signe plus pour afficher les sous-propriétés.

Pour voir un objet à l'aide du bouton Inspecteur d'objet :

1 Dans la fenêtre Script, mettez en surbrillance la partie d'une instruction se rapportant à un objet.

2 Dans la fenêtre Script, cliquez sur Inspecteur d'objet. Si l'objet possède des sous-propriétés, un signe plus (+) est affiché sur sa gauche.

3 Cliquez sur le signe plus. Les propriétés de l'objet s'affichent en dessous de celui-ci. Les propriétés contenant des sous-propriétés sont affichées avec un signe plus sur leur gauche. Cliquez sur chaque signe plus pour afficher les sous-propriétés.

Remarque : Si vous examinez beaucoup d'objets ou de gros objets individuels dans l'Inspecteur d'objet, vous risquez de causer des problèmes de performances durant la programmation, particulièrement lorsque l'option Interrogation automatique est activée. Par exemple, lorsque vous examinez une liste contenant 10 000 entrées, l'affichage de Director peut sembler lent.

Parcourir des objets

Vous pouvez également accéder au contenu de l'inspecteur d'objet à l'aide des touches fléchées de votre clavier.

Pour monter ou descendre dans la liste des éléments :

- Utilisez les touches fléchées Haut et Bas.

Pour visualiser les sous-propriétés d'un élément :

- Sélectionnez l'élément et appuyez sur la touche fléchée Droite.

Pour masquer les sous-propriétés d'un élément :

- Sélectionnez l'élément et appuyez sur la touche fléchée Gauche.

Utilisation de Interrogation automatique

Les propriétés système, telles que `milliseconds` et `colorDepth` ne sont actualisées dans l'inspecteur d'objet que lorsque l'option Interrogation automatique est activée. L'utilisation de l'interrogation automatique augmente la charge de travail du processeur, ce qui risque de ralentir les performances de votre animation lorsque vous ajoutez un certain nombre de propriétés système à l'inspecteur d'objet.

Pour activer l'option Interrogation automatique :

1 Cliquez du bouton droit de la souris (Windows) ou cliquez en maintenant la touche Ctrl enfoncée (Macintosh) dans l'inspecteur d'objet. Le menu contextuel de l'inspecteur d'objet apparaît.

2 Sélectionnez Interrogation automatique dans le menu contextuel. Lorsque l'option Interrogation automatique est activée, une coche apparaît à côté de l'option correspondante dans le menu.

Pour désactiver l'option Interrogation automatique :

- Sélectionnez à nouveau Interrogation automatique dans le menu contextuel.

Modification des valeurs d'un objet ou d'une propriété :

Vous pouvez définir la valeur d'un objet ou d'une propriété dans l'inspecteur d'objet en saisissant une nouvelle valeur dans le champ situé à droite du nom de l'objet ou de la propriété.

Pour définir la valeur d'un objet ou d'une propriété :

1. Double-cliquez sur la valeur, à droite du nom de l'élément.
2. Saisissez la nouvelle valeur de l'élément.
3. Appuyez sur Entrée (Windows) ou sur Retour (Macintosh). La nouvelle valeur est définie et est immédiatement reflétée dans l'animation.

Vous pouvez saisir une expression de script comme valeur pour un élément. Par exemple, vous pouvez définir la valeur de `sprite(3).locH` sur l'expression `sprite(8).locH + 20`.

Suppression d'objets

Vous pouvez également retirer des éléments de l'inspecteur d'objet.

Pour retirer un élément de l'inspecteur d'objet :

- Sélectionnez l'élément et appuyez sur la touche Retour arrière (Windows) ou Suppr (Macintosh).

Pour effacer tout le contenu de l'inspecteur d'objet :

- Cliquez du bouton droit de la souris (Windows) ou cliquez en maintenant la touche Ctrl enfoncée (Macintosh) dans l'inspecteur d'objet et choisissez Effacer tout dans le menu local.

Lorsque vous ouvrez une animation différente de celle sur laquelle vous travaillez actuellement, les objets précédemment entrés dans l'inspecteur d'objet y sont conservés. Ceci facilite la comparaison de différentes versions d'une même animation. Lorsque vous quittez Director, les éléments de l'inspecteur d'objet ne sont pas conservés.

Débogage dans la fenêtre Débogueur

La fenêtre Débogueur constitue un mode spécial de la fenêtre Script. Elle fournit plusieurs outils permettant de localiser la cause de vos problèmes dans les scripts. Le Débogueur vous permet de localiser rapidement les éléments de votre code qui sont à l'origine du problème. La fenêtre Débogueur permet de rédiger des scripts ligne par ligne, d'ignorer les gestionnaires imbriqués, de modifier le texte des scripts et de visualiser les valeurs des variables et d'autres objets au fur et à mesure de leur modification. L'apprentissage des outils de la fenêtre Débogueur permet d'accroître l'efficacité de votre programmation.

La fenêtre Débogueur permet également de localiser et de corriger les erreurs dans vos scripts. Elle comprend plusieurs outils qui vous permettront d'effectuer les opérations suivantes :

- Afficher la partie du script contenant la ligne de code courante.
- Retracer la séquence des gestionnaires appelés avant le gestionnaire courant.
- Exécuter certaines parties du gestionnaire courant.
- Exécuter certaines parties des gestionnaires appelés depuis le gestionnaire courant.
- Afficher la valeur d'une variable locale, d'une variable globale ou d'une propriété associée au code qui fait l'objet de la recherche.

Saisie du mode de débogage

La fenêtre Débogueur ne s'affiche que lorsqu'un script est interrompu. Cette interruption survient lorsque Director détecte une erreur ou un point d'arrêt dans un script.

La boîte de dialogue Erreur de script apparaît lorsqu'une erreur de script survient. Cette boîte de dialogue affiche les informations associées à l'erreur détectée, vous demande si vous souhaitez corriger le bogue dans le script, modifier le script dans la fenêtre Script ou annuler.

Pour passer en mode de débogage, effectuez l'une des opérations suivantes :

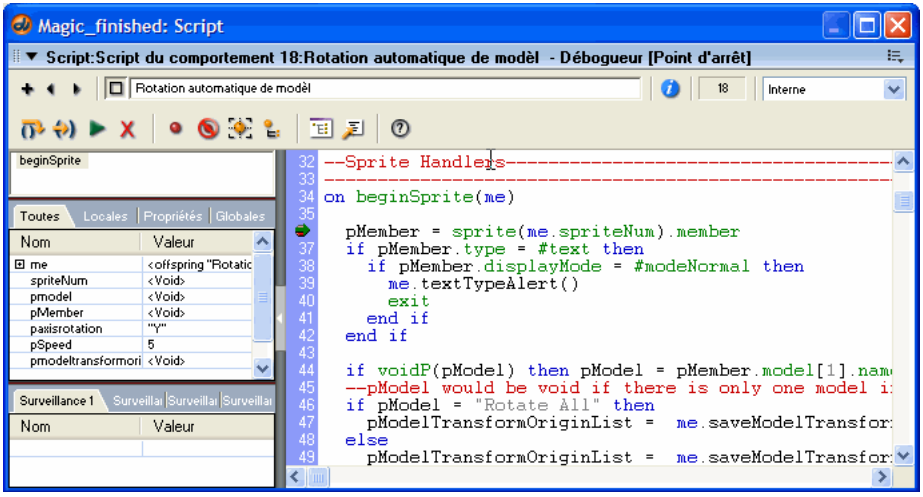
- Cliquez sur Déboguer dans la boîte de dialogue Erreur de script.
- Placez un point d'arrêt dans un script.

Lorsque Director détecte un point d'arrêt en cours d'exécution, l'exécution du script est interrompue et la fenêtre Script passe en mode de débogage. La lecture de l'animation se poursuit, mais l'exécution de vos scripts est interrompue jusqu'à ce que vous utilisiez la fenêtre Débogueur pour indiquer la procédure que Director doit suivre. Si plusieurs fenêtres Script sont ouvertes, Director recherche celle contenant le script dans lequel le point d'arrêt a été détecté et fait passer cette fenêtre en mode de débogage.

Pour ajouter un point d'arrêt afin de provoquer l'ouverture de la fenêtre Débogueur :

1 Dans la fenêtre Script, ouvrez le script qui devrait contenir le point d'arrêt.

2 Cliquez sur la marge gauche de la fenêtre Script, à côté de la ligne de code dans laquelle le point d'arrêt doit apparaître ou placez un point d'insertion sur la ligne de code et cliquez sur Activer/désactiver le point d'arrêt. L'exécution du code sera interrompue au début de cette ligne et la fenêtre Script passera en mode de débogage. Si la fenêtre Script est ouverte lorsque Director détecte une erreur de script ou un point d'arrêt, la fenêtre Débogueur remplacera automatiquement la fenêtre Script.



Pour mettre fin au débogage, effectuez l'une des opérations suivantes :

- Cliquez sur le bouton Relancer le script dans la fenêtre Débogueur. Cette opération rétablit l'exécution normale du script.
- Cliquez sur le bouton Arrêter le débogage dans la fenêtre Débogueur. Cette opération met fin à la session de débogage et à l'animation.

  La fenêtre Script apparaît à nouveau, à la place de la fenêtre Débogueur.

Lorsque la fenêtre Débogueur apparaît, elle présente la ligne de code courante et vous propose plusieurs choix pour la suite de l'exécution.

Pour connaître la ligne de code courante :

- Dans le volet Script, recherchez la flèche verte affichée près d'une ligne de code.

  La flèche verte pointe vers la ligne courante. Vous ne pouvez pas sélectionner une autre ligne de code en cliquant dessus dans le volet Script.

Affichage de la pile d'appels dans la fenêtre Débogueur

Le volet Pile d'appels présente la séquence des gestionnaires imbriqués exécutés avant la ligne de code courante. Cette séquence est appelée « pile d'appels ». Utilisez la pile d'appels pour retracer la structure de votre code lors de la procédure de débogage. Vous pouvez visualiser les variables associées à un gestionnaire spécifique en cliquant sur le nom du gestionnaire dans le volet Pile d'appels. Les variables sont affichées dans le volet des variables.

Affichage des variables dans la fenêtre Débogueur

Le volet des variables de la fenêtre Débogueur affiche les variables associées au gestionnaire courant. Le gestionnaire courant est celui qui est affiché dans le volet Script et le dernier gestionnaire affiché dans le volet Pile d'appels. Vous pouvez également afficher les variables associées aux gestionnaires précédents dans la pile d'appels. Les modifications apportées aux valeurs des variables d'un script sont affichées en rouge. Pour plus d'informations sur la progression dans les scripts, veuillez consulter *Progression dans les scripts dans la fenêtre Débogueur*, page 109.

Pour afficher les variables associées à un gestionnaire dans la pile d'appels :

- Cliquez sur le nom du gestionnaire dans le volet Pile d'appels. Les variables sont affichées dans le volet des variables.

Le volet des variables contient quatre onglets vous permettant de visualiser les variables :

Le volet Toutes affiche les variables globales et locales associées au gestionnaire courant.

Le volet Locales affiche uniquement les variables locales associées au gestionnaire courant.

Le volet Propriétés affiche les propriétés déclarées dans le script courant.

Le volet Globales affiche uniquement les variables globales associées au gestionnaire courant.

Vous pouvez trier les variables dans le volet des variables :

- Pour trier les variables par nom, cliquez sur le mot *Nom* qui apparaît au-dessus des noms de variable.
- Pour trier les variables en ordre alphabétique inversé, cliquez une seconde fois sur le mot *Nom*.

Vous pouvez modifier les valeurs des variables locales du gestionnaire courant et des variables globales dans le volet des variables. Vous ne pouvez pas modifier les valeurs des variables locales qui ne sont pas situées dans le gestionnaire courant.

Pour changer la valeur d'une variable dans le volet des variables :

1. Double-cliquez sur la valeur de la variable dans la colonne Valeur.
2. Saisissez la nouvelle valeur de la variable.
3. Appuyez sur Entrée (Windows) ou sur Retour (Macintosh).

Affichage des objets dans la fenêtre Débogueur

Le volet Surveillance de la fenêtre Débogueur permet de visualiser les variables et autres objets associés au gestionnaire courant, ainsi que les objets associés aux autres gestionnaires. L'ajout d'objets dans le volet Surveillance vous permet de suivre leurs valeurs au fur et à mesure de leur modification grâce aux scripts. Lorsque la valeur d'un objet change en raison de l'exécution d'une ligne de code, Director affiche la couleur du nom de l'objet en rouge dans le volet Surveillance.

Le volet Surveillance affiche uniquement les objets que vous avez ajoutés. Vous pouvez utiliser chacun des quatre onglets du volet Surveillance pour organiser les objets en groupes.

Pour ajouter au volet Surveillance un objet dont le nom est affiché dans le volet Script :

1. Cliquez sur le nom de l'objet dans le volet Script.
2. Cliquez sur le bouton Surveiller l'expression.

Pour ajouter au volet Surveillance un objet dont le nom n'est pas affiché dans le volet Script :

1. Double-cliquez sur la première cellule vide de la colonne Nom du volet Surveillance.
2. Saisissez le nom de l'objet dans la cellule et appuyez sur Entrée (Windows) ou sur Retour (Macintosh).

   Si l'objet possède des propriétés, un signe plus (+) est affiché en regard du nom de l'objet.

Pour afficher les propriétés d'un objet :

- Cliquez sur le signe plus (+) à côté du nom de l'objet.

Le volet Surveillance permet d'organiser les objets de plusieurs façons.

Pour organiser les objets dans le volet Surveillance, effectuez l'une des opérations suivantes :

- Pour trier les objets dans le volet Surveillance, cliquez sur l'en-tête de colonne Nom affiché en haut de la colonne de gauche. Les noms d'objets de la colonne sont présentés par ordre alphabétique.
- Pour trier les objets en ordre alphabétique inversé, cliquez une seconde fois sur l'en-tête de colonne Nom.
- Pour organiser les objets en groupes, utilisez les onglets du volet Surveillance. Pour ajouter un objet à un volet spécifique, cliquez sur l'onglet de votre choix avant d'ajouter l'objet.
- Pour effacer le contenu d'un onglet dans le volet Surveillance, sélectionnez le volet, puis cliquez du bouton droit (Windows) ou en maintenant la touche Ctrl enfoncée (Macintosh) dans le volet Surveillance et sélectionnez Effacer tout.

Progression dans les scripts dans la fenêtre Débogueur

La fenêtre Débogueur fournit un ensemble d'outils permettant une exécution lente des scripts, ce qui vous permet de visualiser l'effet de chaque ligne de code dans votre animation. Vous pouvez exécuter une ligne de code à la fois et décider si vous souhaitez exécuter les gestionnaires ligne par ligne ou pour l'ensemble des lignes.

Pour exécuter uniquement la ligne de code courante indiquée par la flèche verte :

- Cliquez sur le bouton Exécuter le script pas à pas.

La plupart des gestionnaires comprennent des instructions d'appel des autres gestionnaires. Vous pouvez centrer votre attention sur ces gestionnaires imbriqués, ou les ignorer et vous limiter au code du gestionnaire courant.

Lorsque vous savez que les gestionnaires sont exécutés comme prévu et que vous souhaitez vous concentrer sur le code dans le gestionnaire courant, la fenêtre Débogueur peut ignorer les gestionnaires imbriqués et accéder directement à la prochaine ligne de code dans le gestionnaire courant. Lorsque le débogueur ignore un gestionnaire imbriqué, il exécute le gestionnaire, mais n'affiche pas le code du gestionnaire et ne marque pas de pause dans le gestionnaire imbriqué.

Pour ignorer les gestionnaires imbriqués :

- Cliquez sur le bouton Exécuter le script pas à pas dans la fenêtre Débogueur.

  Ce bouton exécute la ligne de code courante, ainsi que les gestionnaires imbriqués appelés par la ligne, puis s'arrête sur la ligne suivante du gestionnaire.

Si vous suspectez un dysfonctionnement des gestionnaires imbriqués et souhaitez examiner leur comportement, la fenêtre Débogueur vous permet également d'exécuter les gestionnaires imbriqués ligne par ligne.

Pour exécuter les gestionnaires imbriqués ligne par ligne :

- Cliquez sur le bouton Exécuter le script en détail dans la fenêtre Débogueur.

  Un clic sur le bouton Exécuter le script en détail lance l'exécution de la ligne de code courante et poursuit le flux normal dans les gestionnaires imbriqués appelés par l'intermédiaire de cette ligne. Une fois le traitement d'un gestionnaire imbriqué terminé, la fenêtre Débogueur s'arrête sur la prochaine ligne de code dans le gestionnaire de niveau supérieur.

Lorsque la procédure de débogage est terminée, vous pouvez quitter le Débogueur à tout moment :

Pour reprendre l'exécution normale de code et quitter la fenêtre Débogueur :

- Cliquez sur le bouton Relancer le script.

Pour quitter le Débogueur et arrêter la lecture de l'animation :

- Cliquez sur le bouton Arrêter le débogage.

Edition de scripts en mode de débogage

Lorsque vous êtes en mode de débogage, vous pouvez éditer vos scripts directement dans la fenêtre Débogueur. Ceci vous permet de corriger les erreurs dès que vous les rencontrez, puis de poursuivre la procédure de débogage.

Pour éditer un script dans la fenêtre Débogueur :

1 Cliquez dans le volet Script et placez le point d'insertion à l'endroit où vous souhaitez commencer à taper.

2 Apportez les modifications au script.

Vous pouvez passer directement à un gestionnaire spécifique en sélectionnant son nom et en cliquant sur le bouton Passer au gestionnaire.

3 Lorsque la procédure de débogage et d'édition des scripts est terminée, cliquez sur le bouton Arrêter le débogage. La fenêtre Script repasse en mode Script.

4 Cliquez sur le bouton Recompiler tous les scripts modifiés.

Débogage de projections et d'animations Shockwave

Cette section traite du débogage durant l'exécution dans les projections et les animations Director qui comprennent un contenu Macromedia Shockwave. Vous pouvez utiliser la fenêtre Messages ou activer les dialogues d'erreurs de script pour déboguer les projections et les animations Shockwave.

Pour déboguer à l'aide de la fenêtre Messages :

- Donnez à la propriété `debugPlaybackEnabled` de l'objet Lecteur la valeur `TRUE`.

  Lorsque cette propriété possède la valeur `TRUE`, la lecture d'une projection ou d'une animation Shockwave ouvre une fenêtre Messages (Microsoft Windows) ou un fichier texte Messages (Macintosh), et les résultats des appels de fonction `put()` ou `trace()` sont insérés dans ces formats.

  Si, à tout moment durant l'animation, vous donnez à la propriété `debugPlaybackEnabled` la valeur `FALSE`, la fenêtre ou le fichier texte Messages se ferme et ne peut être rouverte durant cette session de lecture, même si vous redonnez à la propriété `debugPlaybackEnabled` la valeur `TRUE` plus tard.

Pour déboguer en activant les dialogues d'erreurs de script :

- Dans le fichier .ini d'une projection ou d'une animation Shockwave, donnez à la propriété `DisplayFullLingoErrorText` la valeur 1.

  Vous créez ainsi, dans la boite de dialogue, un texte d'erreurs plus descriptif que le texte d'erreurs générique. Par exemple, un message d'erreur générique peut avoir l'aspect suivant :

  ```
  Erreur de script : Continuer ?
  ```

  L'attribution de la valeur 1 à la propriété the `DisplayFullLingoErrorText` pourrait générer le message d'erreur suivant :

  ```
  Erreur de script : liste attendue
  ```

  Pour obtenir des informations sur la création et la modification d'un fichier .ini pour une projection ou une animation Shockwave, consultez le fichier modèle .ini de Director se trouvant dans le dossier d'installation racine de Director.

Débogage avancé

Si le problème n'est pas facile à identifier, tentez les approches suivantes :

- Déterminez la section dans laquelle se situe le problème. Par exemple, si un clic sur un bouton ne produit pas le résultat escompté, vérifiez le script affecté à ce bouton.

  Si une image-objet exécute une action erronée, vérifiez les valeurs de propriété attachées à l'image-objet. Sont-elles définies sur les valeurs souhaitées ?
- Recherchez la séquence d'exécution du script. Lorsqu'une section de l'animation ne réagit pas comme vous l'espériez, tâchez tout d'abord de retracer la séquence des événements de l'animation. Consultez les autres scripts dans la hiérarchie des messages pour vous assurer que Director exécute le bon gestionnaire.
- Consultez les informations de suivi dans la fenêtre Messages, qui présentent les images parcourues par l'animation, ainsi que les gestionnaires appelés au cours de la lecture de l'animation.
- Essayez d'utiliser les fonctions Exécuter le script pas à pas et Exécuter le script en détail dans la fenêtre Débogueur et voyez si les résultats diffèrent de ce que vous attendiez.
- Vérifiez les variables et les expressions. Analysez le changement des valeurs lors de la lecture de l'animation. Observez si elles changent au mauvais moment ou si elles ne changent pas du tout. Si la même variable est utilisée dans plusieurs gestionnaires, assurez-vous que chaque gestionnaire qui utilise la variable a défini cette variable comme globale.

  Vous pouvez suivre les variables et les expressions en affichant leurs valeurs dans le volet Surveillance de la fenêtre Débogueur ou dans l'inspecteur d'objet.
- N'apportez qu'une modification à la fois. N'hésitez pas à apporter des modifications dans un gestionnaire pour vérifier si les changements peuvent résoudre le problème ou produire des résultats qui aident à le localiser.

  Veillez toutefois à ne pas résoudre un problème en en créant un autre. Apportez une modification à la fois et annulez-la si le problème n'est pas résolu. Si vous apportez trop de modifications avant de résoudre un problème, vous risquez de ne plus pouvoir déterminer quel était le problème initial, voire même d'en créer de nouveaux.
- Recréez la section. Si vous ne trouvez pas de solution, tâchez de recréer la section depuis le début. Par exemple, si une image-objet ne réagit pas correctement lorsque le pointeur la survole, créez une simple animation contenant uniquement cette image-objet et le gestionnaire, avec la méthode `rollOver()`.

  Si vous copiez/collez simplement les scripts, cela risque de copier le problème. En revanche, si vous recréez la section, vous serez amené à reconstruire la logique depuis son premier niveau, et vous pourrez alors vérifier si Director réagit comme vous le souhaitez. Si la section que vous avez recréée ne fonctionne toujours pas comme prévu, il se peut que l'erreur provienne de la logique de la section.

  Si la section que vous avez recréée fonctionne correctement, comparez-la avec l'animation d'origine pour noter leurs différences. Vous pouvez également copier la section dans l'originale et vérifier si le problème est résolu.

CHAPITRE 5
Objets principaux de Director

Les objets principaux de Macromedia Director MX 2004 donnent accès aux fonctionnalités et options disponibles dans Director, les projections et Macromedia Shockwave Player. Les objets principaux incluent le moteur du lecteur de Director, les fenêtres des animations, les images-objets, les sons, etc. Ils représentent la couche de base à travers laquelle on accède à tous les API et autres catégories d'objets, à l'exception des objets de scripting qui étendent les fonctionnalités de base de Director.

Pour voir comment les objets principaux sont liés entre eux et à d'autres objets de Director, consultez *Diagramme de modèles d'objets*, page 59.

Acteur

Représente un acteur au sein d'une bibliothèque de distribution. Les acteurs sont les médias et les éléments de script d'une animation. Les acteurs média peuvent être du texte, des bitmaps, des formes, etc. Les acteurs script incluent les comportements, les scripts d'animation, etc.

Un acteur peut être référencé soit par numéro soit par nom.

- Lorsque vous faites référence à un acteur en utilisant son numéro, Director le recherche dans une bibliothèque précise et en extrait les données. Cette méthode est plus rapide que celle qui consiste à faire référence à l'acteur par son nom. Toutefois, vu que Director ne met pas à jour automatiquement les références aux numéros d'acteurs dans le script, toute référence par numéro à un acteur qui a changé de position dans sa bibliothèque de distribution sera rompue.
- Lorsque vous faites référence à un acteur en utilisant son nom, Director effectue des recherches dans toutes les bibiliothèques de distribution d'une animation, de la première à la dernière, et extrait les données de l'acteur lorsqu'il trouve son nom. Cete méthode est plus lente que celle qui consiste à faire référence au numéro de l'acteur, surtout lorsqu'il s'agit d'animations contenant plusieurs bibliothèques de distribution et acteurs. Toutefois, une référence à un nom d'acteur permet à cette dernière de rester intacte, même si l'acteur change de position dans sa bibliothèque de distribution.

Vous pouvez créer une référence à une bibliothèque de distribution en utilisant la fonction de haut niveau `member()` ou la propriété `member` de l'objet Distribution, Animation ou Image-objet.

Les exemples suivants illustrent la création d'une référence à un acteur.

- Utilisez la fonction de haut niveau `member()`.

```
-- Syntaxe Lingo
objTree = member("arbreBmp")

// Syntaxe JavaScript
var objTree = member("arbreBmp");
```

- Utilisez la propriété `member` de l'objet Image-objet.

```
-- Syntaxe Lingo
objTree = sprite(1).member;

// Syntaxe JavaScript
var objTree = sprite(1).member;
```

Résumé des méthodes pour l'objet Acteur

Méthode
copyToClipBoard()
duplicate() (acteur)
erase()
importFileInto()
move()
pasteClipBoardInto()
preLoad() (acteur)
unLoad() (acteur)

Résumé des propriétés pour l'objet Acteur

Propriété	
castLibNum	modifiedDate
comments	name
creationDate	number (acteur)
fileName (acteur)	purgePriority
height	rect (acteur)
hilite	regPoint
linked	scriptText
loaded	size
media	thumbNail
mediaReady	type (acteur)
modified	width
modifiedBy	

Voir aussi

Types de médias, member(), member (distribution), member (animation), member (image-objet), Animation, Lecteur, Objets de scripting, Image-objet, Fenêtre

Animation

Représente une animation en cours d'exécution dans le lecteur de Director.

Le lecteur de Director peut contenir une ou plusieurs animations. Une animation peut contenir une ou plusieurs bibliothèques. Une bibliothèque de distribution peut consister en un ou plusieurs acteurs qui représentent des médias et des scripts dans une animation. Les acteurs média peuvent être du texte, des bitmaps, des formes, etc. Les acteurs script incluent les comportements, les scripts d'animation, etc. Les images-objets sont créées à partir d'acteurs et utilisées sur la scène d'une animation.

Vous pouvez désigner l'animation en cours d'exécution en utilisant la propriété de haut niveau `_movie`. Vous pouvez désigner une animation quelconque du lecteur en utilisant la propriété `movie` de l'objet Fenêtre.

- Désignez l'animation en cours d'exécution.

```
-- Syntaxe Lingo
objAnimation = _movie

// Syntaxe JavaScript
var objAnimation = _movie;
```

- Utilisez la propriété `movie` de l'objet Fenêtre pour accéder à l'animation d'une fenêtre donnée.

```
-- Syntaxe Lingo
objAnimation = _player.window[2].movie

// Syntaxe JavaScript
var objAnimation = _player.window[2].movie;
```

Dans Director MX 2004, vous pouvez non seulement utiliser une référence à une animation pour accéder aux méthodes et propriétés de l'animation même, mais aussi appeler des gestionnaires Lingo et JavaScript et accéder aux acteurs et images-objets de l'animation, y compris leurs méthodes et propriétés. Cette procédure est différente des versions précédentes de Director dans lesquelles vous deviez utiliser la commande `tell` pour accéder aux animations. L'objet Animation constitue une manière simple de travailler sur les animations.

Remarque : La commande `tell` est obsolète dans Director MX 2004.

Résumé des méthodes pour l'objet Animation

Méthode	
beginRecording()	newMember()
cancelIdleLoad()	preLoad() (animation)
clearFrame()	preLoadMember()
constrainH()	preLoadMovie()
constrainV()	printFrom()
delay()	puppetPalette()
deleteFrame()	puppetSprite()

Méthode (suite)	
duplicateFrame()	puppetTempo()
endRecording()	puppetTransition()
finishIdleLoad()	ramNeeded()
frameReady() (animation)	rollOver()
go()	saveMovie()
goLoop()	sendAllSprites()
goNext()	sendSprite()
goPrevious()	stopEvent()
idleLoadDone()	unLoad() (animation)
insertFrame()	unLoadMember()
label()	unLoadMovie()
marker()	updateFrame()
mergeDisplayTemplate()	updateStage()
beginRecording()	newMember()

Résumé des propriétés pour l'objet Animation

Propriété	
aboutInfo	frameTransition
active3dRenderer	idleHandlerPeriod
actorList	idleLoadMode
allowCustomCaching	idleLoadPeriod
allowGraphicMenu	idleLoadTag
allowSaveLocal	idleReadChunkSize
allowTransportControl	imageCompression
allowVolumeControl	imageQuality
allowZooming	keyboardFocusSprite
beepOn	lastChannel
buttonStyle	lastFrame
castLib	markerList
centerStage	member (animation)
copyrightInfo (animation)	name
displayTemplate	paletteMapping
dockingEnabled	path (animation)
editShortCutsEnabled	preferred3dRenderer
enableFlashLingo	preLoadEventAbort

Propriété (suite)	
exitLock	score
fileFreeSize	scoreSelection
fileSize	script
fileVersion	sprite (animation)
fixStageSize	stage
frame	timeoutList
frameLabel	traceLoad
framePalette	traceLogFile
frameScript	traceScript
frameSound1	updateLock
frameSound2	useFastQuads
frameTempo	xtraList (animation)

Voir aussi

_movie, Bibliothèque de distribution, Acteur, movie, Lecteur, Image-objet, Fenêtre

Bibliothèque de distribution

Représente une seule bibliothèque de distribution dans une animation.

Une animation peut contenir une ou plusieurs bibliothèques. Une bibliothèque de distribution peut consister en un ou plusieurs acteurs qui représentent des médias dans une animation, tels que les sons, textes, graphiques et autres.

Vous pouvez créer une référence à une bibliothèque de distribution en utilisant la fonction de haut niveau `castLib()` ou la propriété `castLib` de l'objet Animation. Par exemple, si une animation contient une bibliothèque de distribution appelée `scripts`, vous pouvez créer une référence à cette bibliothèque en procédant comme suit :

- Utilisez la méthode de haut niveau `castLib()`.

```
-- Syntaxe Lingo
scriptBibli = castLib("scripts")

// Syntaxe JavaScript
var scriptBibli = castLib("scripts");
```

- Utilisez la propriété `castLib` de l'objet Animation.

```
-- Syntaxe Lingo
scriptBibli = _movie.castLib["scripts"]

// Syntaxe JavaScript
var scriptBibli = _movie.castLib["scripts"];
```

Résumé des méthodes pour l'objet Bibliothèque de distribution

Méthode
findEmpty()

Résumé des propriétés pour l'objet Bibliothèque de distribution

Propriété
fileName (distribution)
member (distribution)
name
number (distribution)
preLoadMode
selection

Voir aussi

castLib, castLib(), Acteur, Animation, Lecteur, Image-objet, Fenêtre

Fenêtre

Représente une fenêtre dans laquelle une animation est en cours d'exécution, dont la fenêtre Scène et toute autre animation dans une fenêtre (MIAW) couramment utilisée.

Vous pouvez créer une référence à un objet Fenêtre en utilisant la fonction de haut niveau window(), la propriété window de l'objet Lecteur ou la propriété windowList de l'objet Lecteur.

- Utilisez la méthode de haut niveau window().

```
-- Syntaxe Lingo
objFenêtre = window("Soleil")

// Syntaxe JavaScript
var objFenêtre = window("Soleil");
```

- Utilisez la propriété window de l'objet Lecteur.

```
-- Syntaxe Lingo
objFenêtre = _player.window["Soleil"]

// Syntaxe JavaScript
var objFenêtre = _player.window["Soleil"];
```

- Utilisez la propriété windowList de l'objet Lecteur.

```
-- Syntaxe Lingo
objFenêtre = _player.windowList[1]

// Syntaxe JavaScript
var objFenêtre = _player.windowList[1];
```

Remarque : Lorsque vous créez une référence au nom d'une fenêtre en utilisant soit la fonction de haut niveau window() soit la propriété window de l'objet Lecteur, cette référence n'est créée que si une fenêtre portant ce nom existe. Si une fenêtre de ce nom n'existe pas, la référence contient VOID (Lingo) ou null (syntaxe JavaScript).

Résumé des méthodes pour l'objet Fenêtre

Méthode	
close()	moveToBack()
forget() (fenêtre)	moveToFront()
maximize()	open() (fenêtre)
mergeProps()	restore()
minimize()	

Résumé des propriétés pour l'objet Fenêtre

Propriété	
appearanceOptions	resizable
bgColor (fenêtre)	sizeState
dockingEnabled	sourceRect
drawRect	title (fenêtre)
fileName (fenêtre)	titlebarOptions
image (fenêtre)	type (fenêtre)
movie	visible
name	windowBehind
picture (fenêtre)	windowInFront
rect (fenêtre)	

Voir aussi

Bibliothèque de distribution, Acteur, Animation, Lecteur, Image-objet, window(), window, windowList

Global

Fournit un emplacement de stockage de variables globales. Ces variables sont disponibles dans Lingo et la syntaxe JavaScript.

Vous pouvez accéder à l'objet Global en utilisant la propriété de haut niveau `_global`. Vous pouvez soit affecter `_global` à une variable soit utiliser directement la propriété `_global` pour accéder aux méthodes de l'objet Global et à toute variable globale définie.

- Affectez `_global` à une variable.

```
-- Syntaxe Lingo
objGlobal = _global

// Syntaxe JavaScript
var objGlobal = _global;
```

- Utilisez la propriété `_global` directement.

```
-- Syntaxe Lingo
_global.showGlobals()

// Syntaxe JavaScript
_global.showGlobals();
```

- Accédez à une variable globale.

```
-- Syntaxe Lingo
global gSuccess
...
on mouseDown
  gSuccess = "Félicitations !"
  put(_global.gSuccess) -- affiche  "Félicitations!"
end"

// Syntaxe JavaScript
_global.gSuccess = "Félicitations !";
...
function mouseDown() {
  trace(_global.gSuccess); // affiche  "Félicitations !"
}
```

Résumé des méthodes pour l'objet Global

Méthode
`clearGlobals()`
`showGlobals()`

Voir aussi

`_global`

Image-objet

Représente une occurrence d'un acteur dans une piste d'image-objet du scénario.

Un objet Image-objet couvre une plage d'images-objets, c'est-à-dire la gamme d'images d'une piste d'image-objet donnée. Un objet Piste d'image-objet représente une piste d'image-objet toute entière, quel que soit le nombre d'images-objets qu'elle contient.

Une image-objet peut être référencée soit par numéro soit par nom.

- Lorsque vous faites référence à une image-objet par numéro, Director effectue une recherche dans toutes les images-objets qui existent dans l'image courante du scénario, en commençant par la piste dont le numéro est le plus bas, et extrait les données de l'image-objet lorsqu'il la trouve. Cette méthode est plus rapide que celle qui consiste à faire référence à une image-objet par son nom. Toutefois, vu que Director ne met pas à jour automatiquement les références aux numéros d'images-objets dans le script, toute référence par numéro à une image-objet qui a changé de position dans la scène sera rompue.

- Lorsque vous désignez une image-objet par son nom, Director fait une recherche dans toutes les images-objets qui existent dans l'image courante du scénario, en commençant par la piste portant le numéro le plus bas, et extrait les données de l'image-objet lorsqu'il la trouve. Cete méthode est plus lente que celle qui consiste à faire référence au numéro de l'image-objet, surtout lorsqu'il s'agit d'animations contenant plusieurs bibliothèques de distribution, acteurs et images-objets. Toutefois, une référence à un nom d'image-objet permet à cette référence de rester intacte, même si l'image-objet change de position sur la scène.

Vous pouvez créer une référence à un objet Image-objet en utilisant la fonction de haut niveau `sprite()`, la propriété `sprite` de l'objet Animation ou la propriété `sprite` de l'objet Piste d'image-objet.

- Utilisez la fonction de haut niveau `sprite()`.

```
-- Syntaxe Lingo
objImageObjet = sprite(1)

// Syntaxe JavaScript
var objImageObjet = sprite(1);
```

- Utilisez la propriété `sprite` de l'objet Animation.

```
-- Syntaxe Lingo
objImageObjet = _movie.sprite["saule"]

// Syntaxe JavaScript
var objImageObjet = _movie.sprite["saule"];
```

- Utilisez la propriété `sprite` de l'objet Piste d'image-objet.

```
-- Syntaxe Lingo
objImageObjet = channel(3).sprite

// Syntaxe JavaScript
var objImageObjet = channel(3).sprite;
```

Vous pouvez utiliser une référence à un objet Image-objet pour accéder à l'acteur à partir duquel l'image-objet a été créée. Toute modification effectuée sur l'acteur à partir duquel une image-objet a été créée est également reflétée dans l'image-objet. L'exemple suivant illustre la modification du texte d'un acteur texte à partir duquel l'image-objet 5 a été créée. Ce changement apporté à l'acteur sera également reflété dans l'image-objet 5.

```
-- Syntaxe Lingo
labelText = sprite(5)
labelText.member.text = "Saule pleureur"

// Syntaxe JavaScript
var labelText = sprite(5);
labelText.member.text = "Saule pleureur";
```

Résumé des propriétés pour l'objet Image-objet

Propriété	
backColor	locV
blend (image-objet)	locZ
bottom	member (image-objet)
constraint	name (image-objet)
cursor	quad
editable	rect (image-objet)
endFrame	right
flipH	rotation
flipV	skew
foreColor	spriteNum
height	startFrame
ink	top
left	width
locH	

Voir aussi

Bibliothèque de distribution, Acteur, Animation, Lecteur, sprite (animation), sprite (piste d'image-objet), sprite(), Piste d'image-objet, Fenêtre

Lecteur

Représente le moteur de lecture principal utilisé pour gérer et exécuter l'environnement auteur, les animations dans une fenêtre (MIAW), les projections et Shockwave Player.

L'objet Lecteur donne accès à toutes les animations et fenêtres qu'il gère, en plus des Xtras disponibles.

Vous pouvez créer une référence à l'objet Lecteur en utilisant la propriété de haut niveau `_player`.

- Affectez `_player` à une variable.

```
-- Syntaxe Lingo
objLecteur = _player

// Syntaxe JavaScript
var objLecteur = _player;
```

- Utilisez la propriété `_player` directement.

```
-- Syntaxe Lingo
_player.alert("L'animation est terminée.")

// Syntaxe JavaScript
_player.alert("L'animation est terminée.");
```

Résumé des méthodes pour l'objet Lecteur

Méthode	
alert()	getPref()
appMinimize()	halt()
cursor()	open() (lecteur)
externalParamName()	quit()
externalParamValue()	setPref()
flushInputEvents()	windowPresent()

Résumé des propriétés pour l'objet Lecteur

Propriété	
activeCastLib	netThrottleTicks
activeWindow	organizationName
alertHook	productName
applicationName	productVersion
applicationPath	safePlayer
currentSpriteNum	scriptingXtraList
debugPlaybackEnabled	searchCurrentFolder
digitalVideoTimeScale	searchPathList
disableImagingTransformation	serialNumber
emulateMultibuttonMouse	sound (lecteur)
externalParamCount	switchColorDepth
frontWindow	toolXtraList
inlineImeEnabled	transitionXtraList
lastClick	userName
lastEvent	window
lastKey	window
lastRoll	xtra
mediaXtraList	xtraList (lecteur)
netPresent	

Voir aussi

_player, Bibliothèque de distribution, Acteur, Animation, Image-objet, Fenêtre

Piste audio

Représente une piste audio individuelle au sein de l'objet Son.

Les pistes audio disponibles sont au nombre de huit. Vous pouvez utiliser un objet Piste audio dans un script pour accéder à l'une des huit pistes audio et la modifier.

Remarque : Vous ne pouvez modifier que les deux premières pistes audio dans le scénario de l'interface utilisateur de Director.

Vous pouvez créer une référence à un objet Piste audio en utilisant la méthode de haut niveau `sound()`, la propriété `sound` de l'objet Lecteur ou la méthode `channel()` de l'objet Son. Par exemple, vous pouvez faire référence à la piste audio 2 comme suit :

- Utilisez la méthode de haut niveau `sound()`.

```
-- Syntaxe Lingo
objPisteAudio = sound(2)

// Syntaxe JavaScript
var objPisteAudio = sound(2);
```

- Utilisez la propriété `sound` de l'objet Lecteur.

```
-- Syntaxe Lingo
objPisteAudio = _player.sound[2]

// Syntaxe JavaScript
var objPisteAudio = _player.sound[2];
```

- Utilisez la méthode `channel()` de l'objet Son.

```
-- Syntaxe Lingo
objPisteAudio = _sound.channel(2)

// Syntaxe JavaScript
var objPisteAudio = _sound.channel(2);
```

Résumé des méthodes pour l'objet Piste audio

Méthode	
`breakLoop()`	`playFile()`
`fadeIn()`	`playNext() (piste audio)`
`fadeOut()`	`queue()`
`fadeTo()`	`rewind() (piste audio)`
`getPlayList()`	`setPlayList()`
`pause() (piste audio)`	`stop() (piste audio)`
`play() (piste audio)`	

Résumé des propriétés pour l'objet Piste audio

Propriété	
`channelCount`	`member (piste audio)`
`elapsedTime`	`pan`

Propriété (suite)	
endTime	sampleCount
loopCount	sampleRate
loopEndTime	startTime
loopsRemaining	status
loopStartTime	volume (piste audio)

Voir aussi

channel() (audio), sound (lecteur), sound(), Son

Piste d'image-objet

Représente une piste d'image-objet individuelle dans le scénario.

Un objet Image-objet couvre une plage d'images-objets, c'est-à-dire la gamme d'images d'une piste d'image-objet donnée. Un objet Piste d'image-objet représente une piste d'images-objets toute entière, quel que soit le nombre d'images-objets qu'elle contient.

Les pistes d'images-objets sont contrôlées par le scénario par défaut. Utilisez l'objet Piste d'image-objet pour faire passer le contrôle d'une piste d'image-objet au script lors d'une séance d'enregistrement de scénario.

Une piste d'image-objet peut être référencée soit par numéro soit par nom.

- Lorsque vous désignez une piste d'image-objet par son numéro, vous accédez directement à la piste. Cette méthode est plus rapide que celle qui consiste à faire référence à une piste d'image-objet par son nom. Toutefois, vu que Director ne met pas à jour automatiquement les références aux numéros des piste d'images-objets dans le script, toute référence par numéro à une piste d'image-objet qui a changé de position dans la scène sera rompue.
- Lorsque vous désignez une piste d'image-objet par son nom, Director fait une recherche dans toutes les pistes, en commençant par la piste portant le numéro le plus bas, et extrait les données de la piste d'image-objet lorsqu'il la trouve. Cete méthode est plus lente que celle qui consiste à faire référence au numéro de la piste d'image-objet, surtout lorsqu'il s'agit d'animations contenant plusieurs bibliothèques de distribution, acteurs et images-objets. Toutefois, une référence à une piste d'image-objet permet à cette référence de rester intacte, même si la piste d'image-objet change de position dans le scénario.

Vous pouvez créer une référence à une piste d'image-objet en utilisant la méthode de haut niveau `channel()`.

- Utilisez la méthode de haut niveau `channel()`.

```
-- Syntaxe Lingo
objPisteImageObjet = channel(2)

// Syntaxe JavaScript
var objPisteImageObjet = channel(2);
```

Vous pouvez utiliser une référence à un objet Piste d'image-objet pour accéder à l'image-objet couramment utilisée dans une piste d'image-objet particulière. L'exemple suivant illustre l'accès à la couleur de fond de l'image-objet couramment utilisée dans la piste d'image-objet 2.

```
-- Syntaxe Lingo
libelléImageObjet = channel(2).sprite.backColor

// Syntaxe JavaScript
var libelléImageObjet = channel(2).sprite.backColor;
```

Résumé des méthodes pour l'objet Piste d'image-objet

Méthode
makeScriptedSprite()
removeScriptedSprite()

Résumé des propriétés pour l'objet Piste d'image-objet

Propriété
name (piste d'image-objet)
number (piste d'image-objet)
scripted
sprite (piste d'image-objet)

Voir aussi

Bibliothèque de distribution, channel() (niveau supérieur), Acteur, Animation, Lecteur, Image-objet, Fenêtre

Son

Contrôle la lecture audio dans les huit pistes audio disponibles.

L'objet Son consiste en huit objets Piste audio représentant des pistes audio individuelles.

Vous pouvez créer une référence à l'objet Son en utilisant la propriété de haut niveau `_sound`.

- Affectez `_sound` à une variable.

```
-- Syntaxe Lingo
objSon = _sound

// Syntaxe JavaScript
var objSon = _sound;
```

- Utilisez la propriété `_sound` pour accéder à la propriété `soundDevice` de l'objet Son.

```
-- Syntaxe Lingo
objPériph = _sound.soundDevice

// Syntaxe JavaScript
var objPériph = _sound.soundDevice;
```

Résumé des méthodes pour l'objet Son

Méthode
beep()
channel() (audio)

Résumé des propriétés pour l'objet Son

Propriété
soundDevice
soundDeviceList
soundEnabled
soundKeepDevice
soundLevel
soundMixMedia

Voir aussi

_sound, Piste audio

Souris

Permet d'accéder à l'activité d'un utilisateur liée à la souris, y compris les mouvements et les clics de souris.

Vous pouvez accéder à l'objet Souris en utilisant la propriété de haut niveau _mouse. Vous pouvez soit affecter _mouse à une variable soit utiliser directement la propriété _mouse pour accéder aux propriétés de l'objet Souris.

- Affectez _mouse à une variable.

```
-- Syntaxe Lingo
objSouris = _mouse

// Syntaxe JavaScript
var objSouris = _mouse;
```

- Utilisez la propriété _mouse directement.

```
-- Syntaxe Lingo
doubleClic = _mouse.doubleClick

// Syntaxe JavaScript
var doubleClic = _mouse.doubleClick;
```

Résumé des propriétés pour l'objet Souris

Propriété	
clickLoc	mouseLoc
clickOn	mouseMember
doubleClick	mouseUp

Propriété (suite)	
mouseChar	mouseV
mouseDown	mouseWord
mouseH	rightMouseDown
mouseItem	rightMouseUp
mouseLine	stillDown

Voir aussi

_mouse

Système

Donne accès aux informations sur le système et l'environnement, à commencer par les méthodes de niveau du système.

Vous pouvez créer une référence à l'objet Système en utilisant la propriété de haut niveau _system.

- Affectez _system à une variable.

```
-- Syntaxe Lingo
objSystème = _system

// Syntaxe JavaScript
var objSystème = _system;
```

- Utilisez la propriété _system directement.

```
-- Syntaxe Lingo
dateSystème = _system.date()

// Syntaxe JavaScript
var dateSystème = _system.date();
```

Résumé des méthodes pour l'objet Système

Méthode
date() (Système)
restart()
shutDown()
time() (Système)

Résumé des propriétés pour l'objet Système

Propriété
colorDepth
deskTopRectList
environmentPropList
milliseconds

Voir aussi

_system

Touche

Utilisé pour contrôler l'activité d'un utilisateur au clavier.

Vous pouvez accéder à l'objet Touche en utilisant la propriété de haut niveau _key. Vous pouvez soit affecter _key à une variable soit utiliser directement la propriété _key pour accéder aux méthodes et propriétés de l'objet Touche.

- Affectez _key à une variable.

```
-- Syntaxe Lingo
objTouche = _key

// Syntaxe JavaScript
var objTouche = _key;
```

- Utilisez directement la propriété _key.

```
-- Syntaxe Lingo
ctrlEnfoncée = _key.controlDown

// Syntaxe JavaScript
var ctrlEnfoncée = _key.controlDown;
```

Résumé des méthodes pour l'objet Touche

Méthode
keyPressed()

Résumé des propriétés pour l'objet Touche

Propriété
commandDown
controlDown
key
keyCode
optionDown
shiftDown

Voir aussi

_key

CHAPITRE 6
Types de médias

Les types de médias de Director MX 2004 de Macromedia donnent accès aux fonctionnalités des divers types de médias (RealMedia, DVD, GIF animé, etc.), qui sont ajoutés aux animations en tant qu'acteurs.

Les médias ne sont pas réellement des objets mais plutôt des acteurs qui se rapportent à un type de média précis. Lorsqu'un type de média est ajouté à une animation en tant qu'acteur, il hérite de la fonctionnalité de l'objet Acteur principal et étend l'objet Acteur en fournissant des fonctionnalités supplémentaires qui ne sont disponibles que pour le type de média spécifié. Par exemple, un acteur RealMedia a accès aux méthodes et propriétés de l'objet Acteur, et possède également d'autres méthodes et propriétés propres à RealMedia. Les autres types de médias affichent tous ce comportement.

Pour voir comment les types de médias d'acteurs sont liés entre eux et à d'autres objets de Director, consultez *Diagramme de modèles d'objets*, page 59.

Animation Flash

Représente un acteur ou une image-objet renfermant un contenu Flash.

Vous pouvez ajouter un acteur d'animation Flash à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Syntaxe Lingo
_movie.newMember(#flash)

// Syntaxe JavaScript
_movie.newMember("flash");
```

Un acteur ou une image-objet Flash peut également contenir des composants Flash. Les composants Flash fournissent des fonctionnalités qui étendent les fonctionnalités existantes des acteurs ou des images-objets Flash. Pour plus d'informations sur les composants Flash pris en charge par Director, consultez *Composant Flash*, page 136.

Certaines des méthodes ou propriétés suivantes s'appliquent uniquement aux images-objets créées à partir d'un acteur d'animation Flash.

Résumé des méthodes pour le type de média Animation Flash

Méthode	
callFrame()	printAsBitmap()
clearAsObjects()	rewind() (GIF animé, Flash)
clearError()	setCallback()
findLabel()	setFlashProperty()
flashToStage()	settingsPanel()
getFlashProperty()	setVariable()
getVariable()	showProps()
goToFrame()	stageToFlash()
hitTest()	stop() (Flash)
hold()	stream()
newObject()	tellTarget()
print()	

Résumé des propriétés pour le type de média Animation Flash

Propriété	
actionsEnabled	originPoint
broadcastProps	originV
bufferSize	playBackMode
buttonsEnabled	playing
bytesStreamed	posterFrame
centerRegPoint	quality
clickMode	rotation
defaultRect	scale (acteur)
defaultRectMode	scaleMode
eventPassMode	sound (lecteur)
fixedRate	static
flashRect	streamMode
frameCount	streamSize
imageEnabled	viewH
linked	viewPoint
mouseOverButton	viewScale
originH	viewV
originMode	

Voir aussi

Composant Flash, Acteur

Animation liée

Représente un acteur d'animation liée.

Vous pouvez ajouter un acteur d'animation liée à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Syntaxe Lingo
_movie.newMember(#movie)

// Syntaxe JavaScript
_movie.newMember("movie");
```

Résumé des propriétés pour le type de média Animation liée

Propriété
scriptsEnabled

Voir aussi

Acteur

Bitmap

Représente un acteur bitmap.

Utilisez les objets image bitmaps pour effectuer de simples opérations affectant le contenu d'un acteur bitmap telles que le changement des couleurs de fond et de premier plan de l'acteur ou pour effectuer une délicate manipulation des pixels d'une image telle que le recadrage, le dessin et la copie de pixels.

Vous pouvez ajouter un acteur bitmap à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Syntaxe Lingo
_movie.newMember(#bitmap)

// Syntaxe JavaScript
_movie.newMember("bitmap");
```

Certaines des méthodes ou propriétés suivantes s'appliquent uniquement aux images-objets créées à partir d'un acteur bitmap.

Résumé des méthodes pour le type de média Bitmap

Méthode
crop() (Image)
pictureP()

Résumé des propriétés pour le type de média Bitmap

Propriété	
alphaThreshold	imageCompression
backColor	imageQuality
blend (image-objet)	palette
depth (Bitmap)	picture (acteur)
dither	rect (image)
foreColor	trimWhiteSpace
image (image)	useAlpha

Voir aussi

Acteur

Boucle d'animation

Représente un acteur boucle d'animation.

Vous pouvez ajouter un acteur boucle d'animation à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Syntaxe Lingo
_movie.newMember(#filmloop)

// Syntaxe JavaScript
_movie.newMember("filmloop");
```

Résumé des propriétés pour le type de média Boucle d'animation

Propriété
media
regPoint

Voir aussi

Acteur

Bouton

Représente un acteur bouton ou case à cocher.

Vous pouvez ajouter un acteur bouton à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Syntaxe Lingo
_movie.newMember(#button)

// Syntaxe JavaScript
_movie.newMember("button");
```

Résumé des propriétés pour le type de média Bouton

Propriété
hilite

Voir aussi

Acteur

Champ

Représente un acteur champ.

Vous pouvez ajouter un acteur champ à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Syntaxe Lingo
_movie.newMember(#field)

// Syntaxe JavaScript
_movie.newMember("field");
```

Résumé des méthodes pour le type de média Champ

Méthode	
charPosToLoc()	pointToItem()
lineHeight()	pointToLine()
linePosToLocV()	pointToParagraph()
locToCharPos()	pointToWord()
locVToLinePos()	scrollByLine()
pointToChar()	scrollByPage()

Résumé des propriétés pour le type de média Champ

Propriété	
alignment	fontStyle
autoTab	lineCount
border	margin
boxDropShadow	pageHeight
boxType	scrollTop
dropShadow	selEnd
editable	selStart
font	text
fontSize	wordWrap

Voir aussi

Acteur

Composant Flash

Représente un composant Macromedia Flash imbriqué dans un acteur ou une image-objet comprenant un contenu Flash.

Un composant Flash offre des fonctionnalités qui étendent les fonctionnalités existantes des acteurs ou des images-objets comprenant du contenu Flash. Ils sont créés et supportés entièrement par la communauté de développement de Director.

Director prend en charge les composants Flash suivants :

Composant Flash	Description
Button	Bouton d'interface utilisateur rectangulaire redimensionnable.
CheckBox	Partie importante de toute forme d'application ; vous pouvez l'utiliser à chaque fois que vous avez besoin de rassembler un ensemble de valeurs `true` ou `false` qui ne sont pas mutuellement exclusives.
DateChooser	Calendrier permettant à un utilisateur de choisir une date.
label	Une ligne de texte.
List	Liste défilante à un ou plusieurs choix.
NumericStepper	Permet à un utilisateur de parcourir un ensemble de numéros ordonnés.
RadioButton	Composante essentielle de toute forme d'application web : vous pouvez l'utiliser à chaque fois que vous voulez faire un choix parmi un groupe d'options.
ScrollPane	Affiche des clips, des fichiers JPEG et des fichiers SWF dans une zone à défilement.
TextArea	Champ de texte à plusieurs lignes.
TextInput	Composant à une seule ligne qui entoure l'objet TextField natif d'ActionScript.
Tree	Permet à un utilisateur de visualiser des données hiérarchiques.

Un composant Flash a accès aux mêmes API que ceux auxquels accède un acteur ou une image-objet Flash normal, en plus des fonctionnalités se rapportant à ce composant. Pour plus d'informations sur l'utilisation de ces composants Flash, consultez les rubriques Utilisation de Director dans le panneau d'aide de Director.

Vous pouvez ajouter un acteur de composant Flash à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Syntaxe Lingo
_movie.newMember(#flashcomponent)

// Syntaxe JavaScript
_movie.newMember("flashcomponent");
```

Voir aussi

Animation Flash, Acteur

Curseur

Représente un acteur curseur.

Vous pouvez ajouter un acteur curseur à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Syntaxe Lingo
_movie.newMember(#cursor)

// Syntaxe JavaScript
_movie.newMember("cursor");
```

Résumé des propriétés pour le type de média Curseur

Propriété
`castMemberList`
`cursorSize`
`hotSpot`
`interval`

Voir aussi

`Acteur`

DVD

Représente un acteur DVD.

Vous pouvez ajouter un acteur DVD à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Syntaxe Lingo
_movie.newMember(#dvd)

// Syntaxe JavaScript
_movie.newMember("dvd");
```

Certaines des méthodes ou propriétés suivantes s'appliquent uniquement aux images-objets créées à partir d'un acteur DVD.

Résumé des événements pour le type de média DVD

Les événements DVD suivants sont toujours pris en charge par un gestionnaire d'événement `DVDeventNotification`. Lorsque l'un de ces événements se produit, le gestionnaire d'événement `DVDeventNotification` le reçoit sous forme de paramètre. Certains de ces événements contiennent également d'autres informations qui sont transmises sous forme de deuxième ou troisième paramètre à `DVDeventNotification`. Pour plus d'informations sur l'utilisation des événements suivants avec le gestionnaire `DVDeventNotification`, consultez *on DVDeventNotification*, page 179.

Evénement	
`angleChange`	`noFirstPlayChain`
`audioStreamChange`	`parentalLevelChange`

Evénement (suite)	
buttonChange	playbackStopped
chapterAutoStop	playPeriodAutoStop
chapterStart	rateChange
diskEjected	stillOff
diskInserted	stillOn
domainChange	titleChange
error	UOPchange
karaokeMode	warning

Résumé des méthodes pour le type de média DVD

Méthode	
activateAtLoc()	selectAtLoc()
activateButton()	selectButton()
frameStep()	selectButtonRelative()
pause() (DVD)	stop() (DVD)
play() (DVD)	subPictureType()
returnToTitle()	titleMenu()
rootMenu()	

Résumé des propriétés pour le type de média DVD

Propriété	
angle (DVD)	duration (DVD)
angleCount	folder
aspectRatio	frameRate (DVD)
audio (DVD)	fullScreen
audioChannelCount	mediaStatus (DVD)
audioExtension	playRate (DVD)
audioFormat	resolution (DVD)
audioSampleRate	selectedButton
audioStream	startTimeList
audioStreamCount	stopTimeList
buttonCount	subPicture
chapter	subPictureCount
chapterCount	title (DVD)
closedCaptions	titleCount

Propriété (suite)	
currentTime (DVD)	videoFormat
domain	volume (DVD)

Voir aussi

Acteur

Forme vectorielle

Représente un acteur forme vectorielle.

Vous pouvez ajouter un acteur forme vectorielle à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Syntaxe Lingo
_movie.newMember(#vectorshape)

// Syntaxe JavaScript
_movie.newMember("vectorshape");
```

Certaines des méthodes ou propriétés suivantes s'appliquent uniquement aux images-objets créées à partir d'un acteur forme vectorielle.

Résumé des méthodes pour le type de média Forme vectorielle

Méthode
addVertex()
deleteVertex()
moveVertex()
moveVertexHandle()
newCurve()
showProps()

Résumé des propriétés pour le type de média Forme vectorielle

Propriété	
antiAlias	imageEnabled
backgroundColor	originH
broadcastProps	originMode
centerRegPoint	originPoint
closed	originV
curve	regPointVertex
defaultRect	scale (acteur)
defaultRectMode	scaleMode
endColor	strokeColor

Propriété (suite)	
fillColor	strokeWidth
fillCycles	vertex
fillDirection	vertexList
fillMode	viewH
fillOffset	viewPoint
fillScale	viewScale
flashRect	viewV
gradientType	

Voir aussi

Acteur

GIF animé

Représente un acteur GIF animé.

Vous pouvez ajouter un acteur GIF animé à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Syntaxe Lingo
_movie.newMember(#animgif)

// Syntaxe JavaScript
_movie.newMember("animgif");
```

Certaines des méthodes ou propriétés suivantes s'appliquent uniquement aux images-objets créées à partir d'un acteur GIF animé.

Résumé des méthodes pour le type de média GIF animé

Méthode
resume()
rewind() (GIF animé, Flash)

Résumé des propriétés pour le type de média GIF animé

Propriété
directToStage
frameRate
linked
path (animation)
playBackMode

Voir aussi

Acteur

Palette de couleurs

Représente la palette de couleurs associée à un acteur bitmap.

Une acteur palette de couleurs n'a aucune méthode ou propriété à laquelle il est possible d'accéder directement. Les méthodes et propriétés suivantes sont simplement associées aux palettes de couleurs.

Vous pouvez ajouter un acteur palette de couleurs à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Syntaxe Lingo
_movie.newMember(#palette)

// Syntaxe JavaScript
_movie.newMember("palette");
```

Vous pouvez associer un acteur bitmap à un acteur palette de couleurs à l'aide de la propriété `palette` de l'acteur bitmap. L'exemple suivant attribut la propriété `palette` de l'acteur bitmap `acteurBmp` à l'acteur palette de couleurs `acteurPaletteDeCouleurs`. La valeur de la propriété `palette` reflète le numéro de l'acteur palette de couleurs.

```
-- Syntaxe Lingo
member("acteurBmp").palette = member("acteurPaletteDeCouleurs")

// Syntaxe JavaScript
member("acteurBmp").palette = member("acteurPaletteDeCouleurs");
```

Après avoir associé un acteur bitmap à un acteur palette de couleurs, vous ne pouvez pas supprimer l'acteur palette de couleurs avant d'avoir supprimé son association à l'acteur bitmap.

Résumé des méthodes pour le type de média Palette de couleurs

Méthode
`color()`

Résumé des propriétés pour le type de média Palette de couleurs

Propriété
`depth (Bitmap)`
`palette`
`paletteMapping`

Voir aussi

`Bitmap`, `Acteur`, `palette`

Police

Représente un acteur police.

Vous pouvez ajouter un acteur police à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Syntaxe Lingo
_movie.newMember(#font)

// Syntaxe JavaScript
_movie.newMember("font");
```

Résumé des propriétés pour le type de média Police

Propriété
bitmapSizes
characterSet
fontStyle
originalFont
recordFont

Voir aussi

Acteur

QuickTime

Représente un acteur QuickTime.

Vous pouvez ajouter un acteur QuickTime à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Syntaxe Lingo
_movie.newMember(#quicktimemedia)

// Syntaxe JavaScript
_movie.newMember("quicktimemedia");
```

Certaines des méthodes ou propriétés suivantes s'appliquent uniquement aux images-objets créées à partir d'un acteur QuickTime.

Résumé des méthodes pour le type de média QuickTime

Méthode	
enableHotSpot()	trackEnabled
getHotSpotRect()	trackNextKeyTime
nudge()	trackNextSampleTime
ptToHotSpotID()	trackPreviousKeyTime
quickTimeVersion()	trackPreviousSampleTime
qtRegisterAccessKey()	trackStartTime (acteur)

Méthode (suite)	
qtUnRegisterAccessKey()	trackStopTime (acteur)
setTrackEnabled()	trackText
swing()	trackType (acteur)
trackCount (acteur)	

Résumé des propriétés pour le type de média QuickTime

Propriété	
fieldOfView	nodeType
hotSpotEnterCallback	pan (propriété QTVR)
hotSpotExitCallback	percentStreamed (acteur)
invertMask	preLoad (acteur)
isVRMovie	rotation
loopBounds	scale (acteur)
mask	staticQuality
motionQuality	tilt
mouseLevel	translation
node	triggerCallback
nodeEnterCallback	warpMode
nodeExitCallback	

Voir aussi

Acteur

RealMedia

Représente un acteur RealMedia.

Vous pouvez ajouter un acteur RealMedia à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Syntaxe Lingo
_movie.newMember(#realmedia)

// Syntaxe JavaScript
_movie.newMember("realmedia");
```

Certaines des méthodes ou propriétés suivantes s'appliquent uniquement aux images-objets créées à partir d'un acteur RealMedia.

Résumé des méthodes pour le type de média RealMedia

Méthode
pause() (RealMedia, SWA, Windows Media)
play() (RealMedia, SWA, Windows Media)
realPlayerNativeAudio()
realPlayerPromptToInstall()
realPlayerVersion()
seek()
stop() (RealMedia, SWA, Windows Media)

Résumé des propriétés pour le type de média RealMedia

Propriété	
audio (RealMedia)	password
currentTime (RealMedia)	pausedAtStart (RealMedia, Windows Media)
displayRealLogo	percentBuffered
duration (RealMedia, SWA)	soundChannel (RealMedia)
image (RealMedia)	state (RealMedia)
lastError	userName (RealMedia)
mediaStatus (RealMedia, Windows Media)	video (RealMedia, Windows Media)

Voir aussi

Acteur

Shockwave 3D

Représente un acteur Macromedia Shockwave 3D.

Un acteur Shockwave 3D (ou, tout simplement, 3D) est différent des autres acteurs car il contient un univers 3D complet. Un univers 3D contient un ensemble d'objets propres aux acteurs 3D vous permettant d'ajouter des fonctionnalités 3D à une animation.

Vous pouvez ajouter un acteur 3D à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Syntaxe Lingo
_movie.newMember(#shockwave3d)

// Syntaxe JavaScript
_movie.newMember("shockwave3d");
```

Pour plus d'informations sur les objets et API dont disposent les acteurs 3D, consultez le Chapitre 8, *Objets 3D*, page 153.

Voir aussi

Acteur

Shockwave Audio

Représente un acteur Shockwave Audio.

Vous pouvez ajouter un acteur Shockwave Audio à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Syntaxe Lingo
_movie.newMember(#swa)

// Syntaxe JavaScript
_movie.newMember("swa");
```

Résumé des événements pour le type de média Shockwave Audio

Evénements
on cuePassed

Résumé des méthodes pour le type de média Shockwave Audio

Méthode
getError() (Flash, SWA)
getErrorString()
isPastCuePoint()
pause() (RealMedia, SWA, Windows Media)
play() (RealMedia, SWA, Windows Media)
preLoadBuffer()
stop() (RealMedia, SWA, Windows Media)

Résumé des propriétés pour le type de média Shockwave Audio

Propriété	
bitRate	percentStreamed (acteur)
bitsPerSample	preLoadTime
channelCount	sampleRate
copyrightInfo (SWA)	sampleSize
cuePointNames	soundChannel (SWA)
cuePointTimes	state (Flash, SWA)
duration (RealMedia, SWA)	streamName
loop (acteur)	URL
mostRecentCuePoint	volume (acteur)
numChannels	

Voir aussi

Acteur

Son

Représente un acteur utilisé pour stocker et faire référence à des échantillons de sons.

Les échantillons de sons sont contrôlés par les objets de base Son et Piste audio. Un acteur son ne possède pas d'API et utilise les API des objets Son et Piste audio pour contrôler son comportement.

Vous pouvez ajouter un acteur son à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Syntaxe Lingo
_movie.newMember(#sound)

// Syntaxe JavaScript
_movie.newMember("sound");
```

Pour plus d'informations sur les objets et API que vous pouvez utiliser pour contrôler des échantillons de sons, consultez *Son*, page 126 et *Piste audio*, page 124.

Voir aussi

`Acteur`

Texte

Représente un acteur texte.

Vous pouvez ajouter un acteur texte à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Syntaxe Lingo
_movie.newMember(#text)

// Syntaxe JavaScript
_movie.newMember("text");
```

Résumé des événements pour le type de média Texte

Evénement
`on hyperlinkClicked`

Résumé des méthodes pour le type de média Texte

Méthode
`count()`
`pointInHyperlink()`
`pointToChar()`
`pointToItem()`
`pointToLine()`
`pointToParagraph()`
`pointToWord()`

Résumé des propriétés pour le type de média Texte

Propriété	
antiAlias	hyperlink
antiAliasThreshold	hyperlinkRange
bottomSpacing	hyperlinks
charSpacing	hyperlinkState
firstIndent	kerning
fixedLineSpace	kerningThreshold
font	RTF
fontStyle	selectedText
HTML	useHypertextStyles

Voir aussi

Acteur

Windows Media

Représente un acteur Windows Media.

Vous pouvez ajouter un acteur Windows Media à une animation en utilisant la méthode `newMember()` de l'objet Animation.

```
-- Syntaxe Lingo
_movie.newMember(#windowsmedia)

// Syntaxe JavaScript
_movie.newMember("windowsmedia");
```

Certaines des méthodes ou propriétés suivantes s'appliquent uniquement aux images-objets créées à partir d'un acteur Windows Media.

Résumé des méthodes pour le type de média Windows Media

Méthode
pause() (RealMedia, SWA, Windows Media)
play() (RealMedia, SWA, Windows Media)
playFromToTime()
rewind() (Windows Media)
stop() (RealMedia, SWA, Windows Media)

Résumé des propriétés pour le type de média Windows Media

propriété	
audio (Windows Media)	pausedAtStart (RealMedia, Windows Media)
directToStage	playRate (Windows Media)
duration (acteur)	video (RealMedia, Windows Media)
height	volume (image-objet)
loop (Windows Media)	width
mediaStatus (RealMedia, Windows Media)	

Voir aussi

Acteur

CHAPITRE 7
Objets de scripting

Les objets de scripting, également connus sous le nom d'extensions Xtra dans Macromedia Director MX 2004 fournissent l'accès à la fonctionnalité des composants du logiciel qui sont installés avec Director et étendent la fonctionnalité principale de celui-ci. Les Xtras existants fournissent certaines fonctions telles que l'importation de filtres et la connexion à Internet. Vous pouvez créer vos propres Xtras si vous savez programmer en langage C.

Pour voir comment les objets de scripting sont liés les uns aux autres, ainsi qu'aux autres objets de Director, veuillez consulter *Diagramme de modèles d'objets*, page 59.

Fileio

Permet d'effectuer des opérations d'entrée et de sortie de fichier.

Vous pouvez créer une référence à un objet Fileio à l'aide de l'opérateur `new`.

```
-- Syntaxe Lingo
objFileio = new xtra("fileio")

// Syntaxe JavaScript
var objFileio = new xtra("fileio");
```

Résumé des méthodes pour l'objet Fileio

Méthode	
`closeFile()`	`readFile()`
`createFile()`	`readLine()`
`delete()`	`readToken()`
`displayOpen()`	`readWord()`
`displaySave()`	`setFilterMask()`
`error()`	`setFinderInfo()`
`fileName()`	`setNewLineConversion()`
`getFinderInfo()`	`setPosition()`
`getLength()`	`status()`
`getOSDirectory()`	`version()`

Méthode (suite)	
getPosition()	writeChar()
openFile()	writeReturn()
readChar()	writeString()

NetLingo

Permet d'effectuer des opérations de réseau comme par exemple, obtenir ou lire en continu un support à partir d'un réseau, vérifier la disponibilité du réseau, vérifier la progression d'une opération de réseau, etc.

Vous pouvez créer une référence à un objet NetLingo à l'aide de l'opérateur `new`.

```
-- Syntaxe Lingo
objNetLingo = new xtra("netlingo")

// Syntaxe JavaScript
var objNetLingo = new xtra("netlingo");
```

Résumé des méthodes pour l'objet NetLingo

Méthode	
browserName()	netDone()
cacheDocVerify()	netError()
cacheSize()	netLastModDate()
clearCache	netMIME()
downloadNetThing	netStatus
externalEvent()	netTextResult()
getLatestNetID	postNetText
getNetText()	preloadNetThing()
getStreamStatus()	proxyServer
gotoNetMovie	tellStreamStatus()
gotoNetPage	URLEncode
netAbort	

SpeechXtra

Permet d'ajouter à une animation la fonctionnalité de synthèse de parole à partir du texte.

Vous pouvez créer une référence à un objet SpeechXtra à l'aide de l'opérateur `new`.

```
-- Syntaxe Lingo
objSpeech = new xtra("speechxtra")

// Syntaxe JavaScript
var objSpeech = new xtra("speechxtra");
```

Résumé des méthodes pour l'objet SpeechXtra

Méthode	
voiceCount()	voiceSet()
voiceGet()	voiceSetPitch()
voiceGetAll()	voiceSetRate()
voiceGetPitch()	voiceSetVolume()
voiceGetRate()	voiceSpeak()
voiceGetVolume()	voiceState()
voiceInitialize()	voiceStop()
voicePause()	voiceWordPos()
voiceResume()	

XML Parser

Permet d'effectuer des analyses XML.

Vous pouvez créer une référence à un objet XML Parser à l'aide de l'opérateur `new`.

```
-- Syntaxe Lingo
objXml = new xtra("xmlparser")

// Syntaxe JavaScript
var objXml = new xtra("xmlparser");
```

Résumé des méthodes pour l'objet XML Parser

Méthode
count()
doneParsing()
getError() (XML)
ignoreWhiteSpace()
makeList()
makeSubList()
parseString()
parseURL()

Résumé des propriétés pour l'objet XML Parser

Propriété
attributeName
attributeValue
child (XML)
name (XML)

CHAPITRE 8
Objets 3D

Les objets 3D permettent d'ajouter la fonctionnalité 3D à une animation. Ces objets sont à la fois exposés à la syntaxe Lingo et à la syntaxe JavaScript au sein de Macromedia Director MX 2004, des projections et du lecteur Macromedia Shockwave.

Vous avez accès à ces objets 3D par le biais des acteurs Shockwave 3D (ou 3D uniquement). Vous pouvez également créer des images-objets 3D à partir des acteurs 3D. Les acteurs 3D ainsi que les images-objets 3D comportent des fonctionnalités qui leur sont propres. Ils ont également accès aux fonctionnalités des acteurs et images-objets non 3D dont les API sont respectivement indiqués par les objets principaux `Member` et `Sprite`.

Un acteur 3D est différent des autres acteurs dans le sens où un acteur 3D contient tout un monde en 3D. Un monde en 3D comprend des objets qui permettent l'accès à la fonctionnalité 3D. Tous les objets d'un monde en 3D se fondent sur un objet de base appelé noeud. La forme la plus simple d'un nœud dans un monde en 3D est un objet Groupe ; cet objet Group est fondamentalement le nœud de base. Tous les autres objets d'un monde en 3D se fondent sur un objet Groupe, ce qui signifie que les autres objets héritent de la fonctionnalité d'un objet Groupe en plus de posséder la fonctionnalité propre à ces objets.

Pour voir comment les objets 3D sont liés les uns aux autres, ainsi qu'aux autres objets de Director, veuillez consulter *Diagramme de modèles d'objets*, page 59.

Director est livré avec deux Xtras vous donnant accès aux objets 3D :

- Xtra 3D (3DAuth.x32 sous Windows, 3D Auth Xtra sous Macintosh) prend en charge la fenêtre média 3D dans Director.
- Xtra 3D Media (Shockwave 3D Asset.x32 sous Windows, 3D Asset Xtra sur Macintosh) prend en charge le média 3D proprement dit.

Pour accéder aux objets 3D lors de la programmation ou de l'exécution, votre animation doit comprendre un Xtra 3D.

Acteur

Représente un acteur Shockwave 3D.

Un acteur Shockwave 3D (ou 3D uniquement) comprend tout un monde en 3D. Un monde en 3D comprend un ensemble d'objets vous permettant d'ajouter un fonctionnalité 3D à une animation.

Vous pouvez créer une référence à un acteur 3D soit à l'aide de la fonction `member()`, soit à l'aide de la propriété `member` de l'objet Animation ou Image-objet. Vous pouvez utiliser les mêmes techniques que pour créer une référence à un acteur qui n'est pas en 3D.

- Utilisez la fonction `member()` de haut niveau.

```
-- Syntaxe Lingo
acteur3d = member("magie")

// Syntaxe JavaScript
var acteur3d = member("magie");
```

- Utilisez la propriété `member` de l'objet Image-objet.

```
-- Syntaxe Lingo
acteur3d = sprite(1).member;

// Syntaxe JavaScript
var acteur3d = sprite(1).member;
```

Résumé des méthodes pour l'objet Acteur

Méthode	
camera()	model
cloneModelFromCastmember	modelResource
cloneMotionFromCastmember	motion()
deleteCamera	newCamera
deleteGroup	newGroup
deleteLight	newLight
deleteModel	newMesh
deleteModelResource	newModel
deleteMotion	newModelResource
deleteShader	newShader
deleteTexture	newTexture
extrude3D	resetWorld
group()	revertToWorldDefaults
light()	shader()
loadFile()	texture()

Résumé des propriétés pour l'objet Acteur

Propriété	
ambientColor	loop (3D)
animationEnabled	model
bevelDepth	modelResource
bevelType	motion
bytesStreamed (3D)	percentStreamed (3D)
camera	preLoad (3D)
cameraPosition	reflectivity
cameraRotation	shader
diffuseColor	smoothness
directionalColor	specularColor
directionalPreset	state (3D)
directToStage	streamSize (3D)
displayFace	texture
displayMode	textureMember
group	textureType
light	tunnelDepth

Voir aussi

Caméra, Groupe, Lumière, Modèle, Ressource modèle, Mouvement, Matériau, Image-objet, Texture

Caméra

Représente un objet Caméra.

Une caméra contrôle la manière dont une image-objet 3D visualise le monde en 3D. Une image-objet 3D affiche une vue de caméra particulière dans le monde.

Vous pouvez créer une référence à une caméra à l'aide de la propriété `camera` de l'objet 3D `Member`. La propriété `camera` choisit la caméra située à un endroit précis de l'index dans la liste des caméras. Avec Lingo, vous pouvez directement utiliser la propriété `camera` de l'objet 3D `Member` afin de créer une référence. Dans la syntaxe JavaScript, vous devez utiliser la méthode `getPropRef()` pour créer une référence.

Dans l'exemple suivant, une référence à la deuxième caméra de la « chambre familiale » de l'acteur 3D est créée et affectée à la variable `maCaméra`.

```
-- Syntaxe Lingo
maCaméra = member("chambre familiale").camera[2]

// Syntaxe JavaScript
var maCaméra = member("chambre familiale").getPropRef("camera", 2);
```

Résumé des méthodes pour l'objet Caméra

Méthode
addBackdrop
addOverlay
insertBackdrop
insertOverlay
removeBackdrop
removeOverlay

Résumé des propriétés pour l'objet Caméra

Propriété	
backdrop	fog.far (brouillard)
backdrop[].blend (3D)	fog.near (brouillard)
backdrop[].loc (fond et recouvrement)	hither
backdrop[].regPoint (3D)	orthoHeight
backdrop[].rotation (fond et recouvrement)	overlay
backdrop[].scale (3D)	overlay[].blend (3D)
backdrop[].source	overlay[].loc (fond et recouvrement)
backdrop.count (3D)	overlay[].regPoint (3D)
child (3D)	overlay[].rotation (fond et recouvrement)
colorBuffer.clearAtRender	overlay[].scale (3D)
colorBuffer.clearValue	overlay[].source
fieldOfView (3D)	overlay.count (3D)
fog.color()	projection
fog.decayMode	rootNode
fog.enabled (brouillard)	yon

Voir aussi

Groupe, Lumière, Modèle, Ressource modèle, Mouvement, Matériau, Texture

Groupe

Représente un modèle n'ayant ni ressource ni matériaux.

Un groupe est un nœud de base et correspond tout simplement à un point dans l'espace représenté par une transformation. Vous pouvez affecter des enfants et des parents à ce nœud afin de regrouper des modèles, des lumières, des caméras ou d'autres groupes.

Le groupe de base est appelé monde, ce qui revient fondamentalement au même qu'acteur 3D.

Vous pouvez créer une référence à un groupe à l'aide de la propriété `group` de l'objet 3D `Member`. La propriété `group` choisit le groupe situé à un endroit précis de l'index dans la liste de groupes. En Lingo, vous pouvez directement utiliser la propriété `group` de l'objet 3D `Member` pour créer une référence. Dans la syntaxe JavaScript, vous devez utiliser la méthode `getPropRef()` pour créer une référence.

Dans l'exemple suivant, une référence au premier groupe de l'acteur 3D `espace` est créée et affectée à la variable `monGroupe`.

```
-- Syntaxe Lingo
monGroupe = member("espace").group[1]

// Syntaxe JavaScript
var monGroupe = member("espace").getPropRef("group", 1);
```

Résumé des méthodes pour l'objet Groupe

Méthode	
addChild	pointAt
addToWorld	registerScript()
clone	removeFromWorld
cloneDeep	rotate
getWorldTransform()	scale (commande)
isInWorld()	translate

Résumé des propriétés pour l'objet Groupe

Propriété
name (3D)
parent
pointAtOrientation
transform (propriété)
userData
worldPosition

Voir aussi

Caméra, Lumière, Modèle, Ressource modèle, Mouvement, Matériau, Texture

Image-objet

Représente une image-objet 3D créée à partir d'un acteur Shockwave 3D.

Vous pouvez créer une référence à une image-objet 3D à l'aide de la fonction `sprite()` de haut niveau, de la propriété `sprite` de l'objet Animation ou bien de la propriété `sprite` de l'objet Piste d'images-objets. Vous pouvez utiliser les mêmes techniques que pour créer une référence à une image-objet qui n'est pas 3D.

- Utilisez la fonction `sprite()` de haut niveau.

```
-- Syntaxe Lingo
imageObjet3d = sprite(1)

// Syntaxe JavaScript
var imageObjet3d = sprite(1);
```

- Utilisez la propriété `sprite` de l'objet Animation.

```
-- Syntaxe Lingo
imageObjet3d = _movie.sprite["saule"]

// Syntaxe JavaScript
var imageObjet3d = _movie.sprite["saule"];
```

- Utilisez la propriété `sprite` de l'objet Piste d'images-objets.

```
-- Syntaxe Lingo
imageObjet3d = channel(3).sprite

// Syntaxe JavaScript
var imageObjet3d = channel(3).sprite;
```

Résumé des méthodes pour l'objet Image-objet

Méthode
`addCamera`
`cameraCount()`
`deleteCamera`

Résumé des propriétés pour l'objet Image-objet

Propriété
`antiAliasingEnabled`
`backColor`
`camera`
`directToStage`

Voir aussi

`Caméra`, `Acteur`

Lumière

Représente une lumière dans un monde en 3D.

Les lumières sont utilisées pour éclairer un monde en 3D. Sans les lumières, on ne peut pas voir les objets de ce monde.

Vous pouvez créer une référence à une lumière à l'aide de la propriété `light` de l'objet 3D `Member`. La propriété `light` choisit une lumière à un endroit précis de l'index dans la liste des lumières. En Lingo, vous pouvez directement utiliser la propriété `light` de l'objet 3D `Member` pour créer une référence. Dans la syntaxe JavaScript, vous devez utiliser la méthode `getPropRef()` pour créer une référence.

Dans l'exemple suivant, une référence à la troisième lumière de la « salle d'animation » de l'acteur 3D est créée et affectée à la variable `maLumière`.

```
-- Syntaxe Lingo
maLumière = member("salle d'animation").light[3]

// Syntaxe JavaScript
var maLumière = member("salle d'animation").getPropRef("light", 3);
```

Résumé des propriétés pour l'objet Lumière

Propriété
`attenuation`
`color (lumière)`
`specular (lumière)`
`spotAngle`
`spotDecay`
`type (lumière)`

Voir aussi

`Caméra`, `Groupe`, `Modèle`, `Ressource modèle`, `Mouvement`, `Matériau`, `Texture`

Matériau

Représente la couleur de la surface d'un modèle.

Vous pouvez dessiner des images à la surface d'un modèle en appliquant une ou plusieurs textures à chaque matériau.

Vous pouvez créer une référence à un matériau à l'aide de la propriété `shader` de l'objet 3D `Member`. La propriété `shader` choisit la nuance située à un endroit précis de l'index dans la liste des matériaux. En Lingo, vous pouvez directement utiliser la propriété `shader` de l'objet 3D `Member` afin de créer une référence. Dans la syntaxe JavaScript, vous devez utiliser la méthode `getPropRef()` pour créer une référence.

Dans l'exemple suivant, une référence à la deuxième nuance de l'acteur 3D `triangle` est créée et affectée à la variable `monMatériau`.

```
-- Syntaxe Lingo
monMatériau = member("triangle").shader[2]

// Syntaxe JavaScript
var monMatériau = member("triangle").getPropRef("shader", 2);
```

Modèle

Représente un objet visible qu'un utilisateur peut voir dans un monde en 3D.

Un modèle utilise une ressource modèle et occupe une orientation et un emplacement précis dans un monde en 3D. Une ressource modèle est un élément de la géométrie 3D que vous pouvez utiliser pour dessiner des modèles 3D. Un modèle définit également l'apparence d'une ressource modèle, telle que les textures et les matériaux qui sont utilisés. Pour plus d'informations sur la relation entre les modèles et les ressources modèles, veuillez consulter la rubrique Utilisation de Director dans le panneau d'aide de Director.

Vous pouvez créer une référence à un modèle à l'aide de la propriété `model` de l'objet 3D `Member`. La propriété `model` choisit un modèle situé à un endroit précis de l'index dans la liste des modèles. En Lingo, vous pouvez directement utiliser la propriété `model` de l'objet 3D `Member` afin de créer une référence. Dans la syntaxe JavaScript, vous devez utiliser la méthode `getPropRef()` pour créer une référence.

Dans l'exemple suivant, une référence au deuxième modèle de l'acteur 3D `Transportation` est créée et affectée à la variable `monModèle`.

```
-- Syntaxe Lingo
monModèle = member("Transportation").model[2]

// Syntaxe JavaScript
var monModèle = member("Transportation").getPropRef("model", 2);
```

Un modèle comprend également des modificateurs qui contrôlent le rendu du modèle et le comportement de l'animation. Les modificateurs sont liés au modèle à l'aide de la méthode `addModifier()`. Une fois un modificateur lié à un modèle, ses propriétés peuvent être manipulées par le script.

Les modificateurs suivants sont disponibles avec un modèle :

Modificateur	Description
BonesPlayer	Modifie la forme d'un modèle après l'exécution.
Collision	Permet à un modèle d'être prévenu en cas de collision et d'agir en fonction.
Inker	Ajoute une silhouette, des plis et des bords à un modèle existant.
KeyframePlayer	Modifie les propriétés `transform` du modèle après l'exécution.
LOD	Permet de contrôler le nombre de polygones utilisés par modèle pour le rendu d'un modèle, en fonction de la distance du modèle par rapport à la caméra. Le modificateur LOD est également disponible pour les ressources modèles.

Modificateur	Description
MeshDeform	Modifie la forme d'une ressource modèle existante lors de l'exécution.
SDS	Le rendu du modèle comporte des détails géométriques supplémentaires dans la zone du modèle que la caméra est en train de fixer.
Toon	Modifie le rendu du modèle afin d'imiter le style des bandes dessinées.

Pour plus d'informations sur les méthodes, propriétés et événements disponibles pour les modificateurs, veuillez consulter la rubrique Utilisation de Director dans le panneau d'aide de Director.

Mouvement

Représente une séquence d'animation prédéfinie qui implique le mouvement d'un modèle ou d'un composant d'un modèle.

Vous pouvez définir les mouvements individuels de manière à ce qu'ils se lisent d'eux-mêmes ou avec d'autres mouvements. Par exemple, une course peut être combinée à un saut afin de simuler le saut d'une personne au-dessus d'une flaque d'eau.

Vous pouvez créer une référence à un mouvement à l'aide de la propriété `motion` de l'objet 3D `Member`. La propriété `motion` choisit le mouvement situé à un endroit précis de l'index dans la liste des mouvements. En Lingo, vous pouvez utiliser directement la propriété `motion` de l'objet 3D `Member` pour créer une référence. Dans la syntaxe JavaScript, vous devez utiliser la méthode `getPropRef()` pour créer une référence.

Dans l'exemple suivant, une référence au quatrième mouvement de l'acteur 3D `athlète` est créée et affectée à la variable `monMouvement`.

```
-- Syntaxe Lingo
monMouvement = member("athlète").motion[4]

// Syntaxe JavaScript
var monMouvement = member("athlète").getPropRef("motion", 4);
```

Ressource modèle

Représente un élément de la géométrie 3D utilisé pour dessiner des modèles 3D.

Un modèle utilise une ressource modèle et occupe une orientation et emplacement précis dans un monde en 3D. Un modèle définit également l'apparence d'une ressource modèle, telle que les textures et matériaux utilisés.

Pour plus d'informations sur la relation entre les modèles et les ressources modèles, ainsi que sur l'utilisation des modèles et des ressources modèles, veuillez consulter la rubrique Utilisation de Director dans le panneau d'aide de Director.

Vous pouvez créer une référence à une ressource modèle à l'aide de la propriété `modelResource` de l'objet 3D `Member`. La propriété `modelResource` choisit la ressource modèle située à un endroit précis de l'index dans la liste des ressources modèles. En Lingo, vous pouvez utiliser directement la propriété `modelResource` de l'objet 3D `Member` pour créer une référence. Dans la syntaxe JavaScript, vous devez utiliser la méthode `getPropRef()` pour créer une référence.

Dans l'exemple suivant, une référence à la deuxième ressource modèle de l'acteur 3D `roues` est créée et affectée à la variable `maRessourceModèle`.

```
-- Syntaxe Lingo
maRessourceModèle = member("roues").modelResource[2]

// Syntaxe JavaScript
var maRessourceModèle = member("roues").getPropRef("modelResource", 2);
```

Services de rendu

Représente l'objet global qui comprend une liste de propriétés dont les valeurs influencent les propriétés communes de rendu de tous les acteurs et images-objets 3D.

Vous pouvez accéder à l'objet services de rendu global à l'aide de la fonction `getRendererServices()` de haut niveau.

Dans l'exemple suivant, on accède à la propriété de rendu de l'objet services de rendu global et la valeur est affectée à la variable `monRendu`.

```
-- Syntaxe Lingo
monRendu = getRendererServices().renderer

// Syntaxe JavaScript
var monRendu = getRendererServices().renderer;
```

Résumé des méthodes pour l'objet Services de rendu

Méthode
`getHardwareInfo()`

Résumé des propriétés pour l'objet Services de rendu

Propriété
`modifiers`
`primitives`
`renderer`
`rendererDeviceList`
`textureRenderFormat`

Voir aussi

`Acteur`, `Image-objet`

Texture

Représente la texture appliquée à un matériau.

Vous pouvez créer une référence à une texture à l'aide de la propriété `texture` de l'objet 3D `Member`. La propriété `texture` choisit la texture située à un endroit précis de l'index dans la liste des textures. En Lingo, vous pouvez utiliser directement la propriété `texture` de l'objet 3D `Member` afin de créer une référence. Dans la syntaxe JavaScript, vous devez utiliser la méthode `getPropRef()` pour créer une référence.

Dans l'exemple suivant, une référence à la première texture de l'acteur 3D `triangle` est créée et affectée à la variable `maTexture`.

```
-- Syntaxe Lingo
maTexture = member("triangle").texture[1]

// Syntaxe JavaScript
var maTexture = member("triangle").getPropRef("texture", 1);
```

CHAPITRE 9
Constantes

Cette section propose une liste alphabétique de toutes les constantes disponibles dans Macromedia Director MX 2004.

La plupart de ces constantes ne s'appliquent qu'à Lingo. La syntaxe JavaScript contient des constantes qui sont similaires aux constantes Lingo énumérées ici ; nous vous fournirons donc des explications sur l'usage de la Syntaxe JavaScript ainsi que des exemples pour vous aider à associer les fonctionnalités des constantes Lingo à leurs équivalents les plus proches dans la syntaxe JavaScript. Pour plus d'informations sur les constantes de la syntaxe JavaScript, consultez les nombreuses autres ressources consacrées à ce sujet.

" (chaîne)

Utilisation

```
-- Syntaxe Lingo
"

// Syntaxe JavaScript
"
```

Description

Constante de chaîne ; lorsqu'ils sont utilisés avant et après une chaîne, les guillemets droits signifient que la chaîne est une chaîne littérale, et non une variable, une valeur numérique ou un élément de script. Les guillemets droits doivent toujours encadrer les noms littéraux des acteurs, des distributions, des fenêtres et des fichiers externes.

Exemple

L'instruction suivante utilise les guillemets droits pour indiquer que la chaîne « San Francisco » est une chaîne littérale, le nom d'un acteur :

```
-- Syntaxe Lingo
put member("San Francisco").loaded

// Syntaxe JavaScript
put(member("San Francisco").loaded);
```

Voir aussi

QUOTE

BACKSPACE

Utilisation

```
-- Syntaxe Lingo
BACKSPACE

// Syntaxe JavaScript
51 // value of _key.keyCode
```

Description

Constante; représente la touche Suppression/Retour Arrière. Cette touche est appelée Retour Arrière sur un clavier Windows et Suppr sur un clavier Macintosh.

Exemple

Le gestionnaire `on keyDown` suivant vérifie si l'utilisateur a appuyé sur la touche Retour Arrière et, le cas échéant, appelle le gestionnaire `effacerEntrée`:

```
-- Syntaxe Lingo
on keyDown
  if (_key.key = BACKSPACE) then effacerEntrée
  _movie.stopEvent()
end keyDown

// Syntaxe JavaScript
function keyDown() {
  if (_key.keyCode == 51) {
    clearEntry();
    _movie.stopEvent();
  }
}
```

EMPTY

Utilisation

```
-- Syntaxe Lingo
VIDES

// Syntaxe JavaScript
""
```

Description

Constante de caractères ; représente la chaîne vide, `""`, une chaîne sans caractères.

Exemple

L'instruction suivante efface tous les caractères de l'acteur champ Notice en donnant au champ la valeur `EMPTY` :

```
-- Syntaxe Lingo
member("Notice").text = EMPTY

// Syntaxe JavaScript
member("Notice").text = "";
```

ENTER

Utilisation

```
-- Syntaxe Lingo
ENTER

// Syntaxe JavaScript
3 // value of _key.keyCode
```

Description

Constante de caractères ; représente la touche Entrée (Windows) ou Retour (Macintosh) pour un retour de chariot.

Sur les claviers PC, l'élément `ENTER` ne se rapporte qu'à la touche Entrée du clavier numérique.

Pour une animation lue comme applet, utilisez `RETURN` pour spécifier la touche Retour sous Windows et la touche Entrée sur le Macintosh.

Exemple

Cette instruction vérifie si la touche Entrée est enfoncée et, le cas échéant, envoie la tête de lecture sur l'image `ajouterLaSomme` :

```
-- Syntaxe Lingo
on keyDown
  if (_key.key = ENTER) then _movie.go("ajouterLaSomme")
end

// Syntaxe JavaScript
function keyDown() {
  if (_key.keyCode == 3) {
    _movie.go("ajouterLaSomme");
  }
}
```

Voir aussi

`RETURN (constante)`

FALSE

Utilisation

```
-- Syntaxe Lingo
FALSE

// Syntaxe JavaScript
FALSE
```

Description

Constante ; s'applique à une expression qui est logiquement fausse`FALSE`, telle que 2 > 3. Lorsque traitée comme un nombre, `FALSE` a la valeur numérique de 0. Inversement, 0 est traité comme `FALSE`.

Exemple

L'instruction suivante désactive la propriété `soundEnabled` en lui donnant la valeur `FALSE` :

```
-- Syntaxe Lingo
_sound.soundEnabled = FALSE

// Syntaxe JavaScript
_sound.soundEnabled = false;
```

Voir aussi

`if`, `not`, `TRUE`

PI

Utilisation

```
-- Syntaxe Lingo
PI

// Syntaxe JavaScript
Math.PI
```

Description

Constante ; renvoie la valeur de pi (π), le rapport de la circonférence d'un cercle et de son diamètre, sous la forme d'un nombre à virgule flottante. La valeur est arrondie au nombre de décimales défini par la propriété `floatPrecision`.

Exemple

L'instruction suivante utilise la constante `PI` dans une équation destinée à calculer la surface d'un cercle :

```
-- Syntaxe Lingo
vRadius = 3
vArea = PI*power(vRadius, 2)
trace(vArea) - résultats dans 28.2743

// Syntaxe JavaScript
var vRadius = 3;
vArea = Math.PI*Math.pow(vRadius, 2);
trace(vArea); // résultats dans 28.274333882308138
```

QUOTE

Utilisation

```
-- Syntaxe Lingo
QUOTE

// Syntaxe JavaScript
\"
```

Description

Constante; représente le caractère guillemet dans une chaîne puisque ce caractère est lui-même utilisé dans les scripts Lingo pour délimiter les chaînes.

Exemple

L'instruction suivante insère des guillemets dans une chaîne :

```
-- Syntaxe Lingo
put("Comment épelez-vous" && QUOTE & "Macromedia" & QUOTE & "?")

// Syntaxe JavaScript
put("Comment épelez-vous \"Macromedia\"?");
```

Le résultat est un jeu de guillemets placés autour du mot *Macromedia* :

```
Comment épelez-vous "Macromedia" ?
```

RETURN (constante)

Utilisation

```
-- Syntaxe Lingo
RETURN

// Syntaxe JavaScript
36 // value of _key.keyCode
\n // lorsqu'il est utilisé dans une chaîne
```

Description

Constante ; représente un retour de chariot.

Exemple

L'instruction suivante reprend la lecture d'une animation en pause lorsque l'utilisateur appuie sur le retour de chariot :

```
-- Syntaxe Lingo
if (_key.key = RETURN) then _movie.go(_movie.frame + 1)

// Syntaxe JavaScript
if (_key.keyCode == 36) {
  _movie.go(_movie.frame+1);
}
```

L'instruction suivante utilise la constante de caractère `RETURN` pour insérer un retour de chariot entre deux lignes d'un message d'alerte :

```
-- Syntaxe Lingo
_player.alert("Dernière ligne du fichier." & RETURN & "Cliquer sur OK pour
  fermer.")

// Syntaxe JavaScript
_player.alert("Dernière ligne du fichier.\nCliquez sur OK pour fermer");
```

Sous Windows, vous devez normalement placer un caractère additionnel de saut de ligne à la fin de chaque ligne. L'instruction suivante crée une chaîne de deux caractères nommée CRLF fournissant le saut de ligne additionnel :

```
CRLF = RETURN & numToChar(10)
```

SPACE

Utilisation

```
-- Syntaxe Lingo
SPACE

// Syntaxe JavaScript
49 // value of _key.keyCode
```

Description

Constante ; valeur en lecture seule représentant une espace.

Exemple

L'instruction suivante affiche Age de bronze dans la fenêtre Messages :

```
-- Syntaxe Lingo
put("Age"&SPACE&"de"&SPACE&"bronze")
```

TAB

Utilisation

```
-- Syntaxe Lingo
TAB

// Syntaxe JavaScript
48 // value of _key.keyCode
```

Description

Constante ; représente la touche Tab.

Exemple

L'instruction suivante vérifie si le caractère saisi est le caractère de tabulation et, le cas échéant, appelle le gestionnaire `champSuivant` :

```
-- Syntaxe Lingo
if (_key.key = TAB) then champSuivant

// Syntaxe JavaScript
if (_key.keyCode == 48) {
  doNextField();
}
```

Ces instructions font avancer ou reculer la tête de lecture selon que l'utilisateur appuie sur les touches Tab ou Majuscule-Tab :

```
-- Syntaxe Lingo
if (_key.key = TAB) then
  if (_key.shiftDown) then
    _movie.go(_movie.frame - 1)
  else
    _movie.go(_movie.frame +1)
  end if
end if
```

```
// Syntaxe JavaScript
if (_key.keyCode == 48) {
  if (_key.shiftDown) {
    _movie.go(_movie.frame-1);
  } else {
    _movie.go(_movie.frame+1);
  }
}
```

Voir aussi

BACKSPACE, EMPTY, RETURN (constante)

TRUE

Utilisation

```
-- Syntaxe Lingo
TRUE

// Syntaxe JavaScript
true
```

Description

Constante ; représente la valeur d'une expression logique, telle que `2 < 3`. Elle est traditionnellement une valeur numérique de 1, mais tout chiffre entier non nul donne `TRUE` dans une comparaison.

Exemple

L'instruction suivante active la propriété `soundEnabled` en lui donnant la valeur `TRUE` :

```
-- Syntaxe Lingo
_sound.soundEnabled = TRUE

// Syntaxe JavaScript
_sound.soundEnabled = true;
```

Voir aussi

FALSE, if

VOID

Utilisation

```
-- Syntaxe Lingo
VOID

// Syntaxe JavaScript
null
```

Description

Constante ; indique la valeur `VOID`.

Exemple

L'instruction suivante vérifie si la variable `variableCourante` a la valeur `VOID` :

```
-- Syntaxe Lingo
if variableCourante = VOID then
  put("Cette variable n'a aucune valeur")
end if

// Syntaxe JavaScript
if (currentVariable == null) {
  put("Cette variable n'a aucune valeur");
}
```

Voir aussi

`voidP()`

CHAPITRE 10
Evénements et messages

Cette section propose une liste alphabétique de tous les événements et messages disponibles dans Macromedia Director MX 2004.

on activateApplication

Utilisation

```
-- Syntaxe Lingo
on activateApplication
  instruction(s)
end

// Syntaxe JavaScript
function activateApplication() {
  instruction(s);
}
```

Description

Gestionnaire intégré ; exécuté lorsque la projection passe au premier plan. Ce gestionnaire est pratique lorsqu'une projection est exécutée dans une fenêtre et que l'utilisateur l'envoie à l'arrière-plan pour travailler dans une autre application. Le gestionnaire est exécuté lorsque la projection repasse au premier plan. Toutes les MIAW exécutées dans la projection peuvent également utiliser ce gestionnaire.

En cours de programmation, ce gestionnaire n'est appelé que lorsque l'option Animer en arrière-plan est activée dans les préférences générales.

Sous Windows, ce gestionnaire n'est pas appelé lorsque la projection n'est que réduite et qu'aucune application n'est passée au premier plan.

Exemple

Le gestionnaire suivant lit un son à chaque fois que l'utilisateur ramène la projection au premier plan :

```
-- Syntaxe Lingo
on activateApplication
  sound(1).queue(member("sonDouverture"))
  sound(1).play()
end
```

```
// Syntaxe JavaScript
function activateApplication() {
  sound(1).queue(member("sonDouverture"));
  sound(1).play();
}
```

Voir aussi

on deactivateApplication, activeCastLib, on deactivateWindow

on activateWindow

Utilisation

```
-- Syntaxe Lingo
on activateWindow
     instruction(s)
end

// Syntaxe JavaScript
function activateWindow()
  instruction(s);
}
```

Description

Message système et gestionnaire d'événements ; contient des instructions exécutées dans une animation lorsque l'utilisateur clique sur la fenêtre inactive et que la fenêtre passe au premier plan.

Vous pouvez utiliser un gestionnaire `on activateWindow` dans un script que vous souhaitez exécuter à chaque fois que l'animation devient active.

Le fait de cliquer sur l'animation principale (la scène principale) ne génère pas de gestionnaire `on activateWindow`.

Exemple

Ce gestionnaire lit le son Hourra lorsque la fenêtre dans laquelle l'animation est exécutée devient active :

```
-- Syntaxe Lingo
on activateWindow
  sound(2).play(member("Hourra"))
end

// Syntaxe JavaScript
function activateWindow() {
  sound(2).play(member("Hourra"));
}
```

Voir aussi

activeWindow, close(), on deactivateWindow, frontWindow, on moveWindow, open() (fenêtre)

on beginSprite

Utilisation

```
-- Syntaxe Lingo
on beginSprite
  instruction(s)
end

// Syntaxe JavaScript
function beginSprite() {
  instruction(s);
}
```

Description

Message système et gestionnaire d'événement ; contient des instructions exécutées lorsque la tête de lecture passe à une image contenant une image-objet jusqu'alors inconnue. A l'instar de `endSprite`, cet événement est généré une seule fois, même si la tête de lecture effectue une boucle sur une image, étant donné que le déclencheur est une image-objet que la tête de lecture n'avait pas encore rencontrée. L'événement est généré avant `prepareFrame`.

Director crée des instances de tout script de comportement lié à l'image-objet lorsque le message `beginSprite` est envoyé.

La référence d'objet `me` est transmise à cet événement s'il est utilisé dans un comportement. Le message est envoyé aux comportements et aux scripts d'images.

Si une image-objet commence à la première image lue dans le cadre de l'animation, le message `beginSprite` est envoyé après le message `prepareMovie` mais avant les messages `prepareFrame` et `startMovie`.

Remarque : N'oubliez pas que certaines propriétés d'image-objet telles que `rect` ne seront peut-être pas accessibles dans un gestionnaire beginSprite. En effet, le système doit calculer la propriété, ce qui ne peut se faire tant que l'image-objet n'a pas été dessinée.

Les commandes `go`, `play` et `updateStage` sont désactivées dans un gestionnaire `on beginSprite`.

Exemple

Le gestionnaire suivant lit l'acteur Georges Brassens lorsque l'image-objet commence :

```
-- Syntaxe Lingo
on beginSprite me
  sound(1).play(member("Georges Brassens"))
end

// Syntaxe JavaScript
function beginSprite() {
  sound(1).play(member("Georges Brassens"));
}
```

Voir aussi

`on endSprite`, `on prepareFrame`, `scriptInstanceList`

on closeWindow

Utilisation

```
-- Syntaxe Lingo
on closeWindow
  instruction(s)
end

// Syntaxe JavaScript
function closeWindow() {
  instruction(s);
}
```

Description

Message système et gestionnaire d'événements ; contient les instructions exécutées lorsque l'utilisateur ferme la fenêtre d'une animation en cliquant sur sa case de fermeture.

Le gestionnaire `on closeWindow` est un bon endroit pour placer les commandes que vous souhaitez exécuter à chaque fermeture de la fenêtre de l'animation.

Exemple

Le gestionnaire suivant transmet à Director la commande `forget` pour qu'il libère la fenêtre courante de la mémoire, lorsque l'utilisateur ferme la fenêtre dans laquelle l'animation est en cours de lecture :

```
-- Syntaxe Lingo
on closeWindow
  -- opérations de maintenance standard
  window(1).forget()
end

// Syntaxe JavaScript
function closeWindow() {
  // opérations de maintenance standard
  window(1).forget();
}
```

on cuePassed

Utilisation

```
-- Syntaxe Lingo
on cuePassed{me,} IDdePiste, numéroDeRepère,nomDeRepère)
  instruction(s)
end

// Syntaxe JavaScript
function cuePassed(IDdePiste,numéroDeRepère,nomDeRepère) {
  instruction(s);
}
```

Description

Message système et gestionnaire d'événements ; contient des instructions exécutées à chaque fois qu'un son ou qu'une image-objet passe par un point de repère sur son média.

- `me` Le paramètre facultatif `me` représente la valeur `réfDinstanceDeScript` du script appelé. Vous devez inclure ce paramètre si vous utilisez le message dans un comportement. Si vous omettez ce paramètre, les autres arguments ne seront pas traités correctement.
- *`IDdePiste`* Le numéro de la piste audio ou d'image-objet spécifique du fichier où le point de repère a été franchi.
- *`numéroDeRepère`* Le nombre ordinal du point de repère qui déclenche l'événement dans la liste des points de repère de l'acteur.
- *`nomDeRepère`* Le nom du point de repère rencontré.

Le message est transmis, dans l'ordre, aux scripts d'image-objet, d'acteur, d'image et d'animation. Pour qu'elle reçoive l'événement, l'image-objet doit être la source du son, telle qu'une animation QuickTime ou un acteur SWA. Utilisez la propriété `isPastCuePoint` pour vérifier les points de repère dans les comportements relatifs aux images-objets qui ne génèrent aucun son.

Exemple

Le gestionnaire suivant placé dans un script d'animation ou d'image affiche tous les points de repère pour la piste audio 1 dans la fenêtre Messages :

```
-- Syntaxe Lingo
on cuePassed piste, numéro, nom
  if (channel = #Sound1) then
    put("Le point de repère" && number && "intitulé" && name && "est survenu
  dans le son 1")
  end if
end

// Syntaxe JavaScript
function cuePassed(piste, numéro, nom) {
  if (channel == symbol("Son1")) {
    put("Le point de repère" + number + "intitulé" + name + "est survenu dans
  le son 1");
  }
}
```

Voir aussi

`scriptInstanceList`, `cuePointNames`, `cuePointTimes`, `isPastCuePoint()`

on deactivateApplication

Utilisation

```
-- Syntaxe Lingo
on deactivateApplication
  instruction(s)
end

// Syntaxe JavaScript
function deactivateApplication() {
  instruction(s);
}
```

Description

Gestionnaire intégré ; exécuté lorsque la projection passe à l'arrière-plan. Ce gestionnaire est pratique lorsqu'une projection est exécutée dans une fenêtre et que l'utilisateur l'envoie à l'arrière-plan pour travailler dans une autre application. Toutes les MIAW exécutées dans la projection peuvent également utiliser ce gestionnaire.

En cours de programmation, ce gestionnaire n'est appelé que lorsque l'option Animer en arrière-plan est activée dans les préférences générales.

Sous Windows, ce gestionnaire n'est pas appelé lorsque la projection n'est que réduite et qu'aucune application n'est passée au premier plan.

Exemple

Le gestionnaire suivant lit un son à chaque fois que l'utilisateur envoie la projection à l'arrière-plan :

```
-- Syntaxe Lingo
on deactivateApplication
  sound(1).queue(member("sonDeFermeture"))
  sound(1).play()
end

// Syntaxe JavaScript
function deactivateApplication() {
  sound(1).queue(member("sonDeFermeture"));
  sound(1).play();
}
```

Voir aussi

`add (texture 3D)`, `activeCastLib`, `on deactivateWindow`

on deactivateWindow

Utilisation

```
-- Syntaxe Lingo
on deactivateWindow
  instruction(s)
end

// Syntaxe JavaScript
function deactivateWindow() {
  instruction(s);
}
```

Description

Message système et gestionnaire d'événements ; contient des instructions qui sont exécutées lorsque la fenêtre dans laquelle l'animation est lue, est désactivée. Le gestionnaire d'événements `on deactivate` est un bon endroit pour placer le code que vous souhaitez exécuter dès la désactivation d'une fenêtre.

Exemple

Le gestionnaire suivant lit le son Ronflement lorsque la fenêtre dans laquelle l'animation est lue, est désactivée :

```
-- Syntaxe Lingo
on deactivateWindow
  sound(2).play(member("Ronflement"))
end

// Syntaxe JavaScript
function deactivateWindow() {
  sound(2).play(member("Ronflement"));
}
```

on DVDeventNotification

Utilisation

```
-- Syntaxe Lingo
on DVDeventNotification eventArg1 {, eventArg2} {, eventArg3}
  instruction(s)
}

// Syntaxe JavaScript
function DVDeventNotification(eventArg1 {, eventArg2} {, eventArg3}) {
  instruction(s);
}
```

Description

Gestionnaire d'événements DVD intégré. Contient des instructions exécutées en réponse aux événements qui se produisent durant la lecture d'un DVD.

Ce gestionnaire sert à gérer tous les événements DVD. L'événement qui se produit est toujours transmis en tant que premier paramètre, *eventArg1*. Certains événements contiennent des informations qui sont transmises en tant que second paramètre, *eventArg2*, et, dans certains cas, en tant que troisième paramètre, *eventArg3*.

Le tableau suivant affiche les événements qui peuvent se produire durant la lecture d'un DVD.

Evénements	Description
`angleChange`	Se produit lorsque le nombre d'angles disponibles ou le nombre d'angles de l'utilisateur courant a changé. Les informations supplémentaires suivantes sont transmises à `DVDeventNotification` lorsque cet événement se produit : • *`eventArg2`* - Un nombre entier ou une adresse indiquant le nombre d'angles disponibles. Lorsque le nombre d'angles disponibles est 1, cela signifie que la vidéo courante n'est pas multi angle. • *`eventArg3`* - Un nombre entier ou une adresse indiquant le nombre d'angles de l'utilisateur courant.
`audioStreamChange`	Se produit lorsque le numéro de flux audio de l'utilisateur courant passe au titre principal. Les informations supplémentaires suivantes sont transmises à `DVDeventNotification` lorsque cet événement se produit : • *`eventArg2`* - Un nombre entier ou une adresse indiquant le numéro du flux audio du nouvel utilisateur. Stream `0xFFFFFFFF` indique qu'aucun flux n'a été sélectionné.
`buttonChange`	Se produit lorsque le nombre de boutons disponibles ou le numéro du bouton actuellement sélectionné a changé. Les informations supplémentaires suivantes sont transmises à `DVDeventNotification` lorsque cet événement se produit : • *`eventArg2`* - Un nombre entier ou une adresse indiquant le nombre de boutons disponibles. • *`eventArg3`* - Un nombre entier ou une adresse indiquant le numéro du bouton actuellement sélectionné. Le numéro 0 indique qu'aucun bouton n'a été sélectionné.
`chapterAutoStop`	Se produit lorsque la lecture s'arrête suite à un appel à `chapterPlayAutoStop()`.
`chapterStart`	Se produit lorsque de la lecture d'un nouveau programme dans le domaine `title` commence. Les informations supplémentaires suivantes sont transmises à `DVDeventNotification` lorsque cet événement se produit : • *`eventArg2`* - Un nombre entier ou une adresse indiquant le numéro du nouveau chapitre.
`diskEjected`	Se produit lorsqu'un DVD est éjecté.
`diskInserted`	Se produit lorsqu'un DVD est inséré.

Evénements	Description
`domainChange`	Se produit lorsque le domaine du lecteur DVD change. Les informations supplémentaires suivantes sont transmises à `DVDeventNotification` lorsque cet événement se produit : • *`eventArg2`*. Une valeur indiquant le nouveau domaine. Le nouveau domaine sera l'une des valeurs suivantes. ▪ `firstPlay`. Le Navigateur DVD effectue une initialisation par défaut d'un DVD. ▪ `videoManagerMenu`. Le Navigateur DVD affiche des menus pour la totalité du disque. ▪ `videoTitleSetMenu`. Le Navigateur DVD affiche des menus pour la série de titres en cours. ▪ `title`. Le Navigateur DVD affiche le titre courant. ▪ `stop`. Le Navigateur DVD est dans le domaine `stop`.
`error`	Se produit lorsqu'une condition d'erreur DVD a lieu. Les informations supplémentaires suivantes sont transmises à `DVDeventNotification` lorsque cet événement se produit : • *`eventArg2`*. Une valeur indiquant une condition d'erreur. La condition d'erreur sera l'une des valeurs suivantes. ▪ `copyProtectFail`. Echec de l'échange de clé pour la protection de copie DVD. La lecture est interrompue. ▪ `invalidDVD1_0Disc`. Le DVD créé ne correspond pas aux spécifications de la version 1.*x*. La lecture est interrompue. ▪ `invalidDiscRegion`. Le DVD ne peut pas être lu parce qu'il n'est pas compatible avec la région système. ▪ `lowParentalLevel`. Le niveau parental du lecteur est inférieur au niveau parental le plus bas qui existe dans le contenu du DVD. La lecture est interrompue. ▪ `macrovisionFail`. Echec de la distribution Macrovision. Lecture interrompue. ▪ `incompatibleSystemAndDecoderRegions`. Aucun disque ne peut être lu car la région système ne correspond pas à la région du décodeur. ▪ `incompatibleDiscAndDecoderRegions`. Le DVD ne peut pas être lu parce qu'il n'est pas compatible avec la région du décodeur. ▪ `unexpected`. Une erreur inattendue s'est produite ; il est possible que le contenu n'ait pas été créé correctement. La lecture est interrompue.
`karaokeMode`	A lieu lorsque le mode audio est réglé sur `karaoke`.
`noFirstPlayChain`	A lieu lorsque le DVD ne possède pas de FP_PGC (First Play Program Chain) et que le Navigateur DVD ne charge pas de PGC et ne s'exécute pas automatiquement.
`parentalLevelChange`	A lieu lorsque le niveau parental du contenu créé est sur le point de changer. Les informations supplémentaires suivantes sont transmises à `DVDeventNotification` lorsque cet événement se produit : • *`eventArg2`*. Nombre entier indiquant le nouveau niveau parental défini dans le lecteur.

Evénements	Description
playbackStopped	A lieu lorsque la lecture est interrompue. Le Navigateur DVD a terminé la lecture du PGC et n'a trouvé aucune instruction concernant la poursuite de la lecture.
playPeriodAutoStop	
rateChange	A lieu lorsque la cadence de lecture change. Les informations supplémentaires suivantes sont transmises à DVDeventNotification lorsque cet événement se produit : • *eventArg2*. Nombre entier indiquant la nouvelle cadence de lecture. Une valeur inférieure à (<) 0 indique un mode de lecture inverse. Une valeur supérieure à (>) 0 indique un mode de lecture avant. Cette valeur correspond à la cadence de lecture multipliée par 10 000.
stillOff	A lieu à la fin d'une image fixe (PGC, cellule ou VOBU)
stillOn	A lieu à la fin d'une image fixe (PGC, cellule ou VOBU) Les informations supplémentaires suivantes sont transmises à DVDeventNotification lorsque cet événement se produit : • *eventArg2* - Une valeur booléenne indiquant la disponibilité des boutons. Zéro (0) indique que les boutons sont disponibles. Un (1) indique qu'aucun bouton n'est disponible. • *eventArg3* - Un nombre entier ou une adresse indiquant la durée de l'image fixe en secondes. 0xFFFFFFFF indique une image fixe infinie.
titleChange	A lieu lorsque le numéro du titre en cours change. Les informations supplémentaires suivantes sont transmises à DVDeventNotification lorsque cet événement se produit : • *eventArg2* - Un nombre entier ou une adresse indiquant le nouveau numéro du titre.
UOPchange	A lieu lorsque l'un des mécanismes de lecture ou de recherche disponibles a changé. Les informations supplémentaires suivantes sont transmises à DVDeventNotification lorsque cet événement se produit : • *eventArg2* - Un nombre entier ou une adresse indiquant les mécanismes de lecture ou de recherche explicitement désactivés par le DVD.
warning	A lieu lorsqu'une condition d'avertissement DVD est rencontrée. Les informations supplémentaires suivantes sont transmises à DVDeventNotification lorsque cet événement se produit : • *eventArg2* - Un nombre entier ou une adresse indiquant la condition d'avertissement. La condition d'avertissement sera l'une des valeurs suivantes : ▪ invalidDVD1_0Disc. Le DVD n'a pas été correctement créé. La lecture peut continuer, mais un comportement inattendu risque de se produire. ▪ formatNotSupported. Un décodeur ne supporte pas le format courant. La lecture d'un flux risque de ne pas fonctionner. ▪ illegalNavCommand. Le processeur de commande de navigation DVD interne a essayé de traiter une commande illégale. ▪ open. ▪ seek. ▪ read.

Voir aussi

DVD

on endSprite

Utilisation

```
-- Syntaxe Lingo
on endSprite
  instruction(s)
end

// Syntaxe JavaScript
function endSprite() {
  instruction(s);
}
```

Description

Message système et gestionnaire d'événement ; contient des éléments Lingo exécutés lorsque la tête de lecture sort d'une image-objet et passe à une image dans laquelle l'image-objet n'existe pas. Il est généré après `exitFrame`.

Placez les gestionnaires `on endSprite` dans un script de comportement.

Director détruit les instances des scripts de comportement liés à l'image-objet immédiatement après l'événement `endSprite`.

Le gestionnaire d'événement reçoit la référence de comportement ou de script d'image `me` lorsqu'il est utilisé dans un comportement. Ce message `endSprite` est envoyé après le message `exitFrame` si la tête de lecture continue jusqu'à la fin de l'image.

Les méthodes `go()`, `play()` et `updateStage()` sont désactivées dans un gestionnaire `on endSprite`.

Exemple

Le gestionnaire suivant est exécuté lorsque la tête de lecture quitte une image-objet :

```
-- Syntaxe Lingo
on endSprite me
  -- nettoyage
  gNombreDeRequins = gNombreDeRequins - 1
  sound(5).stop()
end

// Syntaxe JavaScript
function endSprite() {
  // nettoyage
  gNombreDeRequins--;
  sound(5).stop();
}
```

Voir aussi

on beginSprite, on exitFrame

on enterFrame

Utilisation

```
-- Syntaxe Lingo
on enterFrame
  instruction(s)
end

// Syntaxe JavaScript
function enterFrame() {
  instruction(s);
}
```

Description

Message système et gestionnaire d'événement ; contient des instructions exécutées à chaque fois que la tête de lecture entre dans l'image.

Placez les gestionnaires `on enterFrame` dans un script de comportement, d'image ou d'animation, comme suit :

- Pour affecter le gestionnaire à une seule image-objet, placez-le dans un comportement lié à l'image-objet.
- Pour affecter le gestionnaire à une seule image, placez-le dans le script de l'image.
- Pour affecter le gestionnaire à chaque image (à moins que vous n'indiquiez explicitement le contraire), placez le gestionnaire `on enterFrame` dans un script d'animation. Le gestionnaire est exécuté à chaque fois que la tête de lecture entre dans une image à moins que le script d'image n'ait son propre gestionnaire. Si le script d'image a son propre gestionnaire, le gestionnaire `on enterFrame` du script d'image prend priorité sur le gestionnaire `on enterFrame` du script d'animation.

L'ordre des événements d'images est `stepFrame`, `prepareFrame`, `enterFrame` et `exitFrame`.

Cet événement reçoit la référence d'objet `me` lorsqu'il est utilisé dans un comportement.

Exemple

Le gestionnaire suivant désactive la condition d'asservissement pour les images 1 à 5 à chaque fois que la tête de lecture entre dans l'image :

```
-- Syntaxe Lingo
on enterFrame
  repeat with i = 1 to 5
    _movie.puppetSprite(i, FALSE)
  end repeat
end

// Syntaxe JavaScript
function enterFrame() {
  for (i=1;i<=5;i++) {
    _movie.puppetSprite(i, false);
  }
}
```

on EvalScript

Utilisation

```
-- Syntaxe Lingo
on EvalScriptunParamètre
  instruction(s)
end

// Syntaxe JavaScript
function EvalScript(unParamètre) {
  instruction(s);
}
```

Description

Message système et gestionnaire d'événement ; dans une animation à contenu Macromedia Shockwave, renferme des instructions exécutées lorsque le gestionnaire reçoit un message `EvalScript` d'un navigateur. Le paramètre est une chaîne reçue du navigateur web.

- Le message `EvalScript` peut inclure une chaîne que Director peut interpréter comme une instruction Lingo. Lingo n'accepte pas de chaînes imbriquées. Si le gestionnaire que vous appelez attend une chaîne comme paramètre, passez le paramètre comme symbole.
- Le gestionnaire `on EvalScript` est appelé par la méthode JavaScript ou VBScript `EvalScript()` dans un navigateur web.

Seuls les comportements devant être contrôlés par les utilisateurs doivent être inclus dans `on EvalScript`; pour des raisons de sécurité, ne donnez pas d'accès complet à tous les comportements.

Remarque : Si vous placez un mot-clé return à la fin de votre gestionnaire EvalScript, la valeur renvoyée peut être utilisée par JavaScript dans le navigateur web.

Exemple

L'exemple suivant indique comment faire passer la tête de lecture à une image spécifique en fonction de l'image passée comme paramètre :

```
-- Syntaxe Lingo
on EvalScript unParamètre
  _movie.go(unParamètre)
end

// Syntaxe JavaScript
function EvalScript(unParamètre) {
  _movie.go(unParamètre);
}
```

Le gestionnaire suivant exécute l'instruction `_movie.go(unParamètre)` s'il reçoit un message `EvalScript` contenant l'argument chien, chat ou arbre :

```
-- Syntaxe Lingo
on EvalScript unParamètre
  case unParamètre of
    "chien", "chat", "arbre": _movie.go(unParamètre)
  end case
end
```

```
// Syntaxe JavaScript
function EvalScript(unParamètre) {
  switch(unParamètre) {
    case "chien", "chat", "arbre": _movie.go(unParamètre);
  }
}
```

Une instruction d'appel possible dans JavaScript serait `EvalScript ("chien")`.

Le gestionnaire suivant prend un argument qui pourrait être un nombre ou un symbole :

```
-- Syntaxe Lingo
on EvalScript unParamètre
  if word 1 of unParamètre = "monGestionnaire" then
    _movie.go(unParamètre)
  end if
end

// Syntaxe JavaScript
function EvalScript(unParamètre) {
  if (unParamètre.indexOf("monGestionnaire",0)) {
    _movie.go(unParamètre);
  }
}
```

Le gestionnaire suivant nécessite normalement une chaîne comme argument. L'argument est reçu comme un symbole, puis converti en chaîne dans le gestionnaire par la fonction `string` :

```
-- Syntaxe Lingo
on monGestionnaire unParamètre
  _movie.go(chaîne(unParamètre))
end

// Syntaxe JavaScript
fonction monGestionnaire(unParamètre {
  _movie.go(unParamètre.toString());
}
```

Voir aussi

`externalEvent()`, `return (mot-clé)`

on exitFrame

Utilisation

```
-- Syntaxe Lingo
on exitFrame
  instruction(s)
end

// Syntaxe JavaScript
function exitFrame() {
  instruction(s);
}
```

Description

Message système et gestionnaire d'événement ; contient des instructions exécutées à chaque fois que la tête de lecture quitte l'image à laquelle le gestionnaire `on exitFrame` est lié. Le gestionnaire `on exitFrame` est pratique pour placer des instructions permettant de réinitialiser des conditions inutiles une fois sorti de l'image.

Placez les gestionnaires `on exitFrame` dans des scripts de comportements, d'image ou d'animation de la façon suivante :

- Pour affecter le gestionnaire à une seule image-objet, placez-le dans un comportement lié à l'image-objet.
- Pour affecter le gestionnaire à une seule image, placez-le dans le script de l'image.
- Pour affecter le gestionnaire à chaque image (à moins que vous n'indiquiez explicitement le contraire), placez le gestionnaire dans un script d'animation. Le gestionnaire `on exitFrame` est exécuté à chaque fois que la tête de lecture quitte l'image, à moins que le script d'image n'ait son propre gestionnaire `on exitFrame`. Le cas échéant, le gestionnaire `on exitFrame` du script d'image prend priorité sur celui du script d'animation.

Lorsque utilisé dans un comportement, cet événement reçoit la référence `me` du script d'image-objet ou d'image. L'ordre des événements d'image est `prepareFrame`, `enterFrame` et `exitFrame`.

Exemple

Le gestionnaire suivant désactive toutes les conditions d'asservissement lorsque la tête de lecture quitte l'image :

```
-- Syntaxe Lingo
on exitFrame me
  repeat with i = 48 down to 1
    sprite(i).scripted = FALSE
  end repeat
end

// Syntaxe JavaScript
function exitFrame() {
  for (i=48; i>=1; i--);
    sprite(i).scripted = false;
  }
}
```

Le gestionnaire suivant déplace la tête de lecture vers une image spécifiée si la valeur de la variable globale `vTotal` dépasse 1 000 lorsque la tête de lecture quitte l'image :

```
// Syntaxe JavaScript
function exitFrame() {
  if (_global.vTotal > 1000) {
    _movie.go("Terminé");
  }
}
```

Voir aussi

`on enterFrame`

on getBehaviorDescription

Utilisation

```
-- Syntaxe Lingo
on getBehaviorDescription
  instruction(s)
end

// Syntaxe JavaScript
function getBehaviorDescription() {
  instruction(s);
}
```

Description

Message système et gestionnaire d'événement; contient des instructions renvoyant la chaîne apparaissant dans le volet de description de l'inspecteur de comportement lorsque le comportement est sélectionné.

La chaîne de description est facultative.

Director envoie le message `getBehaviorDescription` aux comportements liés à une image-objet à l'ouverture de l'inspecteur de comportement. Placez le gestionnaire `on getBehaviorDescription` dans un comportement.

Le gestionnaire peut contenir des caractères de retour de chariot pour formater les descriptions multilignes.

Exemple

L'instruction suivante affiche « Barre de défilement vertical de champ de texte multiligne » dans le volet de description :

```
-- Syntaxe Lingo
on getBehaviorDescription
  return "Barre de défilement vertical de champ de texte multiligne"
end

// Syntaxe JavaScript
function getBehaviorDescription() {
  return "Barre de défilement vertical de champ de texte multiligne";
}
```

Voir aussi

`on getPropertyDescriptionList`, `on getBehaviorTooltip`, `on runPropertyDialog`

on getBehaviorTooltip

Utilisation

```
-- Syntaxe Lingo
on getBehaviorTooltip
  instruction(s)
end

// Syntaxe JavaScript
function getBehaviorTooltip() {
  instruction(s);
}
```

Description

Message système et gestionnaire d'événement ; contient des instructions renvoyant la chaîne apparaissant dans une info-bulle pour un script de la palette des bibliothèques.

Director envoie le message `getBehaviorTooltip` au script lorsque le curseur s'arrête au-dessus de ce dernier dans la palette des bibliothèques. Placez le gestionnaire `on getBehaviorTooltip` dans le comportement.

L'utilisation du gestionnaire est facultative. Si aucun gestionnaire n'est fourni, le nom de l'acteur apparaît dans l'info-bulle.

Le gestionnaire peut contenir des caractères de retour de chariot pour formater les descriptions multilignes.

Exemple

L'instruction suivante affiche Pièce de puzzle dans le volet de description :

```
-- Syntaxe Lingo
on getBehaviorTooltip
  return "Pièce de puzzle"
end

// Syntaxe JavaScript
function getBehaviorTooltip() {
  return "Pièce de puzzle";
}
```

Voir aussi

`on getPropertyDescriptionList`, `on getBehaviorDescription`, `on runPropertyDialog`

on getPropertyDescriptionList

Utilisation

```
-- Syntaxe Lingo
on getPropertyDescriptionList
  instruction(s)
end

// Syntaxe JavaScript
fonction getPropertyDescriptionList() {
  instruction(s);
}
```

Description

Message système et gestionnaire d'événement ; contient des instructions générant une liste de définitions et de libellés pour les paramètres qui apparaissent dans la boîte de dialogue Paramètres d'un comportement.

Placez le gestionnaire `on getPropertyDescriptionList` dans un script de comportement. Les comportements ne contenant pas de gestionnaire `on getPropertyDescriptionList` n'apparaissent pas dans la boîte de dialogue Paramètres et ne peuvent pas être modifiés à partir de l'interface de Director.

Le message `on getPropertyDescriptionList` est envoyé lorsqu'une action entraînant l'ouverture de l'inspecteur de comportement a lieu : soit lorsque l'utilisateur fait glisser un comportement sur le scénario, soit lorsqu'il double-clique sur un comportement dans l'inspecteur de comportement.

Les paramètres `#default`, `#format` et `#comment` sont obligatoires pour chaque paramètre. Les valeurs possibles de ces paramètres sont :

`#default`	Valeur initiale du paramètre.
`#format`	`#integer #float #string #symbol #member #bitmap #filmloop #field #palette #picture #sound #button #shape #movie #digitalvideo #script #richtext #ole #transition #xtra #frame #marker #ink #boolean`
`#comment`	Chaîne descriptive apparaissant à gauche du champ modifiable dans la boîte de dialogue Paramètres.
`#range`	Fourchette de valeurs possibles pouvant être affectées à une propriété. Cette fourchette est spécifiée sous la forme d'une liste linéaire avec plusieurs valeurs ou comme un minimum et maximum sous la forme d'une liste de propriétés : `[#min: valeurMin, #max: valeurMax]`.

Exemple

Le gestionnaire suivant définit les paramètres d'un comportement apparaissant dans la boîte de dialogue Paramètres. Chaque instruction commençant par `addProp` ajoute un paramètre à la liste appelée Description. Chaque élément ajouté à la liste définit une propriété et ses valeurs `#default`, `#format` et `#comment` :

```
on getPropertyDescriptionList
  description = [:]
  adescription.addProp(#dynamic, [#default:1, #format:#boolean, \
  #comment:"Dynamique"])
  description.addProp(#fieldNum, [#default:1, #format:#integer, \
  #comment:"Image-objet à faire défiler:"])
  description.addProp(#extentSprite,[#default:1,#format:#integer, \
  #comment: "Etendre l'image-objet:"])
  description.addProp(#proportional, [#default:1, #format:#boolean, \
  #comment: "Proportionnel:"])
  return description
end
```

Voir aussi

`addProp`, `on getBehaviorDescription`, `on runPropertyDialog`

on hyperlinkClicked

Utilisation

```
-- Syntaxe Lingo
on hyperlinkClicked me, données, plage
  instruction(s)
end

// Syntaxe JavaScript
function hyperlinkClicked(données, plage){
  instruction(s);
}
```

Description

Message système et gestionnaire d'événement ; utilisé pour déterminer si l'utilisateur a cliqué sur un lien hypertexte.

Ce gestionnaire d'événement comprend les paramètres suivants :

- `me` Utilisé dans un comportement pour identifier l'instance d'image-objet.
- `données` Les données du lien hypertexte même, la chaîne saisie dans l'inspecteur de texte lors de la modification de l'acteur texte.
- `plage` La plage de caractères du lien hypertexte dans le texte (il est possible d'obtenir le texte de la plage même en utilisant la syntaxe `réfActeur.char[plage[1]..plage[2]]`.

Ce gestionnaire doit être attaché à une image-objet en tant que script de comportement. Evitez de placer ce gestionnaire dans un script d'acteur.

Exemple

Le comportement suivant affiche un lien utilisé pour examiner le lien hypertexte sélectionné, passer à une URL si nécessaire, puis afficher le texte du lien dans la fenêtre Messages :

```
-- Syntaxe Lingo
property spriteNum

on hyperlinkClicked(me, données, plage)
  if data starts "http://" then
    gotoNetPage(données)
  end if
  acteurCourant = sprite(spriteNum).member
  chaîneDancre = acteurCourant.char[plage[1]..plage[2]]
  put("Le lien hypertexte"&&chaîneDancre&&"vient d'être activé.")
end
// Syntaxe JavaScript
function hyperlinkClicked(données, plage) {
  var st = data.slice(0,7);
  var ht = "http://";
  if (st = ht) {
    gotoNetPage(données) ;
  }
  var acteurCourant = sprite(this.spriteNum).member;
  var r1 = acteurCourant.getPropRef("car", plage[1]).hyperlinkRange;
  var a = r1[1] - 1;
  var b = r1[2];
  var st = new String(acteurCourant.text);
  var chaîneDancre = st.slice(a, b);
  put("Le lien hypertexte" + chaîneDancre + " vient d'être activé.");
}
```

on idle

Utilisation

```
-- Syntaxe Lingo
on idle
  instruction(s)
end

// Syntaxe JavaScript
function idle() {
  instruction(s);
}
```

Description

Message système et gestionnaire d'événement ; contient des instructions exécutées lorsque l'animation n'a pas d'autres événements à traiter et constitue un endroit pratique pour placer les instructions à exécuter aussi fréquemment que possible, comme celles qui mettent à jour les valeurs des variables globales et affichent les conditions de l'animation courante.

Les instructions des gestionnaires `on idle` étant fréquemment exécutées, il est recommandé de ne pas placer d'instruction Lingo dont le traitement prend trop de temps dans un gestionnaire `on idle`.

Il est souvent préférable de placer les gestionnaires `on idle` dans des scripts d'image plutôt que dans des scripts d'animation pour tirer parti du gestionnaire `on idle` uniquement lorsque nécessaire.

Director peut charger des acteurs à partir d'une distribution interne ou externe pendant un événement `idle`. Cependant, il ne peut pas charger des acteurs liés pendant un événement `idle`.

Le message `idle` est envoyé uniquement aux scripts d'image et d'animation.

Exemple

Le gestionnaire suivant met à jour l'heure affichée dans l'animation lorsqu'il n'y a aucun autre événement à traiter :

```
-- Syntaxe Lingo
on idle
  member("Heure").text = _system.time()
end idle

// Syntaxe JavaScript
function idle() {
  member("Heure").text = _system.time();
}
```

Voir aussi

`idleHandlerPeriod`

on isOKToAttach

Utilisation

```
-- Syntaxe Lingo
on isOKToAttach me, typeDimageObjet, numéroDimageObjet
  instruction(s)
end

// Syntaxe JavaScript
function isOKToAttach(typeDimageObjet, numéroDimageObjet) {
  instruction(s)
}
```

Description

Gestionnaire intégré ; vous pouvez ajouter ce gestionnaire à un comportement en vue de vérifier le type d'image-objet auquel le comportement est associé et empêcher que ce comportement soit attaché à des types d'images-objets non appropriés.

Lorsque le comportement est associé à une image-objet, le gestionnaire est exécuté et Director lui transmet le type et le numéro de l'image-objet. L'argument `me` contient une référence au comportement associé à l'image-objet.

Ce gestionnaire est exécuté avant le gestionnaire `on getPropertyDescriptionList`.

L'auteur Lingo peut vérifier deux types d'images-objets. `#graphic` inclut tous les acteurs graphique tels que les formes, les bitmaps, les vidéos numériques, le texte, etc. `#script` indique le comportement associé à la piste des scripts. Dans ce cas, *`numéroDimageObjet`* est 1.

Pour chacun de ces types d'images-objets, le gestionnaire doit renvoyer la valeur `TRUE` ou `FALSE`. La valeur `TRUE` indique que le comportement peut être associé à l'image-objet. La valeur `FALSE` empêche l'association du comportement à l'image-objet.

Si le comportement ne contient pas de gestionnaire `on isOKToAttach`, le comportement peut être associé à n'importe quelle image ou image-objet.

Ce gestionnaire est appelé lors de l'association initiale du comportement à l'image-objet ou à la piste des scripts et lors de l'association d'un nouveau comportement à une image-objet à l'aide de l'inspecteur de comportement.

Exemple

L'instruction suivante vérifie le type d'image-objet à laquelle le comportement est associé et renvoie la valeur `TRUE` pour toutes les images-objets graphique à l'exception des formes et la valeur `FALSE` pour la piste des scripts :

```
-- Syntaxe Lingo
on isOKToAttach me, typeDimageObjet, numéroDimageObjet
  case typeDimageObjet of
    #graphic: -- tout type d'image-objet graphique
      return sprite(numéroDimageObjet).member.type <> #shape
      -- fonctionne avec tout, sauf les acteurs forme
    #script: --piste des scripts de l'image
      return FALSE -- ne fonctionne pas comme script d'image
  end case
end

// Syntaxe JavaScript
function isOKToAttach(typeDimageObjet, numéroDimageObjet) {
  switch (typeDimageObjet) {
    case symbol("graphique"): // tout type d'image-objet graphique
      return sprite(numéroDimageObjet).member.type != symbol("forme");
      // fonctionne avec tout, sauf les acteurs forme
    case symbol("script"): // piste des scripts de l'image
      return false; // ne fonctionne pas comme script d'image
  }
}
```

on keyDown

Utilisation

```
-- Syntaxe Lingo
on keyDown
  instruction(s)
end

// Syntaxe JavaScript
function keyDown() {
  instruction(s);
}
```

Description

Message système et gestionnaire d'événement ; contient les instructions exécutées lorsqu'une touche est enfoncée.

Lorsqu'une touche est enfoncée, Director recherche un gestionnaire `on keyDown` aux emplacements suivants et dans l'ordre qui suit : gestionnaire d'événement principal, script d'image-objet champ modifiable, script d'acteur champ, script d'image et script d'animation. Pour les images-objets et les acteurs, les gestionnaires `on keyDown` ne fonctionnent qu'avec les acteurs texte et champ modifiables. Un événement `keyDown` n'a aucun effet sur les autres types d'acteurs (acteurs bitmap, par exemple). Si le fait d'appuyer sur une touche doit produire un résultat identique dans toute l'animation, définissez `keyDownScript`.

Director arrête la recherche dès qu'il rencontre le premier gestionnaire `on keyDown`, sauf si celui-ci contient une commande `pass` exigeant le renvoi du message `keyDown` à l'emplacement suivant.

Le gestionnaire d'événement `on keyDown` est idéal pour placer des instructions créant des raccourcis clavier ou d'autres fonctions d'interface, en fonction des actions que vous souhaitez déclencher lorsque l'utilisateur appuie sur une touche du clavier.

Lorsque l'animation est lue sous forme d'applet, un gestionnaire `on keyDown` intercepte toujours les touches enfoncées, même s'il est vide. Lorsque l'utilisateur saisit du texte dans un champ modifiable, le gestionnaire `on keyDown` de ce champ doit inclure la commande `pass` pour que les caractères saisis apparaissent dans le champ.

L'endroit où vous placez un gestionnaire `on keyDown` peut affecter le moment où il est exécuté.

- Pour appliquer le gestionnaire à une image-objet champ modifiable particulière, placez-le dans un script d'image-objet.
- Pour appliquer le gestionnaire à un acteur champ modifiable quelconque, placez-le dans un script d'acteur.
- Pour appliquer le gestionnaire à une image entière, placez-le dans un script d'image.
- Pour appliquer le gestionnaire à l'animation entière, placez-le dans un script d'animation.

Vous pouvez annuler un gestionnaire `on keyDown` en plaçant un autre gestionnaire `on keyDown` dans une position qui sera vérifiée par Lingo avant celle du gestionnaire à annuler. Par exemple, vous pouvez annuler un gestionnaire `on keyDown` affecté à un acteur en plaçant un gestionnaire `on keyDown` dans un script d'image-objet.

Exemple

Le gestionnaire suivant vérifie si la touche Retour a été enfoncée et, le cas échéant, fait passer la tête de lecture à une autre image :

```
-- Syntaxe Lingo
on keyDown
  if (_key.key = RETURN) then _movie.go("additionnerLaSomme")
end keyDown

// Syntaxe JavaScript
function keyDown() {
  if (_key.keyCode == 36) {
    _movie.go("additionnerLaSomme");
  }
}
```

Voir aussi

`charToNum()`, `keyDownScript`, `keyUpScript`, `key`, `keyCode`, `keyPressed()`

on keyUp

Utilisation

```
-- Syntaxe Lingo
on keyUp
  instruction(s)
end

// Syntaxe JavaScript
function keyUp() {
  instruction(s);
}
```

Description

Message système et gestionnaire d'événement ; contient des instructions exécutées lorsque l'utilisateur relâche une touche. Le gestionnaire `on keyUp` est similaire au gestionnaire `on keyDown`, à ceci près que l'événement se produit après l'apparition d'un caractère si l'image-objet champ ou texte est modifiable à l'écran.

Lorsque l'utilisateur relâche une touche, Lingo recherche un gestionnaire `on keyUp` aux emplacements suivants et dans l'ordre qui suit : gestionnaire d'événement principal, script d'image-objet champ modifiable, script d'acteur champ, script d'image et script d'animation. Pour les images-objets et les acteurs, les gestionnaires `on keyUp` ne fonctionnent qu'avec les chaînes modifiables. Un événement `keyUp` n'a aucun effet sur les autres types d'acteurs (acteurs bitmap, par exemple). Si le fait de relâcher une touche doit produire un résultat identique dans toute l'animation, définissez `keyUpScript`.

Lingo arrête la recherche dès qu'il rencontre le premier gestionnaire `on keyUp`, sauf si celui-ci contient une commande `pass` exigeant le renvoi du message `keyUp` au gestionnaire ou script suivant.

Le gestionnaire `on keyUp` est idéal pour placer des instructions créant des raccourcis clavier ou d'autres fonctions d'interface, en fonction des actions que vous souhaitez déclencher lorsque l'utilisateur relâche une touche du clavier.

Lorsque l'animation est lue sous forme d'applet, un gestionnaire `on keyUp` enregistre toujours les touches enfoncées, même s'il est vide. Lorsque l'utilisateur saisit du texte dans un champ modifiable, le gestionnaire `on keyUp` de ce champ doit inclure la commande `pass` pour que les caractères saisis apparaissent dans le champ.

L'endroit où vous placez un gestionnaire `on keyUp` affecte le moment où il est exécuté, tel que :

- Pour appliquer le gestionnaire à une image-objet champ modifiable spécifique, placez-le dans un comportement.
- Pour appliquer le gestionnaire à un acteur champ modifiable quelconque, placez-le dans un script d'acteur.
- Pour appliquer le gestionnaire à une image entière, placez-le dans un script d'image.
- Pour appliquer le gestionnaire à l'animation entière, placez-le dans un script d'animation.

Vous pouvez annuler un gestionnaire `on keyUp` en plaçant un autre gestionnaire `on keyUp` dans une position qui sera vérifiée par Lingo avant celle du gestionnaire à annuler. Par exemple, vous pouvez annuler le gestionnaire `on keyUp` affecté à un acteur en plaçant un gestionnaire `on keyUp` dans un script d'image-objet.

Exemple

Le gestionnaire suivant vérifie si la touche Retour a été relâchée et, le cas échéant, fait passer la tête de lecture à une autre image :

```
-- Syntaxe Lingo
on keyUp
  if (_key.key = RETURN) then _movie.go("additionnerLaSomme")
end keyUp

// Syntaxe JavaScript
function keyUp() {
  if (_key.keyCode == 36) {
    _movie.go("additionnerLaSomme");
  }
}
```

Voir aussi

on keyDown, keyDownScript, keyUpScript

on mouseDown (gestionnaire d'événement)

Utilisation

```
-- Syntaxe Lingo
on mouseDown
  instruction(s)
end

// Syntaxe JavaScript
function mouseDown() {
  instruction(s);
}
```

Description

Message système et gestionnaire d'événement; contient des instructions exécutées lorsque le bouton de la souris est enfoncé.

Lorsque le bouton de la souris est enfoncé, Lingo recherche un gestionnaire `on mouseDown` aux emplacements suivants et dans l'ordre qui suit : gestionnaire d'événement principal, script d'image-objet, script d'acteur, script d'image et script d'animation. Lingo arrête la recherche lorsqu'il atteint le premier emplacement contenant un gestionnaire `on mouseDown`, sauf si ce dernier contient une commande `pass` permettant de passer explicitement le message `mouseDown` à l'emplacement suivant.

Pour que l'animation réponde toujours de la même manière lorsque le bouton de la souris est enfoncé, définissez `mouseDownScript` ou placez un gestionnaire mouseDown dans un script de l'animation.

Les gestionnaires d'événements `on mouseDown` sont pratiques pour placer des instructions faisant clignoter des images, déclenchant des effets sonores ou provoquant le déplacement d'images-objets lorsque l'utilisateur appuie sur le bouton de la souris.

L'endroit où vous placez un gestionnaire `on mouseDown` peut affecter le moment où il est exécuté.

- Pour appliquer le gestionnaire à une image-objet spécifique, placez-le dans un script d'image-objet.
- Pour appliquer le gestionnaire à un acteur quelconque, placez-le dans un script d'acteur.

- Pour appliquer le gestionnaire à une image entière, placez-le dans un script d'image.
- Pour appliquer le gestionnaire à l'animation entière, placez-le dans un script d'animation.

Vous pouvez annuler un gestionnaire `on mouseDown` en plaçant un autre gestionnaire `on mouseDown` dans une position qui sera vérifiée par Lingo avant celle du gestionnaire à annuler. Par exemple, vous pouvez annuler un gestionnaire `on mouseDown` affecté à un acteur en plaçant un gestionnaire `on mouseDown` dans le script d'une image-objet.

Lorsqu'il est utilisé dans un comportement, cet événement reçoit en paramètre la référence au script de l'image-objet ou au script d'image `me`.

Exemple

Le gestionnaire suivant vérifie si l'utilisateur clique à un endroit quelconque de la scène et, le cas échéant, fait passer la tête de lecture à une autre image :

```
-- Syntaxe Lingo
on mouseDown
  if (_mouse.clickOn = 0) then _movie.go("additionnerLaSomme")
end

// Syntaxe JavaScript
function mouseDown() {
  if (_mouse.clickOn == 0) {
    _movie.go("additionnerLaSomme");
  }
}
```

Le gestionnaire suivant, affecté à un script d'image-objet, lit un son lorsque l'utilisateur clique sur l'image-objet :

```
-- Syntaxe Lingo
on mouseDown
  sound(1).play(member("Grillons"))
end

// Syntaxe JavaScript
function mouseDown() {
  sound(1).play(member("Grillons"));
}
```

Voir aussi

`clickOn`, `mouseDownScript`, `mouseUpScript`

on mouseEnter

Utilisation

```
-- Syntaxe Lingo
on mouseEnter
  instruction(s)
end

// Syntaxe JavaScript
function mouseEnter() {
  instruction(s);
}
```

Description

Message système et gestionnaire d'événement ; contient les instructions exécutées lorsque le pointeur de la souris entre en contact avec la zone référencée de l'image-objet. Il n'est pas nécessaire que le bouton de la souris soit enfoncé.

Si l'image-objet est un acteur bitmap dont l'encre est Dessin seul, sa zone référencée correspond à la portion d'image affichée. Dans tous les autres cas, elle correspond au rectangle de délimitation de l'image-objet.

Lorsqu'il est utilisé dans un comportement, cet événement reçoit en paramètre la référence au script de l'image-objet ou au script d'image `me`.

Exemple

L'instruction suivante est un comportement de bouton simple qui change l'image bitmap du bouton lorsque la souris survole, puis sort du bouton :

```
-- Syntaxe Lingo
property spriteNum

on mouseEnter me
  -- détermine l'acteur courant et passe à l'acteur suivant de la
  distribution.
  acteurCourant = sprite(spriteNum).member.number
  sprite(spriteNum).member = acteurCourant + 1
end

on mouseLeave me
  -- détermine l'acteur courant et passe à l'acteur suivant de la
  distribution.
  acteurCourant = sprite(spriteNum).member.number
  sprite(spriteNum).member = acteurCourant -1
end
```

```
// Syntaxe JavaScript
var spriteNum;

function mouseEnter() {
  // détermine l'acteur courant et passe à l'acteur suivant de la
  distribution.
  acteurCourant = sprite(spritenum).member.number;
  sprite(spritenum).member = acteurCourant +1;
}

function mouseLeave() {
  // détermine l'acteur courant et passe à l'acteur précédent de la
  distribution.
  acteurCourant = sprite(spritenum).member.number;
  sprite(spritenum).member = acteurCourant - 1;
}
```

Voir aussi

on mouseLeave, on mouseWithin

on mouseLeave

Utilisation

```
-- Syntaxe Lingo
on mouseLeave
  instruction(s)
end

// Syntaxe JavaScript
function mouseLeave() {
  instruction(s);
}
```

Description

Message système et gestionnaire d'événement ; contient les instructions exécutées lorsque la souris sort de la zone référencée de l'image-objet. Il n'est pas nécessaire que le bouton de la souris soit enfoncé.

Si l'image-objet est un acteur bitmap dont l'encre est Dessin seul, sa zone référencée correspond à la portion d'image affichée. Dans tous les autres cas, elle correspond au rectangle de délimitation de l'image-objet.

Lorsqu'il est utilisé dans un comportement, cet événement reçoit en paramètre la référence au script de l'image-objet ou au script d'image `me`.

Exemple

L'instruction suivante est un comportement de bouton simple qui change l'image bitmap du bouton lorsque la souris survole le bouton, puis en sort :

```
-- Syntaxe Lingo
property spriteNum

on mouseEnter me
  -- détermine l'acteur courant et passe à l'acteur suivant de la
  distribution.
  acteurCourant = sprite(spriteNum).member.number
  sprite(spriteNum).member = acteurCourant + 1
```

```
end

on mouseLeave me
  -- détermine l'acteur courant et passe à l'acteur suivant de la
  distribution.
  acteurCourant = sprite(spriteNum).member.number
  sprite(spriteNum).member = acteurCourant -1
end

// Syntaxe JavaScript
var spriteNum;

function mouseEnter() {
  // détermine l'acteur courant et passe à l'acteur suivant de la
  distribution.
  acteurCourant = sprite(spritenum).member.number;
  sprite(spritenum).member = acteurCourant +1;
}

function mouseLeave() {
  // détermine l'acteur courant et passe à l'acteur précédent de la
  distribution.
  acteurCourant = sprite(spritenum).member.number;
  sprite(spritenum).member = acteurCourant - 1;
}
```

Voir aussi

on mouseEnter, on mouseWithin

on mouseUp (gestionnaire d'événement)

Utilisation

```
-- Syntaxe Lingo
on mouseUp
  instruction(s)
end

// Syntaxe JavaScript
function mouseUp() {
  instruction(s);
}
```

Description

Message système et gestionnaire d'événement ; contient les instructions exécutées lorsque le bouton de la souris est relâché.

Lorsque l'utilisateur relâche le bouton de la souris, Lingo recherche un gestionnaire `on mouseUp` aux emplacements suivants et dans l'ordre qui suit : gestionnaire d'événement principal, script d'image-objet, script d'acteur, script d'image et script d'animation. Lingo arrête la recherche dès qu'il rencontre le premier gestionnaire `on mouseUp`, sauf si celui-ci contient une commande `pass` permettant de passer explicitement le message `mouseUp` à l'emplacement suivant.

Pour obtenir la même réponse dans toute l'animation lorsque l'utilisateur relâche le bouton de la souris, définissez `the mouseUpScript`.

Le gestionnaire d'événement `on mouseUp` est pratique pour appeler un script Lingo modifiant l'apparence des objets (boutons, par exemple) une fois que l'utilisateur a cliqué dessus. Pour ce faire, il suffit de changer l'acteur de l'image-objet sur laquelle l'utilisateur a cliqué, dès qu'il relâche le bouton de la souris.

L'endroit où vous placez un gestionnaire `on mouseUp` affecte le moment où il est exécuté, de la manière suivante :

- Pour appliquer le gestionnaire à une image-objet spécifique, placez-le dans un script d'image-objet.
- Pour appliquer le gestionnaire à un acteur quelconque, placez-le dans un script d'acteur.
- Pour appliquer le gestionnaire à une image entière, placez-le dans un script d'image.
- Pour appliquer le gestionnaire à l'animation entière, placez-le dans un script d'animation.

Vous pouvez annuler un gestionnaire `on mouseUp` en plaçant un autre gestionnaire `on mouseUp` dans une position qui sera vérifiée par Lingo avant celle du gestionnaire à annuler. Par exemple, vous pouvez annuler un gestionnaire `on mouseUp` affecté à un acteur en plaçant un gestionnaire `on mouseUp` dans un script d'image-objet.

Lorsqu'il est utilisé dans un comportement, cet événement reçoit en paramètre la référence au script de l'image-objet ou au script d'image `me`.

Exemple

Le gestionnaire suivant, affecté à l'image-objet 10, change l'acteur associé à cette image-objet à chaque fois que l'utilisateur relâche le bouton de la souris après avoir cliqué sur l'image-objet :

```
-- Syntaxe Lingo
on mouseUp
  sprite(10).member = member ("Gris")
end

// Syntaxe JavaScript
function mouseUp() {
  sprite(10).member = member ("Gris");
}
```

Voir aussi

on mouseDown (gestionnaire d'événement)

on mouseUpOutside

Utilisation

```
-- Syntaxe Lingo
on mouseUpOutside me
  instruction(s)
end

// Syntaxe JavaScript
function mouseUpOutside() {
  instruction(s);
}
```

Description

Message système et gestionnaire d'événement; envoyé lorsque l'utilisateur clique sur une image-objet, puis relâche le bouton de la souris en dehors de cette dernière.

Exemple

L'instruction suivante lit un son lorsque l'utilisateur clique sur une image-objet, puis relâche le bouton de la souris en dehors du rectangle de délimitation de cette dernière :

```
-- Syntaxe Lingo
on mouseUpOutside me
  sound(1).play(member("Professeur Machin"))
end

// Syntaxe JavaScript
function mouseUpOutside() {
  sound(1).play(member("Professeur Machin"));
}
```

Voir aussi

on mouseEnter, on mouseLeave, on mouseWithin

on mouseWithin

Utilisation

```
-- Syntaxe Lingo
on mouseWithin
  instruction(s)
end

// Syntaxe JavaScript
function mouseWithin() {
  instruction(s);
}
```

Description

Message système et gestionnaire d'événement; contient les instructions exécutées lorsque la souris est dans la zone référencée de l'image-objet. Il n'est pas nécessaire que le bouton de la souris soit enfoncé.

Si l'image-objet est un acteur bitmap dont l'encre est Dessin seul, sa zone référencée correspond à la partie d'image affichée. Dans tous les autres cas, elle correspond au rectangle de délimitation de l'image-objet.

Lorsqu'il est utilisé dans un comportement, cet événement reçoit en paramètre la référence au script de l'image-objet ou au script d'image `me`.

Exemple

L'instruction suivante affiche la position de la souris lorsqu'elle se trouve au-dessus d'une image-objet :

```
-- Syntaxe Lingo
on mouseWithin
  member("Affichage").text = string(_mouse.mouseH)
end

// Syntaxe JavaScript
function mouseWithin() {
  member("Affichage").text = _mouse.mouseH.toString();
}
```

Voir aussi

on mouseEnter, on mouseLeave

on moveWindow

Utilisation

```
-- Syntaxe Lingo
on moveWindow
  instruction(s)
end

// Syntaxe JavaScript
function moveWindow() {
  instruction(s);
}
```

Description

Message système et gestionnaire d'événement; contient les instructions exécutées lorsqu'une fenêtre est déplacée, par exemple lorsque l'utilisateur fait glisser une animation vers un nouvel emplacement de la scène. Pratique pour placer le code Lingo qui doit être exécuté à chaque fois que la fenêtre d'une animation est déplacée.

Exemple

Le gestionnaire suivant affiche un message dans la fenêtre Messages lorsque la fenêtre d'une animation en cours de lecture est déplacée :

```
-- Syntaxe Lingo
on moveWindow
  put("Déplacement de la fenêtre contenant" && _movie.name)
end

// Syntaxe JavaScript
function moveWindow() {
  put "Déplacement de la fenêtre contenant + _movie.name);
}
```

Voir aussi

activeWindow, name (3D), windowList

on openWindow

Utilisation

```
-- Syntaxe Lingo
on openWindow
  instruction(s)
end

// Syntaxe JavaScript
function openWindow() {
  instruction(s);
}
```

Description

Message système et gestionnaire d'événements; contient des instructions exécutées lorsque Director ouvre l'animation en tant qu'animation dans une fenêtre. C'est aussi un endroit idéal pour placer des instructions à exécuter à chaque fois que l'animation s'ouvre dans une fenêtre.

Exemple

Le gestionnaire suivant exécute le fichier audio Hourra lorsque la fenêtre de l'animation s'ouvre :

```
-- Syntaxe Lingo
on openWindow
  sound(2).play(member("Hourra"))
end

// Syntaxe JavaScript
function openWindow() {
  sound(2).play(member("Hourra"));
}
```

on prepareFrame

Utilisation

```
-- Syntaxe Lingo
on prepareFrame
  instruction(s)
end

// Syntaxe JavaScript
function prepareFrame {
  instruction(s);
}
```

Description

Message système et gestionnaire d'événements; contient des instructions exécutées immédiatement avant le dessin de l'image courante.

Contrairement aux événements `beginSprite` et `endSprite`, un événement `prepareFrame` est généré à chaque fois que la tête de lecture arrive sur une image.

Le gestionnaire `on prepareFrame` est un bon endroit pour changer les propriétés de l'image-objet avant qu'elle ne soit dessinée.

S'il est utilisé dans un comportement, le gestionnaire `on prepareFrame` reçoit la référence `me`.

Les commandes `go`, `play` et `updateStage` sont désactivées dans un gestionnaire `on prepareFrame`.

Exemple

Le gestionnaire suivant définit la propriété `locH` de l'image-objet à laquelle le comportement est lié :

```
-- Syntaxe Lingo
on prepareFrame me
  sprite(me.spriteNum).locH = _mouse.mouseH
end

// Syntaxe JavaScript
function prepareFrame() {
  sprite(spriteNum).locH = _mouse.mouseH;
}
```

Voir aussi

`on enterFrame`

on prepareMovie

Utilisation

```
-- Syntaxe Lingo
on prepareMovie
  instruction(s)
end

// Syntaxe JavaScript
function prepareMovie() {
  instruction(s);
}
```

Description

Message système et gestionnaire d'événements; contient des instructions exécutées après que l'animation précharge les acteurs, mais avant qu'elle ne :

- crée des instances de comportements liées aux images-objets de la première image lue ;
- prépare la première image lue, y compris son dessin, la lecture des sons et l'exécution des transitions et des effets de palette.

Les nouvelles variables globales utilisées pour le comportement des images-objets de la première image doivent être initialisées dans le gestionnaire `on prepareMovie`. Il est inutile de redéfinir les variables globales déjà définies par l'animation précédente.

Un gestionnaire `on prepareMovie` est un bon endroit pour placer un code Lingo ou JavaScript servant à créer les variables globales, initialiser les variables, lire un son pendant que le reste de l'animation se charge en mémoire ou vérifier et ajuster les paramètres de l'ordinateur tels que le nombre de couleurs.

Les commandes `go`, `play` et `updateStage` sont désactivées dans un gestionnaire `on prepareMovie`.

Exemple

Le gestionnaire suivant crée une variable globale lorsque l'animation débute :

```
-- Syntaxe Lingo
on prepareMovie
  global ScoreActuel
  ScoreActuel = 0
end

// Syntaxe JavaScript
function prepareMovie() {
  _global.ScoreActuel = 0;
}
```

Voir aussi

on enterFrame, on startMovie

on resizeWindow

Utilisation

```
-- Syntaxe Lingo
on resizeWindow
  instruction(s)
end

// Syntaxe JavaScript
function resizeWindow() {
  instruction(s);
}
```

Description

Message système et gestionnaire d'événements; contient les instructions activées lorsqu'une animation est lue dans une fenêtre et que l'utilisateur modifie les dimensions de cette fenêtre en tirant sur sa case de redimensionnement ou sur l'un de ses bords.

Un gestionnaire d'événement `on resizeWindow` est idéal pour insérer des instructions se rapportant aux dimensions de la fenêtre, comme le positionnement des images-objets et le recadrement des animations de type vidéo numérique.

Exemple

Ce gestionnaire déplace l'image-objet 3 vers les coordonnées stockées dans la variable `emplacementCentral` lorsque la fenêtre lue par l'animation est redimensionnée :

```
-- Syntaxe Lingo
on resizeWindow emplacementCentral
  sprite(3).loc = emplacementCentral
end

// Syntaxe JavaScript
function resizeWindow(emplacementCentral) {
  sprite(3).loc = emplacementCentral;
}
```

Voir aussi

drawRect, sourceRect

on rightMouseDown (gestionnaire d'événement)

Utilisation

```
-- Syntaxe Lingo
on rightMousedown
  instruction(s)
end

// Syntaxe JavaScript
function rightMouseDown() {
  instruction(s);
}
```

Description

Message système et gestionnaire d'événements; sous Windows, spécifie les instructions exécutées lorsque l'utilisateur appuie sur le bouton droit de la souris. Sur Macintosh, les instructions sont exécutées lorsque l'utilisateur appuie simultanément sur le bouton de la souris et la touche Ctrl, et que la propriété `emulateMultiButtonMouse` a la valeur `TRUE` ; si cette propriété a la valeur `FALSE`, ce gestionnaire d'événement n'a aucun effet sur Macintosh.

Exemple

Le gestionnaire suivant ouvre la fenêtre Aide lorsque l'utilisateur appuie sur le bouton droit de la souris sous Windows :

```
-- Syntaxe Lingo
on rightMousedown
  window("Aide").open()
end

// Syntaxe JavaScript
function rightMouseDown() {
  window("Aide").open();
}
```

on rightMouseUp (gestionnaire d'événements)

Utilisation

```
-- Syntaxe Lingo
on rightMouseUp
  instruction(s)
end

// Syntaxe JavaScript
function rightMouseUp() {
  instruction(s);
}
```

Description

Message système et gestionnaire d'événements; sous Windows, spécifie les instructions exécutées lorsque l'utilisateur relâche le bouton droit de la souris. Sur Macintosh, les instructions sont exécutées lorsque l'utilisateur relâche le bouton de la souris tout en maintenant la touche Ctrl enfoncée et que la propriété `emulateMultiButtonMouse` a la valeur `TRUE` ; si cette propriété a la valeur `FALSE`, ce gestionnaire d'événement n'a aucun effet sur Macintosh.

Exemple

Le gestionnaire suivant ouvre la fenêtre Aide lorsque l'utilisateur relâche le bouton droit de la souris sous Windows :

```
-- Syntaxe Lingo
on rightMouseUp
  window("Aide").open()
end

// Syntaxe JavaScript
function rightMouseUp() {
  window("Aide").open();
}
```

on runPropertyDialog

Utilisation

```
-- Syntaxe Lingo
on runPropertyDialog me, listeDinitialisationCourante
  instruction(s)
end

// Syntaxe JavaScript
function runPropertyDialog(ListeDinitialisationCourante) {
  instruction(s);
}
```

Description

Message système et gestionnaire d'événements ; contient un code Lingo ou JavaScript définissant des valeurs spécifiques pour les paramètres d'un comportement dans la boîte de dialogue Paramètres. Le message `runPropertyDialog` est envoyé à chaque fois que le comportement est lié à une image-objet ou que l'utilisateur modifie les valeurs initiales de la propriété du comportement d'une image-objet.

Les paramètres courants des propriétés initiales d'un comportement sont passés au gestionnaire sous forme de liste de propriétés. Si le gestionnaire `on runPropertyDialog` n'est pas défini dans le comportement, Director affiche une boîte de dialogue de personnalisation de comportement basée sur la liste de propriétés renvoyée par le gestionnaire `on getPropertyDescriptionList`.

Exemple

Le gestionnaire suivant annule les valeurs des paramètres de la boîte de dialogue Paramètres du comportement. Les nouvelles valeurs sont contenues dans la liste `listeDinitialisation-Courante`. Normalement, la boîte de dialogue Paramètres permet à l'utilisateur de définir les constantes de masse et de gravité. Cependant, le gestionnaire suivant affecte ces valeurs constantes de paramètres sans afficher de boîte de dialogue :

```
-- Syntaxe Lingo
property masse
property constanteGravité

on runPropertyDialog me, listeDinitialisationCourante
  -- forcer la masse à 10
  ListeDinitialisationCourante.setaProp(#masse, 10)
  -- forcer la constanteGravité à 9.8
  ListeDinitialisationCourante.setaProp(#constanteGravité, 9.8)
  return listeDinitialisationCourante
```

```
end

// Syntaxe JavaScript
function runPropertyDialog(ListeDinitialisationCourante) {
  //forcer la masse à 10
  ListeDinitialisationCourante.setaProp("masse", 10)
  //forcer la constanteGravité à 9.8
  ListeDinitialisationCourante.setaProp("constanteGravité", 9.8)
  return(listeDinitialisationCourante)
}
```

Voir aussi

`on getBehaviorDescription`, `on getPropertyDescriptionList`

on savedLocal

Utilisation

```
-- Syntaxe Lingo
on savedLocal
  instruction(s)
end

// Syntaxe JavaScript
function savedLocal() {
  instruction(s);
}
```

Description

Message système et gestionnaire d'événements. Cette propriété est fournie pour permettre des améliorations dans de futures versions de Shockwave Player.

Voir aussi

`allowSaveLocal`

on sendXML

Utilisation

```
-- Syntaxe Lingo
on sendXML "sendxmlstring", "fenêtre", "données"
  instruction(s)
end

// Syntaxe JavaScript
function sendXML(sendxmlstring, fenêtre, données) {
  instruction(s);
}
```

Description

Gestionnaire d'événement; fonctionne de manière similaire à la méthode de scripting `getURL` également disponible avec l'Xtra Macromedia Flash Asset. Le gestionnaire `on sendXML` est appelé en Lingo lorsque la méthode ActionScript *`objetXML`*`.send` est exécutée dans une image-objet Flash ou dans un objet Flash XML.

Dans ActionScript, la méthode *objetXML*.send transmet deux paramètres, en plus des données XML, dans l'objet XML. Ces paramètres sont les suivants :

- *url* – L'URL à laquelle envoyer les données XML. En règle générale, c'est l'URL d'un script serveur qui attend de traiter les données XML.
- *fenêtre* – la fenêtre de navigateur dans laquelle afficher la réponse du serveur.

La méthode ActionScript *objetXML*.send peut être appelée dans Director, soit par une image-objet Flash, soit par un objet Flash XML global créé en Lingo. Dans ce cas, le gestionnaire Lingo on sendXML est appelé, et les mêmes paramètres sont transmis au gestionnaire.

L'instruction Lingo suivante illustre le mode de réception des paramètres par le gestionnaire on sendXML :

```
on sendXML me, Lurl, fenêtreCible, donnéesXml
```

Ces paramètres sont en corrélation avec le paramètre *objetXML*.send, comme suit :

- Lurl – L'URL à laquelle envoyer les données XML.
- fenêtreCible – la fenêtre de navigateur dans laquelle afficher la réponse du serveur.
- donnéesXML – Les données XML de l'objet XML Flash.

En créant un gestionnaire on sendXML dans votre animation Director, vous lui permettez de traiter les événements *objetXML*.send générés dans une image-objet Flash ou un objet Flash global.

Les images-objets Flash peuvent également charger des données XML externes ou analyser des données XML internes. L'Xtra Flash Asset gère ces fonctions de la même manière qu'un contenu Flash 5 ou Flash MX dans votre navigateur.

Exemple

La commande Lingo suivante obtient les informations de méthode *objetXML*.send d'une image-objet Flash, redirige le navigateur vers l'URL et transmet les données XML à l'URL :

```
-- Syntaxe Lingo
on sendXML me, Lurl, fenêtreCible, donnéesXml
  gotoNetPage(Lurl, fenêtreCible)
  postNetText(Lurl, donnéesXml)
end

// Syntaxe JavaScript
function sendXML(Lurl, fenêtreCible, donnéesXml) {
  gotoNetPage(Lurl, fenêtreCible);
  postNetText(Lurl, donnéesXml);
}
```

on startMovie

Utilisation

```
-- Syntaxe Lingo
on startMovie
  instruction(s)
end

// Syntaxe JavaScript
function startMovie() {
  instruction(s);
}
```

Description

Message système et gestionnaire d'événement; contient des instructions exécutées avant que la tête de lecture n'atteigne la première image de l'animation. L'événement `startMovie` se produit après l'événement `prepareFrame` et avant l'événement `enterFrame`.

Un gestionnaire `on startMovie` est idéal pour placer les éléments Lingo initialisant les images-objets de la première image de l'animation.

Exemple

Le gestionnaire suivant rend les images-objets invisibles au démarrage de l'animation :

```
-- Syntaxe Lingo
on startMovie
  repeat with compteur = 10 to 50
    sprite(compteur).visible = 0
  end repeat
end startMovie

// Syntaxe JavaScript
function startMovie() {
  for(compteur=10;compteur<=50;compteur++) {
    sprite(compteur).visible = 0;
  }
}
```

Voir aussi

`on prepareMovie`

on stepFrame

Utilisation

```
-- Syntaxe Lingo
on stepFrame
  instruction(s)
end

// Syntaxe JavaScript
function stepFrame() {
  instruction(s);
}
```

Description

Message système et gestionnaire d'événement ; fonctionne avec les scripts présents dans la liste `actorList` puisque ce sont les seuls à recevoir des messages `on stepFrame`. Ce gestionnaire d'événement est exécuté lorsque la tête de lecture arrive sur une image ou que la scène est mise à jour.

Un gestionnaire `on stepFrame` est un emplacement utile pour les instructions que vous souhaitez exécuter fréquemment pour une série d'objets particulière. Affectez les objets à la liste `actorList` lorsque vous souhaitez que les éléments Lingo du gestionnaire `on stepFrame` soient exécutés; inversement, retirez ces objets de la liste `actorList` pour éviter l'exécution de ces éléments Lingo. Lorsque les objets sont dans `actorList`, leurs gestionnaires `on stepFrame` sont exécutés à chaque fois que la tête de lecture arrive sur une image ou que la commande `updateStage` est émise.

Le message `stepFrame` est envoyé avant le message `prepareFrame`.

Affectez des objets à `actorList` pour qu'ils puissent répondre aux messages `stepFrame`. Les objets doivent disposer d'un gestionnaire `on stepFrame` pour utiliser cette fonctionnalité interne avec `actorList`.

Les commandes `go`, `play` et `updateStage` sont désactivées dans un gestionnaire `on stepFrame`.

Exemple

Si l'objet enfant est affecté à `actorList`, le gestionnaire `on stepFrame` de ce script parent met à jour la position de l'image-objet stockée dans la propriété `monImageObjet` à chaque fois que la tête de lecture arrive sur une image :

```
-- Syntaxe Lingo
property monImageObjet

on new me, LimageObjet
  monImageObjet = LimageObjet
  return me
end

on stepFrame me
  sprite(monImageObjet).loc = point(random(640),random(480))
end
```

```
// Syntaxe JavaScript
// définir une classe de construction contenant la propriété monImageObjet
function Frame(LimageObjet) {
  this.monImageObjet = LimageObjet;
}

function stepFrame() {
  var monImage = new Frame(sprite(spriteName).spriteNum);
  sprite(monImage.monImageObjet).loc = point(random(640),random(480));
end
```

on stopMovie

Utilisation

```
-- Syntaxe Lingo
on stopMovie
  instruction(s)
end

// Syntaxe JavaScript
function stopMovie() {
  instruction(s);
}
```

Description

Message système et gestionnaire d'événement; contient des instructions exécutées à l'arrêt de l'animation.

Un gestionnaire `on stopMovie` est l'emplacement idéal pour les éléments Lingo destinés à éliminer les données superflues, comme la fermeture des fichiers ressource, l'élimination des variables globales ou la suppression de champs et d'objets, lorsque l'animation est terminée.

Lorsque le gestionnaire `on stopMovie` figure dans une animation dans une fenêtre, il n'est appelé que lorsque l'animation est lue jusqu'à la fin ou débouche sur une autre animation. Il n'est pas appelé lorsque la fenêtre est fermée ou supprimée par le biais de la commande `forget window`.

Exemple

Le gestionnaire suivant supprime une variable globale lorsque l'animation s'arrête :

```
-- Syntaxe Lingo
global gScoreActuel

on stopMovie
  gScoreActuel = 0
end

// Syntaxe JavaScript
_global.gScoreActuel;

function stopMovie() {
  _global gScoreActuel = 0;
}
```

Voir aussi

`on prepareMovie`

on streamStatus

Utilisation

```
-- Syntaxe Lingo
on streamStatus URL, état, octetsTéléchargés, totalDesoctets, erreur
  instruction(s)
end

// Syntaxe JavaScript
function streamStatus(URL, état, octetsTéléchargés, totalDesoctets, erreur{
  instruction(s);
}
```

Description

Message système et gestionnaire d'événements; appelé régulièrement pour déterminer la quantité d'un objet déjà téléchargée depuis Internet. Ce gestionnaire n'est appelé que si `tellStreamStatus (TRUE)` a été appelé et qu'il a été ajouté à un script d'animation.

Le gestionnaire `on streamStatus` possède les paramètres suivants :

URL	Affiche l'adresse Internet des données en cours de récupération.
état	Affiche l'état du flux en cours de téléchargement. Les valeurs possibles sont `Connecting` (connexion), `Started` (commencé), `InProgress` (en cours), `Complete` (terminé) et `Error` (erreur).
octets Téléchargés	Affiche le nombre d'octets récupérés depuis le réseau.
totalDes Octets	Affiche le nombre total d'octets transférés , si ce chiffre est connu. Cette valeur peut être 0 si le serveur HTTP ne fait pas figurer la longueur du contenu dans l'en-tête MIME.
erreur	Affiche une chaîne vide ("") si le téléchargement n'est pas terminé, OK (*OK*) si le téléchargement a abouti ou un code d'erreur s'il a échoué.

Ces paramètres sont automatiquement remplis par Director avec des informations concernant l'évolution du téléchargement. Ce gestionnaire est appelé automatiquement par Director et il n'y a aucun moyen de contrôler quand il sera appelé la fois suivante. Si des informations relatives à une opération particulière sont requises, appelez `getStreamStatus()`.

Vous pouvez lancer un téléchargement réseau à l'aide de commandes Lingo, en liant les médias avec une URL ou en utilisant un acteur externe à partir d'une URL. Un gestionnaire `streamStatus` sera appelé pour fournir des informations sur tous les transferts à partir du réseau.

Placez le gestionnaire `streamStatus` dans les scripts d'animation.

Exemple

Le gestionnaire suivant détermine l'état d'un objet transféré et affiche son URL :

```
-- Syntaxe Lingo
on streamStatus URL, état, octetsTéléchargés, totalDesOctets
  if état = "Complete" then
    put(URL && "Téléchargement terminé");
  end if
end streamStatus

// Syntaxe JavaScript
function streamStatus(URL, état, octetsTéléchargés, totalDesOctets) {
  if (état == "Terminé") {
    put(URL + " Téléchargement terminé");
  }
}
```

Voir aussi

`getStreamStatus()`, `tellStreamStatus()`

on timeOut

Utilisation

```
-- Syntaxe Lingo
on timeOut
  instruction(s)
end

// Syntaxe JavaScript
function timeOut() {
  instruction(s);
}
```

Description

Message système et gestionnaire d'événement; contient des instructions exécutées lorsque la souris ou le clavier n'a pas été utilisé pendant la durée spécifiée par `timeOutLength`. Placez toujours un gestionnaire `on timeOut` dans un script d'animation.

Pour que la temporisation produise la même réponse pendant toute l'animation, utilisez `timeoutScript` afin de contrôler le comportement du délai d'inactivité de façon centralisée.

Exemple

Le gestionnaire suivant lit l'animation Boucle lorsque les utilisateurs sont restés inactifs pendant l'intervalle de temps défini dans la propriété `timeoutLength`. Il peut être utilisé pour répondre lorsque les utilisateurs quittent l'ordinateur.

```
-- Syntaxe Lingo
on timeOut
  _movie.play("Boucle")
end timeOut

// Syntaxe JavaScript
function timeOut() {
  _movie.play("Boucle");
}
```

trayIconDoubleClick

Utilisation

```
-- Syntaxe Lingo
on trayIconDoubleClick
  instruction(s)
end

// Syntaxe JavaScript
function trayIconDoubleClick() {
  instruction(s);
}
```

Description

Gestionnaire d'événements d'animations et fenêtres (Microsoft Windows seulement). Contient des instructions qui sont exécutées lorsqu'un utilisateur double clique sur l'icône de la barre d'applications.

L'événement `trayIconDoubleClick` n'est envoyé au gestionnaire que si la propriété `systemTrayIcon` est définie comme étant `TRUE`.

Exemple

Le gestionnaire suivant met une animation en pause quand un utilisateur double-clique sur l'icône de la barre d'état système.

```
-- Syntaxe Lingo
on trayIconDoubleClick
  _movie.delay(500)
end

// Syntaxe JavaScript
function trayIconDoubleClick() {
  _movie.delay(500);
}
```

Voir aussi

Animation, systemTrayIcon, trayIconMouseDown, trayIconRightMouseDown, Fenêtre

trayIconMouseDown

Utilisation

```
-- Syntaxe Lingo
on trayIconMouseDown
  instruction(s)
end

// Syntaxe JavaScript
function trayIconMouseDown() {
  instruction(s);
}
```

Description

Gestionnaire d'événements d'animations et fenêtres (Microsoft Windows seulement). Contient des instructions qui sont exécutées lorsqu'un utilisateur clique une fois sur l'icône de la barre d'applications.

L'événement `trayIconMouseDown` n'est envoyé au gestionnaire que si la propriété `systemTrayIcon` est définie comme étant `TRUE`.

Exemple

Le gestionnaire suivant met une animation en pause quand un utilisateur clique avec la souris alors que le pointeur se trouve au-dessus de l'icône de la barre d'état système.

```
-- Syntaxe Lingo
on trayIconMouseDown
  _movie.delay(500)
end

// Syntaxe JavaScript
function trayIconMouseDown() {
  _movie.delay(500);
}
```

Voir aussi

Animation, systemTrayIcon, trayIconDoubleClick, trayIconRightMouseDown, Fenêtre

trayIconRightMouseDown

Utilisation

```
-- Syntaxe Lingo
on trayIconRightMouseDown
  instruction(s)
end

// Syntaxe JavaScript
function trayIconRightMouseDown() {
  instruction(s);
}
```

Description

Gestionnaire d'événements d'animations et fenêtres (Microsoft Windows seulement). Contient des instructions qui sont exécutées lorsqu'un utilisateur clique avec le bouton droit de la souris sur l'icône de la barre d'applications.

L'événement `trayIconRightMouseDown` n'est envoyé au gestionnaire que si la propriété `systemTrayIcon` est définie comme étant `TRUE`.

Exemple

Le gestionnaire suivant met une animation en pause quand un utilisateur clique sur l'icône de la barre d'état système avec le bouton droit.

```
-- Syntaxe Lingo
on trayIconRightMouseDown
  _movie.delay(500)
end

// Syntaxe JavaScript
function trayIconRightMouseDown() {
  _movie.delay(500);
}
```

Voir aussi

Animation, systemTrayIcon, trayIconDoubleClick, trayIconMouseDown, Fenêtre

on zoomWindow

Utilisation

```
-- Syntaxe Lingo
on zoomWindow
  instruction(s)
end

// Syntaxe JavaScript
function zoomWindow() {
  instruction(s);
}
```

Description

Message système et gestionnaire d'événement ; contient des instructions exécutées lorsqu'une animation dans une fenêtre est redimensionnée. Cela se produit lorsque l'utilisateur clique sur le bouton Réduction ou Agrandissement (Windows) ou sur le bouton Zoom (Macintosh). Le système d'exploitation détermine les dimensions après avoir redimensionné la fenêtre.

Un gestionnaire d'événement `on zoomWindow` est un emplacement idéal pour insérer des éléments Lingo permettant de réorganiser les images-objets lorsque les dimensions de la fenêtre évoluent.

Exemple

Ce gestionnaire déplace l'image-objet 3 vers les coordonnées stockées dans la variable `emplacementCentral` lorsque la fenêtre lue par l'animation est redimensionnée :

```
-- Syntaxe Lingo
on zoomWindow
  emplacementCentral = point(10, 10)
  sprite(3).loc = emplacementCentral
end

// Syntaxe JavaScript
function zoomWindow() {
  var emplacementCentral = point(10, 10);
  sprite(3).loc = emplacementCentral;
}
```

Voir aussi

drawRect, sourceRect, on resizeWindow

CHAPITRE 11
Mots-clés

Cette section propose une liste alphabétique de tous les mots-clés disponibles dans Macromedia Director MX 2004.

Ces mots-clés ne s'appliquent qu'à Lingo. La syntaxe JavaScript contient des mots-clés et des expressions dont la fonction est similaire aux mots-clés Lingo suivants, mais qui ne sont pas traités ici. Pour plus d'informations au sujet des mots-clés et expressions de la syntaxe JavaScript, consultez le Chapitre 2, *Principes de base du scripting dans Director*, page 9.

\ (continuation)

Utilisation

```
-- Syntaxe Lingo
première partie d'une instruction sur cette ligne \
deuxième partie de l'instruction \
troisième partie de l'instruction
```

Description

Symbole de continuation; lorsqu'il est utilisé comme dernier caractère d'une ligne, indique que l'instruction continue à la ligne suivante. Lingo interprète alors ces lignes comme une seule instruction.

Exemple

L'instruction suivante utilise le caractère \ pour diviser l'instruction en deux lignes :

```
-- Syntaxe Lingo
if sprite("monImageObjet").member = member("monActeur") then \
_player.alert("L'image-objet a été créée à partir de monActeur")
```

case

Utilisation

```
-- Syntaxe Lingo
case expression of
  expression1 : instruction
  expression2 : Instruction(s)
  expression3, expression4 : instruction
  {otherwise: Instruction(s)}
end case
```

Description

Mot-clé ; lance une structure de branchements multiples plus facile à rédiger qu'une suite d'instructions `if...then`.

Lingo compare la valeur de *`case expression`* aux expressions des lignes suivantes. Cette comparaison commence au début de ces lignes et continue dans l'ordre jusqu'à ce que Lingo rencontre une expression identique à *`case expression`*.

Le cas échéant, Lingo exécute la ou les instructions correspondantes suivant les deux points placés après l'expression identique. Si une seule instruction suit l'expression identique, l'instruction peut être placée sur la même ligne. Les instructions multiples doivent apparaître sur des lignes immédiatement en retrait après l'expression identique.

Lorsque plusieurs correspondances possibles pourraient entraîner Lingo à exécuter les mêmes instructions, les expressions doivent être séparées par des virgules. (La ligne contenant *`expression3`* et *`expression4`* est un exemple de ce genre de situation.)

Lingo suspend sa recherche de correspondance dès qu'il rencontre la première expression correspondant à celle recherchée.

Si l'instruction facultative `otherwise` figure à la fin de la structure case, les instructions qui suivent `otherwise` sont exécutées si Lingo ne rencontre aucune expression identique.

Exemple

Le gestionnaire suivant teste la touche que l'utilisateur vient d'enfoncer et répond en conséquence.

- Si l'utilisateur a appuyé sur A, l'animation passe à l'image Pomme.
- Si l'utilisateur a appuyé sur B ou C, l'animation exécute la transition demandée et passe à l'image Oranges.
- Si l'utilisateur a appuyé sur n'importe quelle autre touche, l'ordinateur émet un bip sonore.

```
on keyDown
  case (_key.key) of
    "a" : _movie.go("Pomme")
    "b", "c":
       _movie.puppetTransition(99)
       _movie.go("Oranges")
    otherwise: _sound.beep()
  end case
end keyDown
```

L'instruction `case` suivante vérifie si le curseur se trouve sur l'image-objet 1, 2 ou 3 et exécute l'élément Lingo approprié :

```
case _movie.rollOver() of
  1: sound(1).play(member("Clarinette"))
  2: sound(1).play(member("Tambour"))
  3: sound(1).play(member("Bongos"))
end case
```

char...of

Utilisation

```
-- Syntaxe Lingo
expressionActeurTexte.char[quelCaractère]
charquelCaractèreof variableChaîneouChamp
expressionActeurTexte.char[premierCaractère..dernierCaractère]
char premierCaractère to dernierCaractère of variableChaîneouChamp
```

Description

Mot-clé ; identifie un caractère ou une plage de caractères dans une sous-chaîne. Une expression de sous-chaîne peut être n'importe quel caractère, mot, élément ou ligne dans n'importe quelle source de texte (telle que des acteurs champ et des variables) contenant une chaîne.

- Une expression utilisant *quelCaractère* identifie un caractère spécifique.
- Une expression utilisant *premierCaractère* et *dernierCaractère* identifie une plage de caractères.

Ces expressions doivent être des nombres entiers spécifiant un caractère ou une plage de caractères dans une sous-chaîne. Les caractères peuvent être des lettres, des nombres, des signes de ponctuation, des espaces et des caractères de contrôle comme Tab ou Retour.

Le mot-clé `char...of` peut être testé, mais pas défini. Utilisez la commande `put...into` pour modifier les caractères d'une chaîne.

Exemple

L'instruction suivante affiche le premier caractère de la chaîne 9,00 euros :

```
put(("9,00 euros").char[1..1])
-- "$"
```

L'instruction suivante affiche la chaîne 9,00 euros entière :

```
put(("9,00 euros").char[1.0.5])
-- "$9.00"
```

L'instruction suivante modifie les cinq premiers caractères du deuxième mot de la troisième ligne d'un acteur texte :

```
member("question").line[3].word[2].char[1..5] = "?????"
```

Voir aussi

`mouseMember`, `mouseItem`, `mouseLine`, `mouseWord`

end

Utilisation

```
-- Syntaxe Lingo
end
```

Description

Mot-clé ; marque la fin des gestionnaires et des structures de contrôle à plusieurs lignes.

Exemple

Le gestionnaire `mouseDown` suivant se termine par une instruction `end mouseDown`.

```
on mouseDown
  _player.alert("Le bouton de la souris a été enfoncé")
end mouseDown
```

end case

Utilisation

```
-- Syntaxe Lingo
end case
```

Description

Mot-clé ; termine une instruction `case`.

Exemple

Le gestionnaire suivant utilise le mot-clé `end case` pour terminer l'instruction `case` :

```
on keyDown
  case (_key.key) of
    "a" : _movie.go("Pomme")
    "b", "c":
       _movie.puppetTransition(99)
       _movie.go("Oranges")
    otherwise: _sound.beep()
  end case
end keyDown
```

Voir aussi

`case`

exit

Utilisation

```
-- Syntaxe Lingo
exit
```

Description

Mot-clé ; indique à Lingo de quitter un gestionnaire et de retourner où le gestionnaire a été appelé. Si le gestionnaire est imbriqué dans un autre gestionnaire, Lingo retourne au gestionnaire principal.

Exemple

La première instruction du script suivant vérifie si le moniteur est en noir et blanc et, le cas échéant, sort du gestionnaire :

```
on définitionDesCouleurs
  if _system.colorDepth = 1 then exit
  sprite(1).forecolor = 35
end
```

Voir aussi

abort, halt(), quit(), pass, return (mot-clé)

exit repeat

Utilisation

```
-- Syntaxe Lingo
exit repeat
```

Description

Mot-clé ; indique à Lingo de quitter une boucle et de passer à l'instruction suivant l'instruction `end repeat`, sans toutefois quitter la méthode ou le gestionnaire courant.

Le mot-clé `exit repeat` est pratique pour sortir d'une boucle lorsqu'une condition spécifiée, telle que l'égalité de deux valeurs ou la présence d'une valeur donnée dans une variable, existe.

Exemple

Le gestionnaire suivant cherche la position de la première voyelle dans une chaîne représentée par la variable `chaîneDeTest`. Dès que la première voyelle est trouvée, la commande `exit repeat` indique à Lingo de quitter la boucle et de passer à l'instruction `return i` :

```
on rechercheDeVoyelle chaîneDeTest
  repeat with i = 1 to chaîneDeTest.char[chaîneDeTest.char.count]
    if "aeiou" contains chaîneDeTest.char[i] then exit repeat
  end repeat
  return i
end
```

Voir aussi

repeat while, repeat with

field

Utilisation

```
field( quelChamp)
```

Description

Mot-clé ; fait référence à l'acteur champ spécifié par *quelChamp*.

- Lorsque *quelChamp* est une chaîne, il est utilisé comme nom d'acteur.
- Lorsque *quelChamp* est un nombre entier, il est utilisé comme numéro d'acteur.

Les chaînes de caractères et expressions de sous-chaînes peuvent être lues ou placées dans le champ.

Le terme `field` était utilisé dans les versions précédentes de Director et est conservé pour une compatibilité amont. Pour les nouvelles animations, utilisez `member` pour faire référence aux acteurs champs.

Exemple

L'instruction suivante place les caractères 5 à 10 du champ nommé Entrée dans la variable `monMotClé` :

```
monMotClé = field("Entrée").char[5..10]
```

L'instruction suivante vérifie si l'utilisateur a saisi le mot *bureau* et, le cas échéant, passe à l'image devisBureau :

```
if member("devis") contains "bureau" then _movie.go("devisBureau")
```

Voir aussi

`char...of`, `item...of`, `line...of`, `word...of`

global

Utilisation

`global variable1 {, variable2} {, variable3}...`

Description

Mot-clé ; définit une variable comme variable globale pour que les autres gestionnaires ou animations puissent la partager.

Chaque gestionnaire qui examine ou change le contenu d'une variable globale doit utiliser le mot-clé `global` pour identifier la variable comme une variable globale. Autrement, le gestionnaire traite la variable comme une variable locale, même si un autre gestionnaire l'a déclarée comme globale.

Remarque : Pour assurer que les variables globales soient disponibles dans l'ensemble d'une animation, déclarez et initialisez-les dans le gestionnaire `prepareMovie`. Ensuite, si vous quittez l'animation et revenez à celle-ci à partir d'une autre animation, vos variables globales reprendront leurs valeurs initiales à moins que vous ne vérifiiez d'abord qu'elles ne sont pas déjà définies.

Une variable globale peut être déclarée dans n'importe quel gestionnaire ou script. Sa valeur peut être utilisée par d'autres gestionnaires ou scripts qui déclarent également la variable comme globale. Si le script change la valeur de la variable, la nouvelle valeur est disponible pour tous les autres gestionnaires traitant la variable comme globale.

Une variable globale est disponible dans n'importe quel script ou animation, quel que soit l'endroit où elle a été d'abord déclarée; elle n'est pas automatiquement supprimée lorsque vous naviguez vers une autre image, animation ou fenêtre.

Les variables manipulées dans la fenêtre Messages sont automatiquement globales, même si elles ne sont pas explicitement déclarées comme telles.

Les animations à contenu Macromedia Shockwave exécutées sur Internet ne peuvent pas accéder aux variables globales d'autres animations, y compris les animations exécutées sur la même page HTML. Les animations peuvent uniquement partager des variables globales si une animation intégrée navigue vers une autre animation et est remplacée par le biais des commandes `goToNetMovie` ou `go movie`.

Exemple

L'exemple suivant donne à la variable globale PointDeDépart/ sa valeur initiale de 1 si elle ne contient pas déjà une valeur. Cela permet une navigation vers l'animation et aussi à partir de celle-ci, sans perte des données enregistrées.

```
global gPointDeDépart

on prepareMovie
  if voidP(gPointDeDépart) then gPointDeDépart = 1
end
```

Voir aussi

showGlobals(), property, gotoNetMovie

if

Utilisation

```
ifexpressionLogique then instruction
if expressionLogique then instruction
else instruction
end if
if expressionLogique then
     instruction(s)
end if
if expressionLogique then
     instruction(s)
else
     instruction(s)
end if
if expressionLogique1 then
     instruction(s)
else if expressionLogique2 then
     instruction(s)
else if expressionLogique3 then
     instruction(s)
end if
if expressionLogique1 then
     instruction(s)
else expressionLogique2
end if
```

Description

Mot-clé ; la structure `if...then` évalue l'expression logique spécifiée par *expressionLogique*.

- Si la condition a la valeur `TRUE`, Lingo exécute les instructions suivant `then`.
- Si la condition a la valeur `FALSE`, Lingo exécute les instructions suivant `else`. Si aucune instruction ne suit `else`, Lingo quitte la structure `if...then`.
- Toutes les parties de la condition doivent être évaluées, l'exécution ne s'arrêtant pas à la première condition remplie ou non remplie. Un code plus rapide peut donc être créé en imbriquant des instructions `if...then` sur des lignes séparées au lieu de toutes les placer sur la première ligne à évaluer.

Lorsque la condition est une propriété, Lingo vérifie automatiquement si elle a la valeur `TRUE`. Vous n'avez pas besoin d'ajouter explicitement la phrase `= TRUE` après la propriété.

La partie `else` de l'instruction est facultative. Pour utiliser plus d'une instruction *`then-statement`* ou *`else`*, vous devez conclure par la forme `end if`.

La partie `else` correspond toujours à l'instruction `if` précédente ; vous devez donc parfois inclure une instruction `else nothing` pour associer un mot-clé `else` au mot-clé `if` approprié.

Remarque : Une façon rapide de déterminer dans la fenêtre Script si le script est correctement lié est d'appuyer sur la touche Tab. Cela force Director à vérifier la fenêtre Script ouverte et afficher la présentation en retrait du contenu. Les différences seront immédiatement visibles.

Exemple

L'instruction suivante vérifie si l'utilisateur a appuyé sur un retour chariot et, le cas échéant, continue :

```
if the key = RETURN then go the frame + 1
```

Le gestionnaire suivant vérifie si les touches Cmd et Q ont été enfoncées simultanément et, le cas échéant, exécute les instructions suivantes :

```
on keyDown
  if (_key.commandDown) and (_key.key = "q") then
    nettoyage
    quit
  end if
end keyDown
```

Comparez les deux constructions suivantes et les résultats en matière de performance. La première construction évalue les deux conditions et doit déterminer la mesure du temps, ce qui peut prendre un moment. La seconde construction évalue la première condition; la seconde condition étant vérifiée uniquement si la première condition a la valeur `TRUE`.

```
imageObjetSousLeCurseur = rollOver()
if (imageObjetSousLeCurseur > 25) and MesureDuTempsDepuisLeDépart() then
  _player.alert("Vous avez trouvé le trésor caché !")
end if
```

La construction plus rapide serait :

```
imageObjetSousLeCurseur = rollOver()
if (imageObjetSousLeCurseur > 25) then
  if MesureDuTempsDepuisLeDépart() then
    _player.alert("Vous avez trouvé le trésor caché !")
  end if
end if
```

Voir aussi

`case`

INF

Utilisation

```
-- Syntaxe Lingo
INF
```

Description

Valeur renvoyée ; indique qu'une expression Lingo spécifiée est évaluée comme un nombre infini.

Voir aussi

`NAN`

item...of

Utilisation

```
-- Syntaxe Lingo
texpressionActeurTexte.item[quelElément]
item quelElément of variableChaîneOuChamp
expressionActeurTexte.item[premierElément..dernierElément]
item premierElément to dernierElément of variableChaîneOuChamp
```

Description

Mot-clé ; spécifie un élément ou une série d'éléments d'une sous-chaîne. Un élément est une série de caractères délimités par le séparateur défini dans la propriété `itemDelimiter`.

Les termes *quelElément*, *premierElément* et *dernierElément* doivent être des nombres entiers ou des expressions entières faisant référence à la position des éléments dans la sous-chaîne.

Les sous-chaînes permettent de faire référence à tout caractère, mot, élément ou ligne de n'importe quelle chaîne de texte. Les chaînes possibles sont les acteurs champ et texte et les variables contenant des chaînes.

Lorsque le nombre spécifiant le dernier élément est supérieur à celui correspondant à la position de cet élément dans la sous-chaîne, le dernier élément est spécifié à la place.

Exemple

L'instruction suivante détermine le troisième élément d'une sous-chaîne composée de noms de couleurs et affiche le résultat dans la fenêtre Messages :

```
put("rouge,jaune,bleu vert,orange".item[3])
-- "bleu vert"
```

Le résultat est la sous-chaîne « bleu vert » car tous les caractères placés entre virgules sont pris en compte.

L'instruction suivante détermine les éléments du troisième au cinquième élément de la sous-chaîne. Puisque celle-ci ne contient que quatre éléments, seuls le troisième et le quatrième sont renvoyés. Le résultat apparaît dans la fenêtre Messages.

```
put("rouge,jaune,bleu vert,orange".item[3..5])
-- "bleu vert, orange"
put item 5 of "rouge, jaune, bleu vert, orange"
-- ""
```

L'instruction suivante insère l'élément Bureau en quatrième position dans la deuxième ligne de l'acteur champ Tous les devis :

```
member("Tous les devis").line[2].item[4] = "Bureau"
```

Voir aussi

`char...of`, `itemDelimiter`, `number of members`, `word...of`

line...of

Utilisation

```
-- Syntaxe Lingo
expressionActeurTexte.line[quelleLigne]
line quelleLigne of ChampOuVariableChaîne
expressionActeurTexte.line[premièreLigne..dernièreLigne]
line premièreLigne to dernièreLigne of ChampOuVariableChaîne
```

Description

Mot-clé ; identifie une ligne ou une série de lignes d'une sous-chaîne. Une ligne est constituée d'une série de caractères délimités par des retours de chariot et non par des retours à la ligne automatiques.

Les expressions *quelleLigne*, *premièreLigne* et *dernièreLigne* doivent être des nombres entiers spécifiant une ligne de la sous-chaîne.

Les sous-chaînes peuvent représenter n'importe quel caractère, mot, élément ou ligne d'un groupe de caractères. Les sources de texte peuvent être des acteurs champ et des variables contenant des chaînes.

Exemple

L'instruction suivante affecte les quatre premières lignes de la variable `Action` à l'acteur champ Tâches :

```
member("Tâches").text = Action.line[1..4]
```

L'instruction suivante insère le mot `et` après le deuxième mot de la troisième ligne de la chaîne affectée à la variable `Notes` :

```
put "et" after Notes.line[3].word[2]
```

Voir aussi

`char...of`, `item...of`, `word...of`, `number of members`

loop (mot-clé)

Utilisation

```
-- Syntaxe Lingo
_movie.goLoop()
```

Description

Mot-clé ; fait référence au repère.

Exemple

Le gestionnaire suivant fait boucler l'animation entre le repère précédent et l'image courante :

```
on exitFrame
  _movie.goLoop()
end exitFrame
```

me

Utilisation

```
-- Syntaxe Lingo
me
```

Description

Variable spéciale; s'utilise à l'intérieur de scripts parents et de comportements pour faire référence à l'objet courant lorsqu'il est une instance du script parent, du comportement ou d'une variable contenant l'adresse mémoire de l'objet.

Ce terme n'a aucune signification prédéfinie en Lingo. Le terme `me` est utilisé par convention.

Vous pourrez voir un exemple de `me` dans une animation en consultant l'animation Parent Scripts du dossier Learning/Lingo, lui-même dans le dossier de Director.

Exemple

L'instruction suivante affecte l'objet `monOiseau1` au script Oiseau. Le mot-clé `me` accepte le script Oiseau et sert à renvoyer ce paramètre.

```
monOiseau1 = new script ("Oiseau")
```

Voici le gestionnaire `on new` du script Oiseau :

```
on new me
  return me
end
```

Les deux ensembles de gestionnaires suivants forment un script parent. Le premier ensemble utilise `me` pour faire référence à l'objet enfant. Le second ensemble utilise la variable `monAdresse` pour faire référence à l'objet enfant. Pour ce qui est du reste, les scripts parents sont les mêmes.

Premier ensemble :

```
property mesDonnées

on new me, lesDonnées
  mesDonnées = lesDonnées
  return me
end

on stepFrame me
  traiterLesDonnées me
end
```

Second ensemble :

```
property mesDonnées

on new monAdresse, lesDonnées
  mesDonnées = lesDonnées
  return monAdresse
end

on stepFrame monAdresse
  traiterLesDonnées monAdresse
end
```

Voir aussi

`new()`, `ancestor`

menu

Utilisation

```
-- Syntaxe Lingo
menu: nomDeMenu
nomDélément | script
nomDélément | script
...
```

or

```
menu: nomDeMenu
nomDélément | script
nomDélément | script
...
[plus de menus]
```

Description

Mot-clé ; utilisé avec la commande `installMenu`, spécifie le contenu des menus personnalisés. Les acteurs champ contiennent des définitions de menus. Faites référence à ces définitions en utilisant le nom ou le numéro de l'acteur.

Le mot-clé `menu` est immédiatement suivi d'un deux-points, d'un espace et du nom du menu. Spécifiez les éléments de ce menu sur les lignes suivantes. Vous pouvez définir un script qui sera exécuté lorsque l'utilisateur choisit un élément du menu en plaçant le script après le symbole barre verticale (|). Un nouveau menu est défini par les occurrences suivantes du mot-clé `menu`.

Remarque : Les menus ne sont pas disponibles dans Shockwave Player.

Sur Macintosh, vous pouvez utiliser des caractères spéciaux pour définir des menus personnalisés. Ces caractères spéciaux sont sensibles à la casse. Par exemple, pour qu'un élément de menu apparaisse en gras, la lettre *B* doit être en majuscules.

Des symboles spéciaux doivent suivre le nom de l'élément de menu et précéder le symbole barre verticale (|). Vous pouvez également utiliser plus d'un caractère spécial pour définir un élément de menu. L'utilisation de <B<U, par exemple, définit le style Gras et Souligné.

Evitez de formater les caractères spéciaux des animations qui seront lues sur des plates-formes différentes, Windows ne supportant pas toujours ce formatage.

Symbole	Exemple	Description
`@`	`menu: @`	*Sur Macintosh, crée le symbole Pomme et active les éléments de la barre des menus lorsque vous définissez un menu Pomme.
`!Ã`	`!ÃSélection rapide`	*Sur Macintosh, place une coche (Option+v) près du menu.
`<B`	`Gras<B`	*Sur Macintosh, donne à l'élément de menu le style Gras.
`<I`	`Italique<I`	*Sur Macintosh, définit le style Italique.
`<U`	`Souligné<U`	*Sur Macintosh, définit le style Souligné.
`<O`	`Relief<O`	*Sur Macintosh, définit le style Relief.
`<S`	`Ombré<S`	*Sur Macintosh, définit le style Ombré.

Symbole	Exemple	Description
|	`Ouvrir/O | go to frame "Ouvrir"`	Associe un script à l'élément de menu.
/	`Quitter/Q`	Définit un équivalent commande-touche.
(	`Enregistrer(`	Désactive l'élément de menu.
(-	`(-`	Crée une ligne désactivée dans le menu.

* identifie les symboles de formatage qui ne fonctionnent que sur Macintosh.

Exemple

Voici le texte d'un acteur champ appelé menuPersonnalisé2 qui permet d'indiquer le contenu d'un menu Fichier personnalisé. Pour installer ce menu, utilisez `installMenu member("menuPersonnalisé2")` pendant la lecture de l'animation. L'élément de menu Convertir exécute le gestionnaire personnalisé `convertirCeci`.

```
menu: Fichier
Open/O | _movie.go("Ouvrir")
Close/W | _movie.go("Fermer")
Convertir/C | convertirCeci
  (-
Quit/Q | _movie.go("Quitter")
```

Voir aussi

`installMenu`, `name`, `number (éléments de menu)`, `checkMark`, `enabled`, `script`

NAN

Utilisation

```
-- Syntaxe Lingo
NAN
```

Description

Valeur renvoyée ; indique que l'expression Lingo spécifiée n'est pas un nombre.

L'instruction suivante tente d'afficher la racine carrée de -1, qui n'est pas un nombre, dans la fenêtre Messages :

```
-- Syntaxe Lingo
put((-1).sqrt) -- NAN
```

Voir aussi

`INF`

next

Utilisation

```
-- Syntaxe Lingo
next
```

Description

Mot-clé ; fait référence au repère suivant de l'animation et revient à utiliser la phrase `the marker (+ 1)`.

Exemple

L'instruction suivante envoie la tête de lecture sur le repère suivant de l'animation :

```
go next
```

Le gestionnaire suivant fait passer la tête de lecture au repère suivant du scénario lorsque l'utilisateur appuie sur la touche flèche vers la droite et au repère précédent lorsqu'il appuie sur la touche flèche vers la gauche :

```
on keyUp
  if (_key.keyCode = 124) then _movie.goNext()
  if (_key.keyCode = 123) then _movie.goPrevious()
end keyUp
```

Voir aussi

loop (mot-clé), goPrevious()

next repeat

Utilisation

```
-- Syntaxe Lingo
next repeat
```

Description

Mot-clé ; fait passer Lingo à l'étape suivante d'une boucle d'un script. Cette fonction est différente de celle du mot-clé `exit repeat`.

Exemple

La boucle suivante n'affiche que les nombres impairs dans la fenêtre Messages :

```
repeat with i = 1 to 10
  if (i mod 2) = 0 then next repeat
  put(i)
end repeat
```

on

Utilisation

```
-- Syntaxe Lingo
on nomDeGestionnaire {argument1}, {argument2}, {argument3} ...
  instruction(s)
end nomDeGestionnaire
```

Description

Mot-clé ; indique le début d'un gestionnaire, une suite d'instructions Lingo que vous pouvez exécuter en utilisant le nom du gestionnaire. Un gestionnaire peut accepter des arguments comme valeurs d'entrée et renvoyer une valeur comme résultat d'une fonction.

Les gestionnaires peuvent être définis dans les comportements, les scripts d'animations et les scripts d'acteurs. Le gestionnaire d'un script d'acteur ne peut être appelé que par les autres gestionnaires du même script. Le gestionnaire d'un script d'animation peut être appelé de partout.

Vous pouvez utiliser le même gestionnaire dans plusieurs animations en plaçant son script dans une distribution partagée.

otherwise

Utilisation

```
-- Syntaxe Lingo
otherwise instruction(s)
```

Description

Mot-clé ; précède les instructions que Lingo exécute si aucune des conditions précédentes d'une instruction `case` n'est remplie.

Ce mot-clé peut s'utiliser pour avertir les utilisateurs d'une entrée hors limites ou d'un type non valide. Il peut aussi s'avérer très utile pour les opérations de débogage pendant le développement.

Exemple

Le gestionnaire suivant détermine la dernière touche sur laquelle l'utilisateur a appuyé et répond en conséquence :

- Si l'utilisateur a appuyé sur A, B ou C, l'animation exécute l'action correspondante qui suit le mot-clé `of`.
- Si l'utilisateur a appuyé sur une autre touche, l'animation exécute l'instruction qui suit le mot-clé `otherwise`. Le cas échéant, l'instruction est un simple message d'alerte.

```
on keyDown
  case (_key.key) of
    "a" : _movie.go("Pomme")
    "b", "c":
       _movie.puppetTransition(99)
       _movie.go("Oranges")
    otherwise: _player.alert("Cette touche n'est pas valide.")
  end case
end keyDown
```

property

Utilisation

```
-- Syntaxe Lingo
property {propriété1}{, propriété2} {,propriété3} {...}
```

Description

Mot-clé ; déclare que les propriétés indiquées par *propriété1*, *propriété2* etc,. sont des variables de propriétés.

Declarez les variables de propriétés au début du script parent ou du script comportement. L'opérateur `the` permet d'y accéder en dehors de ces scripts.

Remarque : La propriété `spriteNum` est disponible à tous les comportements et il suffit de la déclarer pour y accéder.

Vous pouvez faire référence à une propriété dans un script parent ou de comportement sans utiliser le mot-clé `me`. Cependant, pour faire référence à une propriété de l'ancêtre d'un script parent, utilisez la forme `me.propriété`.

Pour les comportements, les propriétés définies dans un script de comportement sont disponibles aux autres comportements associés à la même image-objet.

Vous pouvez manipuler la propriété d'un objet enfant directement en dehors de ses scripts parents au moyen d'une syntaxe semblable à celle utilisée pour manipuler d'autres propriétés. Par exemple, l'instruction suivante définit la propriété `styleDeDéplacement` d'un objet enfant :

```
set the styleDeDéplacement of monObjet to #frénétique
```

Utilisez la fonction `count` pour déterminer le nombre de propriétés contenues dans le script parent d'un objet enfant. Récupérez le nom de ces propriétés au moyen de `getPropAt`. Ajoutez des propriétés à un objet au moyen de `setaProp()`.

Vous pourrez voir un exemple de `property` dans une animation en consultant l'animation Parent Scripts du dossier Learning/Lingo, lui-même dans le dossier de Director.

Exemple

L'instruction suivante permet à chaque objet enfant créé à partir d'un script parent unique de posséder ses propres paramètres de position et de vitesse :

```
property position, vitesse
```

Le gestionnaire de script parent suivant déclare une propriété `pMonNumDimageObjet` pour la rendre disponible :

```
-- script Ancien
property pMaPiste
on new me, quelleImageObjet
  me.pMaPiste = quelleImageObjet
  return me
end
```

Le script de comportement d'origine définit l'ancêtre et transmet la propriété `spriteNum` à tous les comportements :

```
property spriteNum
property ancestor
on beginSprite me
  ancestor = new script("Ancien", spriteNum)
end
```

Voir aussi

`me`, `ancestor`, `spriteNum`

put...after

Utilisation

```
-- Syntaxe Lingo
put expression after expressionSousChaîne
```

Description

Commande ; évalue une expression Lingo, convertit la valeur en chaîne et insère celle-ci à la fin d'une sous-chaîne spécifiée dans un conteneur, sans remplacer le contenu de ce dernier. Si *`expressionSousChaîne`* spécifie une sous-chaîne cible qui n'existe pas, la valeur de la chaîne est insérée de manière appropriée dans le conteneur.

Les expressions de sous-chaîne peuvent représenter n'importe quel caractère, mot, élément ou ligne dans un conteneur quelconque. Les conteneurs peuvent être des acteurs champ, des acteurs texte et des variables contenant des chaînes, ainsi que des caractères, des mots, des éléments, des lignes et des plages spécifiés dans des conteneurs.

Exemple

L'instruction suivante ajoute la chaîne « renard chien chat » après le contenu de l'acteur champ Liste d'animaux :

```
put("renard chien chat") after member("Liste d'animaux")
```

Le même résultat peut s'obtenir avec l'instruction suivante :

```
put "renard chien chat" after member("Liste d'animaux").line[1]
```

Voir aussi

char...of, item...of, line...of, paragraph, word...of, put...before, put...into

put...before

Utilisation

```
-- Syntaxe Lingo
put expression before expressionSousChaîne
```

Description

Commande ; évalue une expression Lingo, convertit la valeur en chaîne et insère celle-ci avant une sous-chaîne spécifiée dans un conteneur, sans remplacer le contenu de ce dernier. Si *expressionSousChaîne* spécifie une sous-chaîne cible qui n'existe pas, la valeur de la chaîne est insérée de manière appropriée dans le conteneur.

Les expressions de sous-chaîne peuvent représenter n'importe quel caractère, mot, élément ou ligne dans un conteneur quelconque. Les conteneurs peuvent être des acteurs champ, des acteurs texte et des variables contenant des chaînes, ainsi que des caractères, des mots, des éléments, des lignes et des plages spécifiés dans des conteneurs.

Exemple

L'instruction suivante affecte à la variable `listeDanimaux` la chaîne « renard chien chat », puis insère le mot ***élan*** avant le deuxième mot de la liste :

```
put "renard chien chat" into listeDanimaux
put "élan" before word 2 of listeDanimaux
```

Le résultat correspond à la chaîne « renard élan chien chat ».

Le même résultat peut s'obtenir avec la syntaxe suivante :

```
put "renard chien chat" into listeDanimaux
put "élan" before listeDanimaux.word[2]
```

Voir aussi

char...of, item...of, line...of, paragraph, word...of, put...after, put...into

put...into

Utilisation

```
-- Syntaxe Lingo
put expression into expressionSousChaîne
```

Description

Commande ; évalue une expression Lingo, convertit la valeur en une chaîne et insère celle-ci pour remplacer une sous-chaîne spécifiée d'un conteneur. Si *expressionSousChaîne* spécifie une sous-chaîne cible qui n'existe pas, la valeur de la chaîne est insérée de manière appropriée dans le conteneur.

Les expressions de sous-chaîne peuvent représenter n'importe quel caractère, mot, élément ou ligne dans un conteneur quelconque. Les conteneurs peuvent être des acteurs champ, des acteurs texte et des variables contenant des chaînes, ainsi que des caractères, des mots, des éléments, des lignes et des plages spécifiés dans des conteneurs.

Lorsqu'une animation est lue en tant qu'applet, la commande `put...into` remplace tout le texte d'un conteneur, et non uniquement des sous-chaînes de texte.

Pour affecter des valeurs à des variables, utilisez la commande `set`.

Exemple

L'instruction suivante change la seconde ligne de l'acteur champ Critiques en Critique par Olivier Pognon :

```
put "Critique par Olivier Pognon" into line 2 of member("Critiques")
```

Le même résultat peut s'obtenir avec un acteur texte au moyen de la syntaxe suivante :

```
put "Critique par Olivier Pognon" into member("Critiques").line[2]
```

Voir aussi

`char...of`, `item...of`, `line...of`, `paragraph`, `word...of`, `put...before`, `put...after`, `set...to`, `set...=`

repeat while

Utilisation

```
-- Syntaxe Lingo
repeat whileconditionTest
    instruction(s)
end repeat
```

Description

Mot-clé ; exécute les *instructions* tant que la condition spécifiée par *conditionTest* a la valeur `TRUE`. Cette structure peut servir pour des instructions Lingo qui lisent des chaînes jusqu'à ce que la fin d'un fichier soit atteinte, qui vérifient des éléments jusqu'à la fin d'une liste, ou qui effectuent une action en boucle jusqu'à ce que l'utilisateur clique sur le bouton de la souris ou le relâche.

Tant qu'il est dans une boucle, Lingo ignore les autres événements. Pour déterminer la touche courante dans une boucle, utilisez la propriété `keyPressed`.

Un seul gestionnaire peut être exécuté à la fois. Si Lingo reste dans une boucle d'une façon prolongée, d'autres événements s'empilent en attente d'évaluation. Les boucles sont donc préférables pour les opérations courtes et rapides ou lorsque vous ne prévoyez aucune action de l'utilisateur.

Si vous devez traiter quelque chose pendant au moins plusieurs secondes, évaluez la fonction dans une boucle avec un compteur quelconque ou testez sa progression.

Si la condition d'arrêt n'est jamais atteinte ou s'il n'existe aucune sortie de la boucle, vous pouvez forcer Director à s'arrêter avec Ctrl+Alt+point (Windows) ou Cmd+point (Macintosh).

Exemple

Le gestionnaire suivant lance le compteur en le remettant à 0 et le fait compter jusqu'à 60 millisecondes :

```
on countTime
  _system.milliseconds
  repeat while _system.milliseconds < 60
  -- en attente
  end repeat
end countTime
```

Voir aussi

`exit`, `exit repeat`, `repeat with`, `keyPressed()`

repeat with

Utilisation

```
-- Syntaxe Lingo
repeat with compteur = début to fin
  instruction(s)
end repeat
```

Description

Mot-clé ; exécute le Lingo spécifié par *`instruction(s)`* autant de fois que spécifié par *`compteur`*. La valeur de *`compteur`* est la différence entre la valeur indiquée par *`début`* et celle indiquée par *`fin`*. Le compteur augmente de 1 à chaque fois que Lingo parcourt la boucle.

La structure `repeat with` sert à appliquer en continu le même effet à un ensemble d'images-objets ou à calculer une série de nombres à une certaine puissance.

Tant qu'il est dans une boucle, Lingo ignore les autres événements. Pour déterminer la touche courante dans une boucle, utilisez la propriété `keyPressed`.

Un seul gestionnaire peut être exécuté à la fois. Si Lingo reste dans une boucle d'une façon prolongée, d'autres événements s'empilent en attente d'évaluation. Les boucles sont donc préférables pour les opérations courtes et rapides ou lorsque vous ne prévoyez aucune action de l'utilisateur.

Si vous devez traiter quelque chose pendant au moins plusieurs secondes, évaluez la fonction dans une boucle avec un compteur ou testez sa progression.

Si la condition d'arrêt n'est jamais atteinte ou s'il n'existe aucune sortie de la boucle, vous pouvez forcer Director à s'arrêter avec Ctrl+Alt+point (Windows) ou Cmd+point (Macintosh).

Exemple

Le gestionnaire suivant transforme les images-objets 1 à 30 en esclaves :

```
on esclaves
  repeat with channel = 1 to 30
    _movie.puppetSprite(channel, TRUE)
  end repeat
end esclaves
```

Voir aussi

exit, exit repeat, repeat while, repeat with...down to, repeat with...in list

repeat with...down to

Utilisation

```
-- Syntaxe Lingo
repeat with variable = valeurInitiale down to valeurFinale
```

Description

Mot-clé ; décompte par incréments de 1 à partir de *valeurInitiale* jusqu'à *valeurFinale*.

Un seul gestionnaire peut être exécuté à la fois. Si Lingo reste dans une boucle d'une façon prolongée, d'autres événements s'empilent en attente d'évaluation. Les boucles sont donc préférables pour les opérations courtes et rapides ou lorsque vous ne prévoyez aucune action de l'utilisateur.

Tant qu'il est dans une boucle, Lingo ignore les autres événements. Pour déterminer la touche courante dans une boucle, utilisez la propriété `keyPressed`.

Si vous devez traiter quelque chose pendant au moins plusieurs secondes, évaluez la fonction dans une boucle avec un compteur ou testez sa progression.

Si la condition d'arrêt n'est jamais atteinte ou s'il n'existe aucune sortie de la boucle, vous pouvez forcer Director à s'arrêter avec Ctrl+Alt+point (Windows) ou Cmd+point (Macintosh).

Exemple

Le gestionnaire suivant contient une boucle qui décompte de 20 à 15 :

```
on countDown
  repeat with i = 20 down to 15
    sprite(6).member = 10 + i
    _movie.updateStage()
  end repeat
end
```

repeat with...in list

Utilisation

```
-- Syntaxe Lingo
repeat with variable in uneListe
```

Description

Mot-clé ; affecte à la variable les valeurs successives issues de la liste spécifiée.

Tant qu'il est dans une boucle, Lingo ignore les autres événements, à l'exception des touches. Pour déterminer la touche courante dans une boucle, utilisez la propriété `keyPressed`.

Un seul gestionnaire peut être exécuté à la fois. Si Lingo reste dans une boucle d'une façon prolongée, d'autres événements s'empilent en attente d'évaluation. Les boucles sont donc préférables pour les opérations courtes et rapides ou lorsque vous ne prévoyez aucune action de l'utilisateur.

Si vous devez traiter quelque chose pendant au moins plusieurs secondes, évaluez la fonction dans une boucle avec un compteur ou testez sa progression.

Si la condition d'arrêt n'est jamais atteinte ou s'il n'existe aucune sortie de la boucle, vous pouvez forcer Director à s'arrêter avec Ctrl+Alt+point (Windows) ou Cmd+point (Macintosh).

Exemple

L'instruction suivante affiche quatre valeurs dans la fenêtre Messages :

```
repeat with i in [1, 2, 3, 4]
  put(i)
end repeat
```

return (mot-clé)

Utilisation

```
-- Syntaxe Lingo
return expression
```

Description

Mot-clé*; renvoie la valeur* expression et quitte le gestionnaire. L'argument *expression* peut être une valeur Lingo quelconque.

Lorsque vous appelez un gestionnaire qui sert de fonction définie par l'utilisateur et possède une valeur de renvoi, vous devez entourer de parenthèses les listes d'arguments, même lorsque ces listes sont vides, comme dans le cas du gestionnaire de fonction `lancerLesDés` illustré sous la fonction `result`.

La fonction du mot-clé `return` est similaire à celle de la commande `exit`, à l'exception près que `return` renvoie également une valeur à l'instruction qui a appelé le gestionnaire. La commande `return` dans un gestionnaire entraîne la sortie immédiate de ce gestionnaire, mais peut renvoyer une valeur au Lingo l'ayant appelé.

L'utilisation de `return` dans des scripts orientés objet peut être difficile à comprendre. Il est plus aisé de commencer par utiliser `return` pour créer des fonctions et sortir de gestionnaires. Vous verrez ensuite que la ligne `return me` dans un gestionnaire `on new` fournit un moyen de passer une référence à un objet créé de façon qu'il puisse être affecté à un nom de variable.

Le mot-clé `return` n'est pas identique à la constante de caractère `RETURN`, qui correspond à un retour de chariot. La fonction dépend du contexte.

Pour récupérer une valeur renvoyée, utilisez des parenthèses après le nom du gestionnaire dans l'instruction d'appel pour indiquer que ce nom de gestionnaire est une fonction.

Vous pourrez voir un exemple de `return (mot-clé)` dans une animation en consultant l'animation Parent Scripts du dossier Learning/Lingo, lui-même dans le dossier de Director.

Exemple

Le gestionnaire suivant renvoie un multiple aléatoire de cinq compris entre 5 et 100 :

```
on scoreAléatoire
  score = 5 * random(20)
  return score
end scoreAléatoire
```

Vous appelez ce gestionnaire avec une instruction semblable à la suivante :

```
score = scoreAléatoire()
```

Dans cet exemple, la variable `score` reçoit la valeur renvoyée par la fonction `scoreAléatoire()`. Un script parent remplit la même fonction : en renvoyant la référence de l'objet, le nom de la variable du code d'appel fournit un pointeur pour références ultérieures à cet objet.

Voir aussi

`result`, `RETURN (constante)`

set...to, set...=

Utilisation

```
-- Syntaxe Lingo
propriétéLingo = expression
variable = expression
```

Description

Commande ; évalue une expression et affecte le résultat à la propriété spécifiée par *`propriétéLingo`* ou à la variable spécifiée par *`variable`*.

Exemple

L'instruction suivante donne à l'acteur 3 le nom de Coucher de soleil :

```
member(3).name = "Coucher de soleil"
```

L'instruction suivante inverse l'état de la propriété `soundEnabled`. Si la propriété `soundEnabled` a la valeur `TRUE` (son activé), cette instruction le désactive. Si la propriété `soundEnabled` a la valeur `FALSE` (son désactivé), cette instruction l'active.

```
_sound.soundEnabled = not (_sound.soundEnabled)
```

L'instruction suivante affecte à la variable `voyelles` la chaîne « aeiou » :

```
voyelles = "aeiou"
```

Voir aussi

`property`

sprite...intersects

Utilisation

```
-- Syntaxe Lingo
sprite(image-objet1).intersects(image-objet2)
spriteimageObjet1).intersects imageObjet2
```

Description

Mot-clé ; opérateur comparant la position de deux images-objets pour déterminer si le quadrilatère de l'*imageObjet1* est en contact avec celui de l'imageObjet2 (TRUE) ou non (FALSE).

Si l'encre Dessin seul est appliquée aux deux images-objets, la comparaison porte sur leurs contours, non sur leurs quadrilatères. Le contour d'une image-objet est défini par l'ensemble des pixels autres que blanc formant sa bordure.

Il s'agit d'un opérateur de comparaison avec niveau de priorité de 5.

Remarque : L'opérateur point est requis lorsque l'imageObjet1 n'est pas une expression simple – une expression contenant une opération mathématique.

Exemple

L'instruction suivante vérifie si deux images-objets se croisent et, le cas échéant, modifie le contenu de l'acteur champ Notice pour qu'il contienne le texte « Position correcte ».

```
if sprite i intersects j then put("Position correcte") \
  into member("Notice")
```

Voir aussi

sprite...within, quad

sprite...within

Utilisation

```
-- Syntaxe Lingo
sprite(image-objet1).within(image-objet2)
sprite(imageObjet1within imageObjet2)
```

Description

Mot-clé ; opérateur comparant la position de deux images-objets pour déterminer si le quadrilatère de l'*imageObjet1* est complètement à l'intérieur de celui de l'*imageObjet2* (TRUE) ou non (FALSE).

Si l'encre Dessin seul est appliquée aux deux images-objets, la comparaison porte sur leurs contours, non sur leurs quadrilatères. Le contour d'une image-objet est défini par l'ensemble des pixels autres que blanc formant sa bordure.

Il s'agit d'un opérateur de comparaison avec niveau de priorité de 5.

Remarque : L'opérateur point est requis lorsque l'imageObjet1 n'est pas une expression simple – une expression contenant une opération mathématique.

Exemple

L'instruction suivante vérifie si deux images-objets se croisent et, le cas échéant, appelle le gestionnaire Intérieur :

```
if sprite(3).within(2) then Intérieur
```

Voir aussi

sprite...intersects, quad

version

Utilisation

```
-- Syntaxe Lingo
_player.productVersion
```

Description

Mot-clé ; variable système contenant le numéro de version de Director. La même chaîne apparaît dans la boîte de dialogue Lire les informations du Finder, sur le Macintosh.

Exemple

L'instruction suivante affiche la version de Director dans la fenêtre Messages :

```
put(_player.productVersion)
```

word...of

Utilisation

```
-- Syntaxe Lingo
member(quelActeur).word[quelMot]
expressionActeurTexte.word[quelMot]
expressionSousChaîne[quelMot]
word quelMot of variableChampOuChaîne
variableChampOuChaîne word[quelMot
expressionActeurTexte.word[premierMot..dernierMot]
member(quelActeur).word[premierMot..dernierMot]
word premierMot to dernierMot of expressionSousChaîne
expressionSousChaîne[quelMot..dernierMot]
```

Description

Expression de sous-chaîne ; spécifie un mot ou une série de mots dans une sous-chaîne. Une sous-chaîne de mots est une suite de caractères délimitée par des espaces. (Tout caractère invisible, comme une tabulation ou un retour de chariot, est considéré comme une espace.)

Les expressions *quelMot*, *premierMot* et *dernierMot* doivent avoir pour valeur un nombre entier correspondant à un mot de la sous-chaîne.

Les sous-chaînes peuvent représenter n'importe quel caractère, mot, élément ou ligne d'un groupe de caractères. Les sources de texte peuvent être des acteurs champ et texte et des variables contenant des chaînes.

Vous pouvez voir un exemple de `word...of` dans une animation en consultant l'animation Text du dossier Learning/Lingo, lui-même dans le dossier de Director.

Exemple

Les instructions suivantes affectent à la variable `listeDanimaux` la chaîne « renard chien chat », puis insèrent le mot *élan* avant le deuxième mot de la liste :

```
listeDanimaux = "renard chien chat"
put "élan" before listeDanimaux.word[2]
```

Le résultat correspond à la chaîne « renard élan chien chat ».

L'instruction suivante demande à Director d'afficher le cinquième mot de la même chaîne dans la fenêtre Messages :

```
put "renard élan chien chat".word[5]
```

Cette chaîne ne comportant pas de cinquième mot, la fenêtre Messages affiche deux guillemets droits(""), ce qui indique une chaîne vide.

Voir aussi

`char...of`, `line...of`, `item...of`, `count()`, `number (mots)`

CHAPITRE 12
Méthodes

Cette section fournit une liste alphabétique de toutes les méthodes disponibles dans Director.

abort

Utilisation

```
-- Syntaxe Lingo
abort

// Syntaxe JavaScript
abort();
```

Description

Commande ; indique à Lingo de sortir du gestionnaire courant et de tout autre gestionnaire qui l'a appelé sans exécuter les instructions restantes du gestionnaire concerné. Elle est différente du mot-clé `exit`, qui revient au gestionnaire à partir duquel le gestionnaire courant a été appelé.

La commande `abort` ne quitte pas Director.

Paramètres

Aucun.

Exemple

L'instruction suivante indique à Lingo de sortir du gestionnaire courant et de tout gestionnaire l'ayant appelé si la quantité de mémoire disponible est inférieure à 50 Ko :

```
-- Syntaxe Lingo
if the freeBytes < 50*1024 then abort

// Syntaxe JavaScript
if (_player.freeBytes < 50*1024) {
  abort()
}
```

Voir aussi

`exit`, `halt()`, `quit()`

abs()

Utilisation

```
-- Syntaxe Lingo
abs (expressionNumérique)

// Syntaxe JavaScript
 Math.abs (expressionNumérique)
```

Description

Fonction mathématique (Lingo uniquement) ; calcule la valeur absolue d'une expression numérique.

La fonction `abs()` peut être utilisée de différentes façons. Elle permet de simplifier le suivi du mouvement de la souris et des images-objets en convertissant leurs coordonnées (qu'elles soient positives ou négatives) en distances (celles-ci sont toujours positives). La fonction `abs()` permet également de traiter les fonctions mathématiques telles que `sqrt()` et `log()`.

Dans la syntaxe JavaScript, utilisez la fonction `abs()` de l'objet Math.

Paramètres

`expressionNumérique` Requis. Nombre entier ou à virgule flottante à partir duquel une valeur absolue est calculée. Si *`expressionNumérique`* est un nombre entier, la valeur absolue est également un nombre entier. Si *`expressionNumérique`* est un nombre à virgule flottante, la valeur absolue est également un nombre à virgule flottante.

Exemple

L'instruction suivante permet de calculer si la valeur absolue de la différence entre la position actuelle de la souris et la valeur de la variable `débutV` est supérieure à 30 (puisque vous n'allez pas utiliser un nombre négatif pour exprimer une distance). Si c'est le cas, la couleur de premier plan de l'image-objet 6 est modifiée.

```
-- Syntaxe Lingo
if (the mouseV - débutV).abs > 30 then sprite(6).forecolor = 95

// Syntaxe JavaScript
if ((_mouse.mouseV - Math.abs(_mouse.débutV)) > 30) {
       sprite(6).foreColor = 95;
}
```

activateAtLoc()

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.activateAtLoc(point(x, y))

// Syntaxe JavaScript
réfObjDvd.activateAtLoc(point(x, y));
```

Description

Méthode de DVD ; Active la sélection de l'élément de menu DVD intégré qui se trouve sous l'emplacement spécifié.

Cette méthode renvoie la valeur 0 en cas de réussite.

Paramètres

`point(x, y)` Requis. Point spécifiant l'emplacement de l'élément de menu DVD intégré.

Exemple

Cette instruction active la sélection de l'élément de menu à un emplacement spécifié :

```
-- Syntaxe Lingo
member("animation1").activateAtLoc(point(100, 200))

// Syntaxe JavaScript
member("animation1").activateAtLoc(point(100, 200));
```

Voir aussi

DVD

activateButton()

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.activateButton()

// Syntaxe JavaScript
réfObjDvd.activateButton();
```

Description

Méthode de DVD ; Active le bouton de menu actuellement sélectionné.

Cette méthode renvoie la valeur 0 en cas de réussite.

Paramètres

Aucun.

Exemple

Cette instruction active le bouton Menu sur un acteur spécifié :

```
-- Syntaxe Lingo
sprite(1).member.activateButton()

// Syntaxe JavaScript
sprite(1).member.activateButton();
```

Voir aussi

DVD

add

Utilisation

```
-- Syntaxe Lingo
listeLinéaire.add(valeur)

// Syntaxe JavaScript
tableau.push(valeur)
```

Description

Commande de liste (pour les listes linéaires uniquement) ; ajoute une valeur à une liste linéaire. Dans le cas d'une liste triée, cette valeur est placée dans l'ordre correct. Dans le cas d'une liste non triée, cette valeur est placée à la fin de la liste.

Cette commande provoque une erreur quand elle est utilisée dans une liste de propriétés.

Remarque : Veillez à ne pas confondre la commande `add` et l'opérateur `+` utilisé pour les additions ou l'opérateur `&` utilisé pour concaténer les chaînes.

Paramètres

`valeur` Requis. Valeur à ajouter à la liste linéaire.

Exemple

Les instructions suivantes ajoutent la valeur 2 à la liste nommée `devis`. La liste résultante est `[3, 4, 1, 2]`.

```
-- Syntaxe Lingo
devis = [3, 4, 1]
devis.add(2)

// Syntaxe JavaScript
devis = new Array(3,4,1);
devis.push(2);
```

L'instruction suivante ajoute 2 à la liste linéaire triée [1, 4, 5]. Le nouvel élément reste en ordre alphanumérique, la liste étant une liste triée.

```
-- Syntaxe Lingo
devis.add(2)

// Syntaxe JavaScript
devis.push(2);
// trier la liste en utilisant la syntaxe JavaScript
devis.sort();
```

Voir aussi

`sort`

add (texture 3D)

Utilisation

```
-- Syntaxe Lingo
member(quelActeur).model(quelModèle).meshdeform.mesh[index].\
  textureLayer.add()

// Syntaxe JavaScript
member(quelActeur).model(quelModèle).meshdeform.mesh[index].\
  textureLayer.add()
```

Description

Commande de modificateur 3D `meshdeform` ; ajoute une couche de texture vide à la maille du modèle.

Vous pouvez copier les coordonnées de texture entre les couches à l'aide du code suivant :

```
réfDeModèle.meshdeform.texturelayer[a].texturecoordinatelist =
  réfDeModèle.meshdeform.texturelayer[b].texturecoordinatelist
```

Paramètres

Aucun.

Exemple

L'instruction suivante crée une nouvelle couche de texture pour la première maille du modèle appelé Oreille.

```
-- Syntaxe Lingo
member("Scène").model("Oreille").meshdeform.mesh[1].textureLayer.add()

// Syntaxe JavaScript
member("Scène").model("Oreille").meshdeform.mesh[1].textureLayer.add();
```

Voir aussi

`meshDeform (modificateur)`, `textureLayer`, `textureCoordinateList`

addAt

Utilisation

```
liste.AddAt(position, valeur)
```

Description

Commande de liste (pour les listes linéaires uniquement) ; ajoute une valeur à une position spécifiée dans la liste.

Cette commande provoque une erreur quand elle est utilisée avec une liste de propriétés.

Paramètres

`position` Requis. Nombre entier qui indique la position à laquelle la valeur spécifiée par *`value`* est ajoutée à la liste.

`valeur` Requis. Valeur à ajouter à la liste.

Exemple

L'instruction suivante ajoute la valeur 8 à la quatrième position de la liste `devis`, qui vaut `[3, 2, 4, 5, 6, 7]`:

```
devis = [3, 2, 4, 5, 6, 7]
devis.addAt(4,8)
```

La valeur de `devis` est maintenant [3, 2, 4, 8, 5, 6, 7].

addBackdrop

Utilisation

```
-- Syntaxe Lingo
sprite(quelleImageObjet).camera{(index)}.addBackdrop(texture,
  emplacementDansLimageObjet, rotation)
member(quelActeur).camera(quelleCaméra).addBackdrop(texture,
  emplacementDansLimageObjet, rotation)

// Syntaxe JavaScript
sprite(quelleImageObjet).camera{(index)}.addBackdrop(texture,
  emplacementDansLimageObjet, rotation);
member(quelActeur).camera(quelleCaméra).addBackdrop(texture,
  emplacementDansLimageObjet, rotation);
```

Description

Commande de caméra 3D ; ajoute un fond à la fin de la liste des fonds de la caméra.

Paramètres

texture Requis. Texture à appliquer au fond.

emplacementDansLimageObjet Requis. Emplacement 2D dont le fond est affiché dans l'image-objet 3D. Cet emplacement est mesuré à partir du coin supérieur gauche de l'image-objet.

rotation Requis. Nombre entier qui définit, en degrés, la rotation à appliquer à la texture.

Exemple

La première ligne de l'instruction suivante crée la texture Rugueuse à partir de l'acteur Cèdre et l'enregistre dans la variable t1. La deuxième ligne applique la texture comme fond à l'emplacement (220, 220) dans l'image-objet 5 avec une rotation de zéro degré. La dernière ligne applique la même texture comme fond pour la caméra 1 de l'acteur Scène à l'emplacement (20, 20) avec une rotation de 45°.

```
t1 = member("Scène").newTexture("Rugueuse", #fromCastMember,\
  member("Cèdre"))
sprite(5).camera.addBackdrop(t1, point(220, 220), 0)
member("Scène").camera[1].addBackdrop(t1, point(20, 20), 45)
```

Voir aussi

removeBackdrop

addCamera

Utilisation

```
-- Syntaxe Lingo
sprite(quelleImageObjet).addCamera(quelleCaméra, index)
-- Syntaxe JavaScript
sprite(quelleImageObjet).addCamera(quelleCaméra, index);
```

Description

Commande 3D ; ajoute une caméra à la liste des caméras de l'image-objet. Les vues des différentes caméras sont affichées devant celles des caméras en position *index* inférieures. Vous pouvez définir la propriété `rect` de chaque caméra afin d'afficher plusieurs vues au sein de l'image-objet.

Paramètres

quelleCaméra Requis. Référence à la caméra à ajouter à la liste des caméras de l'image-objet.

index Requis. Nombre entier spécifiant l'index au niveau duquel la caméra *quelleCaméra* est ajoutée à la liste des caméras. Si la valeur *index* est supérieure à la valeur de `cameraCount()`, la caméra est ajoutée à la fin de la liste.

Exemple

L'instruction suivante insère la caméra caméraDeVol en cinquième position de la liste des caméras de l'image-objet 12 :

```
-- Syntaxe Lingo
sprite(12).addCamera(member("scène").camera("caméraDeVol"), 5)

// Syntaxe JavaScript
sprite(12).addCamera(member("scène").camera("caméraDeVol"), 5);
```

Voir aussi

`cameraCount()`, `deleteCamera`

addChild

Utilisation

```
-- Syntaxe Lingo
member(quelActeur).nœud(quelNœudParent).addChild(member\
  (quelActeur).nœud(quelNœudEnfant) {,#conserverLunivers})

// Syntaxe JavaScript
member(quelActeur).nœud(quelNœudParent).addChild(member\
  (quelActeur).nœud(quelNœudEnfant) {,#conserverLunivers})
```

Description

Commande 3D ; ajoute un nœud à la liste des enfants d'un autre nœud et le supprime de la liste des enfants de son parent précédent.

Cette méthode équivaut à définir la propriété `parent` du nœud enfant sur le nœud parent.

Paramètres

quelActeur Requis. Référence à l'acteur qui contient le nœud à ajouter.

nœud Requis. Référence au nœud à ajouter. Ce nœud peut être un modèle, un groupe, une caméra ou une lumière.

conserverLunivers Facultatif. Référence à la caméra à ajouter à la liste des caméras de l'image-objet. Les valeurs correctes sont #preserveWorld et #preserveParent. Lorsque l'enfant est ajouté et que #preserveParent est spécifié, la transformation relative au parent de l'enfant reste la même et ce dernier passe à cette transformation dans l'espace de son nouveau parent. La transformation de l'enfant dans l'univers est recalculée. Lorsque l'enfant est ajouté et que #preserveWorld est spécifié, la transformation de l'enfant dans l'univers reste la même et ce dernier ne passe pas à sa transformation dans l'espace de son nouveau parent. Sa transformation relative au parent est recalculée.

Exemple

L'instruction suivante ajoute le modèle Pneu à la liste des enfants du modèle Voiture.

```
-- Syntaxe Lingo
member("3D").model("Voiture").addChild(member("3D").model("Pneu"))

// Syntaxe JavaScript
member("3D").model("Voiture").addChild(member("3D").model("Pneu"));
```

L'instruction suivante ajoute le modèle Oiseau à la liste des enfants de la caméra MaCaméra et utilise l'argument #preserveWorld pour conserver la position du modèle Oiseau dans l'univers.

```
-- Syntaxe Lingo
member("3D").camera("MaCaméra").addChild(member("3D").model
("Oiseau"), #preserveWorld)

// Syntaxe JavaScript
member("3D").camera("MaCaméra").addChild(member("3D").model
("Oiseau"), symbol("preserveWorld"));
```

Voir aussi

parent, addToWorld, removeFromWorld

addModifier

Utilisation

```
-- Syntaxe Lingo
member(quelActeur).model(quelModèle).addModifier\
  (#typeDeModificateur)

// Syntaxe JavaScript
member(quelActeur).model(quelModèle).addModifier\
  (symbol(typeDeModificateur));
```

Description

Commande 3D de modèle ; ajoute le modificateur spécifié au modèle. Il n'existe aucune valeur par défaut pour cette commande.

Paramètres

typeDeModificateur Requis. Symbole qui spécifie le modificateur à ajouter. Les modificateurs possibles sont les suivants :

- #bonesPlayer
- #collision
- #inker
- #keyframePlayer
- #lod (niveau de détail)
- #meshDeform
- #sds
- #toon

Pour plus d'informations, consultez les entrées des différents modificateurs.

Exemple

L'instruction suivante ajoute le modificateur toon au modèle Boîte.

```
-- Syntaxe Lingo
member("formes").model("Boîte").addModifier(#toon)

// Syntaxe JavaScript
member("formes").model("Boîte").addModifier(symbol("toon"));
```

Voir aussi

bonesPlayer (modificateur), collision (modificateur), inker (modificateur), keyframePlayer (modificateur), lod (modificateur), meshDeform (modificateur), sds (modificateur), toon (modificateur), getRendererServices(), removeModifier, modifier, modifier[], modifiers

addOverlay

Utilisation

```
-- Syntaxe Lingo
sprite(quelleImageObjet).camera{(index)}.addOverlay(texture, \
  emplacementDansLimageObjet, rotation)
member(quelActeur).camera(quelleCaméra).addOverlay(texture, \
  emplacementDansLimageObjet, rotation)

// Syntaxe JavaScript
sprite(quelleImageObjet).camera{(index)}.addOverlay(texture, \
  emplacementDansLimageObjet, rotation)
member(quelActeur).camera(quelleCaméra).addOverlay(texture, \
  emplacementDansLimageObjet, rotation)
```

Description

Commande 3D de caméra ; ajoute un recouvrement à la fin de la liste des recouvrements d'une caméra.

Paramètres

texture Requis. Texture à appliquer au recouvrement.

emplacementDansLimageObjet Requis. Emplacement 2D où le recouvrement est affiché dans l'image-objet 3D. Cet emplacement est mesuré à partir du coin supérieur gauche de l'image-objet.

rotation Requis. Nombre entier qui définit, en degrés, la rotation à appliquer à la texture.

Exemple

La première ligne de l'instruction suivante crée la texture Rugueuse à partir de l'acteur Cèdre et l'enregistre dans la variable t1. La deuxième ligne applique la texture comme recouvrement à l'emplacement (220, 220) dans l'image-objet 5 avec une rotation de zéro degré. La dernière ligne de l'instruction applique la même texture comme recouvrement pour la caméra 1 de l'acteur Scène, au point (20, 20). Une rotation de 45° est appliquée à la texture.

```
-- Syntaxe Lingo
t1 = member("Scène").newTexture("Rugueuse", #fromCastMember,\
  member("Cèdre"))
sprite(5).camera.addOverlay(t1, point(220, 220), 0)
member("Scène").camera[1].addOverlay(t1, point(20, 20), 45)

// Syntaxe JavaScript
t1 = member("Scène").newTexture("Rugueuse", symbol("fromCastMember"),\
  member("Cèdre"));
sprite(5).camera.addOverlay(t1, point(220, 220), 0);
member("Scène").camera[1].addOverlay(t1, point(20, 20), 45);
```

Voir aussi

removeOverlay

addProp

Utilisation

```
liste.addProp(propriété, valeur)
addProp list, propriété, valeur
```

Description

Commande de liste de propriétés ; ajoute, pour les listes de propriétés uniquement, une propriété spécifiée et sa valeur à une liste de propriétés.

Dans le cas d'une liste non triée, cette valeur est placée à la fin de la liste. Dans le cas d'une liste triée, cette valeur est placée dans l'ordre correct.

Si la propriété spécifiée existe déjà dans la liste, la syntaxe Lingo et JavaScript en créent une copie. Afin d'éviter d'avoir des propriétés en double, utilisez la commande `setaProp()` pour modifier la propriété correspondant à la nouvelle entrée de liste.

Cette commande provoque une erreur quand elle est utilisée avec une liste linéaire.

Paramètres

propriété Requis. Propriété à ajouter à la liste.

valeur Requis. Valeur de la propriété à ajouter à la liste.

Exemple

L'instruction suivante ajoute la propriété `dupont` et sa valeur de 3 à la liste de propriétés `devis`, qui contient `[#avatar: 4, #soldes: 1]`. Cette liste étant triée, la nouvelle entrée est placée par ordre alphabétique :

```
devis.addProp(#dupont, 3)
```

La liste qui en résulte est `[#avatar: 4, #dupont: 3, #soldes: 1]`.

L'instruction suivante ajoute l'entrée `dupont: 7` à la liste `devis`, qui contient maintenant `[#avatar: 4, #dupont: 3, #soldes: 1]`. Cette liste contenant déjà la propriété `dupont`, le script en crée une copie :

```
devis.addProp(#dupont, 7)
```

La liste qui en résulte est `[#avatar: 4, #dupont: 3, #dupont: 7, #soldes: 1]`.

addToWorld

Utilisation

```
-- Syntaxe Lingo
member(quelActeur).model(quelModèle).addToWorld()
member(quelActeur).group(quelGroupe).addToWorld()
member(quelActeur).camera(quelleCaméra).addToWorld()
member(quelActeur).light(quelleLumière).addToWorld()

// Syntaxe JavaScript
member(quelActeur).model(quelModèle).addToWorld()
member(quelActeur).group(quelGroupe).addToWorld()
member(quelActeur).camera(quelleCaméra).addToWorld()
member(quelActeur).light(quelleLumière).addToWorld()
```

Description

Commande 3D ; insère le modèle, le groupe, la caméra ou la lumière, dans l'univers 3D de l'acteur comme enfant du groupe World.

Lorsqu'un modèle, un groupe, une caméra ou une lumière, est créé ou cloné, il est automatiquement ajouté à l'univers. Utilisez la commande `removeFromWorld` pour retirer un modèle, un groupe, une caméra ou une lumière, de l'univers 3D sans le supprimer. Utilisez la commande `isInWorld()` pour vérifier si un modèle, un groupe, une caméra ou une lumière, a été ajouté ou retiré de l'univers.

Paramètres

Aucun.

Exemple

L'instruction suivante ajoute le modèle gbCyl à l'univers 3D de l'acteur Scène.

```
-- Syntaxe Lingo
member("Scène").model("gbCyl").addToWorld()

// Syntaxe JavaScript
member("Scène").model("gbCyl").addToWorld();
```

Voir aussi

`isInWorld()`, `removeFromWorld`

addVertex()

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.addVertex(indexOùAjouter, pointOùAjouterSommet {,[
  emplacementVContrôleHoriz, \
  emplacementVContrôleVert ], [ emplacementHContrôleHoriz,
  emplacementHContrôleVert ]})

// Syntaxe JavaScript
réfObjActeur.addVertex(indexOùAjouter, pointOùAjouterSommet {,[
  emplacementVContrôleHoriz, \
  emplacementVContrôleVert ], [ emplacementHContrôleHoriz,
  emplacementHContrôleVert ]});
```

Description

Commande de formes vectorielles ; ajoute un nouveau sommet à un acteur forme vectorielle à la position spécifiée.

Les positions horizontale et verticale du nouveau sommet sont déterminées par rapport à l'origine de l'acteur forme vectorielle.

Si vous utilisez les deux derniers paramètres facultatifs, vous pouvez spécifier la position des poignées de contrôle pour le sommet. La position de ces poignées de contrôle doit être donnée par rapport à celle du sommet ; par conséquent, si aucune position n'est spécifiée, la poignée sera placée sans décalage horizontal ou vertical (valeur 0,0).

Paramètres

indexOùAjouter Requis. Nombre entier qui spécifie l'index au niveau duquel le membre est ajouté.

pointOùAjouterSommet Requis. Nombre entier qui spécifie la position à laquelle le membre est ajouté.

emplacementHContrôleHoriz Facultatif. Nombre entier qui spécifie l'emplacement de la partie horizontale de la poignée de contrôle horizontal.

emplacementVContrôleHoriz Facultatif. Nombre entier qui spécifie l'emplacement de la partie verticale de la poignée de contrôle horizontal.

emplacementHContrôleVert Facultatif. Nombre entier qui spécifie l'emplacement de la partie horizontale de la poignée de contrôle vertical.

emplacementVContrôleVert Facultatif. Nombre entier qui spécifie l'emplacement de la partie verticale de la poignée de contrôle vertical.

Exemple

La ligne suivante ajoute un point de sommet dans la forme vectorielle Archie entre les deux points de sommet existants, à la position horizontale 25 et verticale 15 :

```
-- Syntaxe Lingo
member("Archie").addVertex(2, point(25, 15))

// Syntaxe JavaScript
member("Archie").addVertex(2, point(25, 15));
```

Voir aussi

vertexList, moveVertex(), deleteVertex(), originMode

alert()

Utilisation

```
-- Syntaxe Lingo
_player.alert(chaîneAffichée)

// Syntaxe JavaScript
_player.alert(chaîneAffichée);
```

Description

Méthode de lecteur ; déclenche un bip sonore du système et affiche une boîte de dialogue d'avertissement contenant une chaîne spécifiée.

Le message d'avertissement doit être une chaîne. Pour inclure une variable numérique dans le message d'avertissement, convertissez la variable en chaîne avant de la transmettre à `alert()`.

Paramètres

chaîneAffichée Requis. Chaîne qui représente le texte affiché dans la boîte de dialogue d'avertissement. La chaîne peut contenir jusqu'à 255 caractères.

Exemple

L'instruction suivante crée un message d'alerte indiquant qu'aucun lecteur de CD-ROM n'est connecté :

```
-- Syntaxe Lingo
_player.alert("Aucun lecteur de CD-ROM n'est connecté.")

// Syntaxe JavaScript
_player.alert("Aucun lecteur de CD-ROM n'est connecté.");
```

L'instruction suivante crée un message d'alerte indiquant qu'un fichier est introuvable :

```
-- Syntaxe Lingo
_player.alert("Le fichier" && QUOTE & nomDuFichier & QUOTE && "est
  introuvable.")

// Syntaxe JavaScript
_player.alert("Le fichier \"" + nomDuFichier + "\" est introuvable.");
```

Voir aussi

Lecteur

append

Utilisation

```
liste.append(valeur)
append liste, valeur
```

Description

Commande de liste (pour les listes linéaires uniquement) ; ajoute la valeur spécifiée à la fin d'une liste linéaire. Elle est différente de la commande `add`, qui insère une valeur à une liste triée à l'endroit appropriée de cette liste.

Cette commande, utilisée avec une liste de propriétés, provoque une erreur de script.

Propriétés

`valeur` Requis. Valeur à ajouter à la fin de la liste linéaire.

Exemple

L'instruction suivante ajoute la valeur 2 à la fin de la liste triée appelée devis, qui contient `[1, 3, 4]`, alors même que la position ne correspond pas à l'ordre trié de la liste :

```
set devis = [1, 3, 4]
devis.append(2)
```

La valeur de devis est maintenant `[1, 3, 4, 2]`.

Voir aussi

`add (texture 3D)`, `sort`

appMinimize()

Utilisation

```
-- Syntaxe Lingo
_player.appMinimize()

// Syntaxe JavaScript
_player.appMinimize();
```

Description

Méthode de lecteur ; dans Microsoft Windows, entraîne la réduction d'une projection en une icône dans la barre des tâches Windows. Sur Macintosh, entraîne le masquage d'une projection.

Sur Macintosh, ouvre à nouveau une projection masquée depuis le menu des applications du Macintosh.

Cette méthode est pratique pour les projections et animations MIAW lues sans barre de titre.

Paramètres

Aucun.

Exemple

```
-- Syntaxe Lingo
on mouseUp me
  _player.appMinimize()
end


// Syntaxe JavaScript
function mouseUp() {
  _player.appMinimize();
}
```

Voir aussi

Lecteur

atan()

Utilisation

```
-- Syntaxe Lingo
(nombre).atan
atan (nombre)

// Syntaxe JavaScript
Math.atan(nombre);
```

Description

Fonction mathématique (Lingo uniquement) ; calcule l'arctangente, qui correspond à l'angle dont la tangente est un nombre spécifié. Le résultat est une valeur en radians comprise entre pi/2 et +pi/2.

Dans la syntaxe JavaScript, utilisez la fonction `atan()` de l'objet Math.

Paramètres

Aucun.

Exemple

L'instruction suivante donne l'arctangente de 1 :

```
(1).atan
```

Le résultat, à quatre chiffres après la virgule, est 0,7854, soit approximativement pi/4.

La plupart des fonctions trigonométriques utilisent les radians ; vous devrez alors peut-être convertir les valeurs de degrés en radians.

Le gestionnaire suivant vous permet d'effectuer les conversions de degrés en radians :

```
-- Syntaxe Lingo
on degrésEnRadians valeurEnDegrés
  return valeurEnDegrés * PI/180
end

// Syntaxe JavaScript
function degrésEnRadians (valeurEnDegrés) {
  return valeurEnDegrés * PI/180
}
```

Le gestionnaire affiche le résultat de la conversion de 30 degrés en radians dans la fenêtre Messages :

```
put degrésEnRadians(30)
-- 0.5236
```

Voir aussi

cos(), PI, sin()

beep()

Utilisation

```
-- Syntaxe Lingo
_sound.beep({entNombreDeBits})

// Syntaxe JavaScript
_sound.beep({entNombreDeBits});
```

Description

Méthode audio ; déclenche l'émission d'un bip sonore sur le haut-parleur de l'ordinateur. Ce bip se répète autant de fois que spécifié par *entNombreDeBips*. Si vous n'avez pas spécifié *entNombreDeBips*, le bip sonore ne retentit qu'une seule fois.

- Sous Windows, le bip sonore est le son désigné dans la boîte de dialogue des propriétés sonores.
- Sur Macintosh, le bip sonore est le son sélectionné dans la section Alertes du tableau de bord Moniteurs et son. Si le volume a la valeur 0, le bip sonore est remplacé par un clignotement de la barre de menus.

Paramètres

entNombreDeBits Facultatif. Nombre entier qui spécifie combien de fois les haut-parleurs de l'ordinateur doivent émettre un bip sonore.

Exemple

```
-- Syntaxe Lingo
on mouseUp me
  _sound.beep(1)
end mouseUp

// Syntaxe JavaScript
function mouseUp() {
  _sound.beep(1);
}
```

Voir aussi

Son

beginRecording()

Utilisation

```
-- Syntaxe Lingo
_movie.beginRecording()

// Syntaxe JavaScript
_movie.beginRecording();
```

Description

Méthode d'animation ; lance une session de création du scénario.

Lorsque vous appelez `beginRecording()`, la tête de lecture avance automatiquement d'une image et commence l'enregistrement à partir de cette image. Pour éviter ce comportement et commencer l'enregistrement dans l'image où `beginRecording()` est appelée, insérez une instruction telle que `_movie.go(_movie.frame - 1)` entre les appels de `beginRecording()` et de `endRecording()`.

Dans une animation, vous ne pouvez procéder qu'à une seule session à la fois de mise à jour du scénario.

A chaque appel de `beginRecording()` doit correspondre un appel de `endRecording()`, qui termine la session de création du scénario.

Paramètres

Aucun.

Exemple

Lorsque vous l'utilisez dans le gestionnaire suivant, le mot-clé `beginRecording` démarre une session de création du scénario qui anime l'acteur Balle en l'affectant à la piste d'image-objet 20, puis en déplaçant l'image-objet horizontalement et verticalement sur une série d'images. Le nombre des images est déterminé par l'argument `nombreDimages`.

```
-- Syntaxe Lingo
on animBalle(nombreDimages)
  _movie.beginRecording()
  horizontal = 0
  vertical = 100
  repeat with i = 1 to nombreDimages
    _movie.go(i)
    sprite(20).member = member("Balle")
    sprite(20).locH = horizontal
    sprite(20).locV = vertical
    sprite(20).forecolor = 255
    horizontal = horizontal + 3
    vertical = vertical + 2
    _movie.updateFrame()
  end repeat
  _movie.endRecording()
end animBalle
```

```
// Syntaxe JavaScript
function animBalle(nombreDimages) {
  _movie.beginRecording();
  var horizontal = 0;
  var vertical = 100;
  for (var i = 1; i <= nombreDimages; i++) {
    _movie.go(1);
    sprite(20).member = member("Balle");
    sprite(20).locH = horizontal;
    sprite(20).locV = vertical;
    sprite(20).foreColor = 255;
    horizontal = horizontal + 3;
    vertical = vertical + 2;
    _movie.updateFrame();
  }
  _movie.endRecording();
}
```

Voir aussi

endRecording(), Animation

bitAnd()

Utilisation

bitAnd(*entier1*, *entier2*)

Description

Fonction (Lingo uniquement) ; convertit les deux nombres entiers spécifiés en nombres binaires 32 bits et renvoie un nombre binaire constitué de 1 aux positions où les deux nombres comportaient un 1 et de 0 à toutes les autres positions. Le résultat est un nouveau nombre binaire, que Lingo affiche sous la forme d'un entier en base 10.

Entier	Nombre binaire (abrégé)
6	00110
7	00111
Résultat	
6	00110

Dans la syntaxe JavaScript, utilisez l'opérateur au niveau du bit « & ».

Paramètres

entier1 Requis. Le premier nombre entier.

entier2 Requis. Le second nombre entier.

Exemple

L'instruction suivante compare les versions binaires des entiers 6 et 7 et renvoie le résultat sous la forme d'un entier :

```
put bitAnd(6, 7)
-- 6
```

Voir aussi

bitNot(), bitOr(), bitXor()

bitNot()

Utilisation

```
(entier).bitNot
bitNot(entier)
```

Description

Fonction (Lingo uniquement) ; convertit le nombre entier spécifié en nombre binaire 32 bits et inverse la valeur de chaque chiffre binaire en remplaçant les 1 par des 0 et les 0 par des 1. Le résultat est un nouveau nombre binaire, que Lingo affiche sous la forme d'un nombre entier en base 10.

Entier	Nombre binaire
1	00000000000000000000000000000001
Résultat	
-2	11111111111111111111111111111110

Dans la syntaxe JavaScript, utilisez l'opérateur au niveau du bit « ~ ».

Paramètres

Aucun.

Exemple

L'instruction suivante inverse la représentation binaire de l'entier 1 et renvoie un nouveau nombre.

```
put (1).bitNot
-- -2
```

Voir aussi

bitAnd(), bitOr(), bitXor()

bitOr()

Utilisation

```
bitOr(entier1, entier2)
```

Description

Fonction (Lingo uniquement) ; convertit les deux nombres entiers spécifiés en nombres binaires 32 bits et renvoie un nombre binaire constitué de 1 aux positions où l'un ou l'autre nombre comportait un 1 et de 0 à toutes les autres positions. Le résultat est un nouveau nombre binaire, que Lingo affiche sous la forme d'un entier en base 10.

Entier	Nombre binaire (abrégé)
5	0101
6	0110
Résultat	
7	0111

Dans la syntaxe JavaScript, utilisez l'opérateur au niveau du bit « | ».

Paramètres

entier1 Requis. Le premier nombre entier.

entier2 Requis. Le second nombre entier.

Exemple

L'instruction suivante compare les versions binaires des entiers 5 et 6 et renvoie le résultat sous la forme d'un entier :

```
put bitOr(5, 6)
-- 7
```

Voir aussi

bitNot(), bitAnd(), bitXor()

bitXor()

Utilisation

```
bitXor(entier1, entier2)
```

Description

Fonction ; convertit les deux entiers spécifiés en nombres binaires 32 bits et renvoie un nombre binaire constitué de 1 aux positions où les chiffres des nombres donnés ne correspondent pas et de 0 aux positions où les chiffres sont les mêmes. Le résultat est un nouveau nombre binaire, que Lingo affiche sous la forme d'un entier en base 10.

Entier	Nombre binaire (abrégé)
5	0101
6	0110

Entier	Nombre binaire (abrégé)
Résultat	
3	0011

Dans la syntaxe JavaScript, utilisez l'opérateur au niveau du bit « ^ ».

Paramètres

entier1 Requis. Le premier nombre entier.

entier2 Requis. Le second nombre entier.

Exemple

L'instruction suivante compare les versions binaires des entiers 5 et 6 et renvoie le résultat sous la forme d'un entier :

```
put bitXor(5, 6)
-- 3
```

Voir aussi

bitNot(), bitOr(), bitAnd()

breakLoop()

Utilisation

```
-- Syntaxe Lingo
réfObjPisteAudio.breakLoop()

// Syntaxe JavaScript
réfObjPisteAudio.breakLoop();
```

Description

Méthode de canal audio ; interrompt la lecture du son actuellement mis en boucle dans la piste *réfObjPisteAudio* et entraîne sa lecture jusqu'à la limite endTime.

Si aucun son n'est actuellement en boucle, cette méthode n'a aucun effet.

Paramètres

Aucun.

Exemple

Le gestionnaire suivant arrête la lecture du son mis en boucle dans la piste audio 2 et entraîne sa lecture jusqu'à la fin.

```
-- Syntaxe Lingo
on continuerLaMusiqueDeFond
  sound(2).breakLoop()
end

// Syntaxe JavaScript
function continuerLaMusiqueDeFond() {
  sound(2).breakLoop();
}
```

Voir aussi

endTime, Piste audio

browserName()

Utilisation

```
browserName chemin
browserName()
browserName(#enabled, trueOuFalse)
```

Description

Propriété système, commande et fonction ; spécifie le chemin ou l'emplacement du navigateur web. Vous pouvez utiliser l'Xtra FileIO pour afficher une boîte de dialogue permettant à l'utilisateur de rechercher un navigateur. La méthode `displayOpen()` de l'Xtra FileIO est utile pour afficher une boîte de dialogue d'ouverture.

La forme `browserName()` renvoie le nom du navigateur actuellement spécifié. Si vous placez un nom de chemin, tel celui trouvé au moyen de l'Xtra FileIO, comme argument dans la forme `browserName(CheminCompletVersLapplication)`, vous pouvez définir la propriété. La forme `browserName(#enabled, trueOuFalse)` détermine si le navigateur spécifié est automatiquement lancé par la commande `goToNetPage`.

Cette commande est utile uniquement lors de la lecture dans une projection ou dans Director et n'a aucun effet pour la lecture dans un navigateur.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante indique l'emplacement du navigateur Netscape :

```
browserName "Mon Disque:Mon Dossier:Netscape"
```

L'instruction suivante affiche le nom du navigateur dans une fenêtre Messages :

```
put browserName()
```

build()

Utilisation

```
-- Syntaxe Lingo
member(quelActeur).modelResource(quelleRessDeMod).build()

// Syntaxe JavaScript
member(quelActeur).modelResource(quelleRessDeMod).build();
```

Description

Commande 3D de maille ; construit une maille. Cette commande n'est utilisée qu'avec les ressources de modèle de type `#mesh`.

Vous devrez utiliser la commande `build()` pour la construction initiale de la maille, après avoir modifié l'une de ses propriétés `face`, et après avoir utilisé la commande `generateNormals()`.

Paramètres

Aucun.

Exemple

Cet exemple crée une simple ressource de modèle de type `#mesh`, en spécifie les propriétés, puis l'utilise pour créer un nouveau modèle. Le processus est décrit dans les explications accompagnant l'exemple suivant :

La ligne 1 crée une maille nommée Plan, qui consiste en une face, trois sommets et un maximum de trois couleurs. Le nombre de normales et de coordonnées de textures n'est pas défini. Les normales sont créées par la commande `generateNormals`.

La ligne 2 définit les vecteurs qui seront utilisés comme sommets de Plan.

La ligne 3 affecte les vecteurs aux sommets de la première face de Plan.

La ligne 4 définit les trois couleurs permises par la commande `newMesh`.

La ligne 5 affecte les couleurs à la première face de Plan. La troisième couleur de la liste est appliquée au premier sommet de Plan, la deuxième couleur au deuxième sommet, et la première couleur au troisième sommet. Les couleurs seront étalées sur la première face de Plan en dégradés.

La ligne 6 crée les normales de Plan avec la commande `generateNormals()`.

La ligne 7 appelle la commande `build()` pour construire la maille.

```
-- Syntaxe Lingo
nm = member("Formes").newMesh("Plan",1,3,0,3,0)
  nm.vertexList = [vector(0,0,0), vector(20,0,0), vector(20, 20, 0)]
  nm.face[1].vertices = [ 1,20,3 ]
  nm.colorList = [rgb(255,255,0), rgb(0, 255, 0), rgb(0,0,255)]
  nm.face[1].colors = [3,20,1]
  nm.generateNormals(#smooth)
  nm.build()
  nm = member("Formes").newModel("triModèle", nm)
```

```
// Syntaxe JavaScript
nm = member("Formes").newMesh("Plan",1,3,0,3,0);
  nm.vertexList = [vector(0,0,0), vector(20,0,0), vector(20, 20, 0)];
  nm.face[1].vertices = [ 1,2,3];
  nm.colorList = [rgb(255,255,0), rgb(0, 255, 0), rgb(0,0,255)];
  nm.face[1].colors = [3,2,1];
  nm.generateNormals(#smooth);
  nm.build();
  nm = member("Formes").newModel("triModèle", nm);
```

Voir aussi

generateNormals(), newMesh, face[ ]

cacheDocVerify()

Utilisation

```
-- Syntaxe Lingo
cacheDocVerify #paramètre
cacheDocVerify()

// Syntaxe JavaScript
cacheDocVerify symbol(paramètre);
cacheDocVerify();
```

Description

Fonction ; définit la fréquence de rafraîchissement du contenu d'une page web sur la base des informations contenues dans la mémoire cache de la projection.

La forme `cacheDocVerify()` renvoie le paramètre courant de la mémoire cache.

La fonction `cacheDocVerify` n'est valide que pour les animations exécutées dans Director ou sous forme de projections. Elle n'est pas valide pour les animations avec un contenu Macromedia Shockwave, qui utilisent les paramètres réseau du navigateur web dans lequel elles sont exécutées.

```
-- Syntaxe Lingo
on resetCache
  valeurCourante = cacheDocVerify()
  if valeurCourante = #once then
    alert "Vérification du cache activée"
    cacheDocVerify #always
  end if
end


// Syntaxe JavaScript
function resetCache() {
  valeurCourante = cacheDocVerify();
  if (valeurCourante == symbol("once")) {
    alert ("Vérification du cache activée");
    cacheDocVerify(symbol("always"))
  }
}
```

Paramètres

paramètreCache Facultatif. Symbole qui spécifie la fréquence de rafraîchissement du contenu d'une page web. Les valeurs possibles sont #once (une seule fois, valeur par défaut) et #always (autant de fois que nécessaire). La valeur #once indique à une animation de télécharger une fois un fichier depuis Internet, puis de l'utiliser depuis la mémoire cache sans rechercher une version actualisée sur Internet. La valeur #always indique à une animation d'essayer de télécharger une version actualisée du fichier à chaque fois que l'animation en question appelle une URL.

Voir aussi

cacheSize(), clearCache

cacheSize()

Utilisation

```
-- Syntaxe Lingo
cacheSize taille
cacheSize()

// Syntaxe JavaScript
cacheSize(taille);
cacheSize();
```

Description

Fonction et commande ; définit la taille de la mémoire cache de Director.

La fonction cacheSize n'est valide que pour les animations exécutées sous Director ou sous forme de projections. Elle n'est pas valide pour les animations avec un contenu Macromedia Shockwave, qui utilisent les paramètres réseau du navigateur web dans lequel elles sont exécutées.

Paramètres

nouvelleTailleDeCache Facultatif. Nombre entier qui spécifie la taille de la mémoire cache en kilo-octets.

Exemple

Le gestionnaire suivant vérifie si le paramètre de cache du navigateur a une valeur inférieure à 1 Mo. Si c'est le cas, il affiche un message d'alerte et définit la taille de la mémoire cache à 1 Mo.

```
-- Syntaxe Lingo
on checkCache if
  cacheSize()<1000 then
    alert "augmentation de la mémoire cache à 1Mo"
    cacheSize 1000
  end if
end

// Syntaxe JavaScript
function checkCache() {
  if (cacheSize() < 1000) {
    alert ("augmentation de la mémoire cache à 1 Mo");
    cacheSize(1000);
  }
}
```

Voir aussi

cacheDocVerify(), clearCache

call

Utilisation

```
call #nomDeGestionnaire, script, {args...}
call (#nomDeGestionnaire, instanceDeScript, {args...})
```

Description

Commande ; envoie un message qui appelle un gestionnaire dans un script particulier ou une liste de scripts donnés.

La commande `call` peut utiliser une variable comme nom de gestionnaire. Les messages transmis à l'aide de `call` ne sont pas transmis aux autres scripts liés à l'image-objet, aux scripts d'acteurs, scripts d'image ou scripts d'animation.

Paramètres

`nomDeGestionnaire` Requis. Symbole qui spécifie le gestionnaire à activer.

`instanceDeScript` Requis. Référence au script ou à la liste de scripts qui contient le gestionnaire. Si *`instanceDeScript`* est une instance de script unique, un message d'alerte est envoyé si le gestionnaire n'est pas défini dans le script ancêtre du script. Si *`instanceDeScript`* est une liste d'instances de scripts, le message est envoyé à chaque élément de la liste tour à tour ; aucune alerte n'est générée si le gestionnaire n'est pas défini dans le script ancêtre.

`args` Facultatif. Tous les paramètres facultatifs à transmettre au gestionnaire.

Exemple

Le gestionnaire suivant envoie le message augmenterLeCompteur au premier script de comportement lié à l'image-objet 1 :

```
-- Syntaxe Lingo
on mouseDown me
  -- obtenir la référence du premier comportement de l'image-objet 1
  set xref = getAt (the scriptInstanceList of sprite 1,1)
  -- exécuter le gestionnaire augmenterLeCompteur dans le script référencé,
  -- avec un paramètre
  call (#augmenterLeCompteur, xref, 2)
end

// Syntaxe JavaScript
function mouseDown() {
  // obtenir la référence du premier comportement de l'image-objet 1
  xref = getAt(sprite(1).script(1));
   // exécuter le gestionnaire augmenterLeCompteur dans le script référencé
  call(symbol("augmenterLeCompteur"), xref, 2);
}
```

L'exemple suivant illustre la façon dont une instruction `call` peut appeler les gestionnaires d'un comportement ou d'un script parent et ceux de son ancêtre.

- Le script suivant est un script parent :

```
-- Syntaxe Lingo
-- script Homme
property ancestor
on new me
  set ancestor = new(script "Animal", 2)
  return me
end
on courir me, nouvelOutil
  put "Homme courant sur "&the nombreDePattes of me&" pattes"
end
```

- Le script suivant est le script ancêtre :

```
-- script Animal
property nombreDePattes
on new me, nouveauNombreDePattes
  set nombreDePattes = nouveauNombreDePattes
  return me
end
on courir me
  put "Animal courant sur "& nombreDePattes &" pattes"
end
on marcher me
  put "Animal marchant sur "& nombreDePattes &" pattes"
end
```

- Les instructions suivantes utilisent le script parent et le script ancêtre.

L'instruction suivante crée une instance du script parent :

```
set h = new(script "homme")
```

L'instruction suivante fait marcher l'homme :

```
call #marcher, h
-- "Animal marchant sur 2 pattes"
```

L'instruction suivante fait courir l'homme :

```
set msg = #courir
call msg, h
-- "Homme courant sur 2 pattes"
```

L'instruction suivante crée une seconde instance du script parent :

```
set h2 = new(script "homme")
```

L'instruction suivante envoie un message aux deux instances du script parent :

```
call msg, [h, h2]

-- "Homme courant sur 2 pattes "
-- "Homme courant sur 2 pattes "
```

callAncestor

Utilisation

```
callAncestor nomDeGestionnaire, script, {args...}
```

Description

Commande ; envoie un message au script ancêtre d'un objet enfant.

Les ancêtres peuvent, à leur tour, avoir leurs propres ancêtres.

Lorsque vous utilisez `callAncestor`, le nom du gestionnaire peut être une variable et vous pouvez explicitement contourner les gestionnaires du script principal et accéder directement au script ancêtre.

Paramètres

`nomDeGestionnaire` Requis. Symbole qui spécifie le gestionnaire à activer.

`instanceDeScript` Requis. Référence au script ou à la liste de scripts qui contient le gestionnaire. Si *`instanceDeScript`* est une instance de script unique, un message d'alerte est envoyé si le gestionnaire n'est pas défini dans le script ancêtre du script. Si *`instanceDeScript`* est une liste d'instances de scripts, le message est envoyé à chaque élément de la liste tour à tour ; aucune alerte n'est générée si le gestionnaire n'est pas défini dans le script ancêtre.

`args` Facultatif. Tous les paramètres facultatifs à transmettre au gestionnaire.

Exemple

L'exemple suivant présente la façon dont une instruction `callAncestor` peut appeler des gestionnaires dans l'ancêtre d'un comportement ou d'un script parent.

- Le script suivant est un script parent :

```
-- script "homme"
property ancestor
on new me, nouvelOutil
  set ancestor = new(script "Animal", 2)
  return me
end
on courir me
  put "Homme courant sur "&the nombreDePattes of me&" pattes"
end
```

- Le script suivant est le script ancêtre :

```
-- script "animal"
property nombreDePattes
on new me, nouveauNombreDePattes
  set nombreDePattes = nouveauNombreDePattes
  return me
end
on courir me
  put "Animal courant sur "& nombreDePattes &" pattes"
end
on marcher me
  put "Animal marchant sur "& nombreDePattes &" pattes"
end
```

- Les instructions suivantes utilisent le script parent et le script ancêtre.

  L'instruction suivante crée une instance du script parent :

  ```
  set h = new(script "homme")
  ```

  L'instruction suivante fait marcher l'homme :

  ```
  call #marcher, h
  -- "Animal marchant sur 2 pattes"
  ```

  L'instruction suivante fait courir l'homme :

  ```
  set msg = #courir
  callAncestor msg, h
  -- "Animal courant sur 2 pattes"
  ```

  L'instruction suivante crée une seconde instance du script parent :

  ```
  set h2 = new(script "homme")
  ```

  L'instruction suivante envoie un message au script ancêtre des deux hommes :

  ```
  callAncestor #courir,[h,h2]
  -- "Animal courant sur 2 pattes"
  -- "Animal courant sur 2 pattes"
  ```

Voir aussi

ancestor, new()

callFrame()

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.callFrame(nomOuNumImageFlash)

// Syntaxe JavaScript
réfObjImageObjet.callFrame(nomOuNumImageFlash);
```

Description

Commande ; utilisée pour appeler une série d'actions résidant dans une image d'une image-objet d'animation Flash.

Cette commande transmet un message au moteur ActionScript de Flash et déclenche l'exécution des actions dans l'animation Flash.

Paramètres

nomOuNumImageFlash Requis. Chaîne ou numéro qui spécifie le nom ou le numéro de l'image à appeler.

Exemple

Cette instruction lance l'exécution des actions associées à l'image 10 de l'animation Flash dans l'image-objet 1 :

```
-- Syntaxe Lingo
sprite(1).callFrame(10)

// Syntaxe JavaScript
sprite(1).callFrame(10);
```

camera()

Utilisation

```
member(quelActeur).camera(quelleCaméra)
member(quelActeur).camera[index]
member(quelActeur).camera(quelleCaméra).quellePropriétéDeCaméra
member(quelActeur).camera[index].quellePropriétéDeCaméra
sprite(quelleImageObjet).camera{(index)}
sprite(quelleImageObjet).camera{(index)}.quellePropriétéDeCaméra
```

Description

Elément 3D ; objet à une position de vecteur à partir de laquelle l'univers 3D est observé.

Chaque image-objet possède une liste de caméras. Les vues des différentes caméras de la liste sont affichées au-dessus de celles des caméras en position *index* inférieures. Vous pouvez définir la propriété `rect (caméra)` de chaque caméra afin d'afficher plusieurs vues au sein de l'image-objet.

Les caméras sont enregistrées dans la palette des caméras de l'acteur. Utilisez les commandes `newCamera` et `deleteCamera` pour créer et supprimer les caméras d'un acteur 3D.

La propriété `camera` d'une image-objet est la première caméra de la liste des caméras de l'image-objet. La caméra référencée par `sprite(quelleImageObjet).camera` est la même que `sprite(quelleImageObjet).camera(1)`. Utilisez les commandes `addCamera` et `deleteCamera` pour construire la liste des caméras d'une image-objet 3D.

Exemple

L'instruction suivante affecte à l'image-objet 1 la caméra camArbre de l'acteur Picnic.

```
sprite(1).camera = member("Picnic").camera("camArbre")
```

L'instruction suivante affecte à l'image-objet 1 la caméra 2 de l'acteur Picnic.

```
sprite(1).camera = member("Picnic").camera[2]
```

Voir aussi

`bevelDepth`, `overlay`, `modelUnderLoc`, `spriteSpaceToWorldSpace`, `fog`, `clearAtRender`

cameraCount()

Utilisation

```
-- Syntaxe Lingo
sprite(quelleImageObjet).cameraCount()

// Syntaxe JavaScript
sprite(quelleImageObjet).cameraCount();
```

Description

Commande 3D ; renvoie le nombre d'éléments de la liste des caméras de l'image-objet.

Paramètres

Aucun.

Exemple

L'instruction suivante indique que l'image-objet 5 contient trois caméras.

```
-- Syntaxe Lingo
put sprite(5).cameraCount()
-- 3

// Syntaxe JavaScript
put(sprite(5).cameraCount());
// 3
```

Voir aussi

addCamera, deleteCamera

cancelIdleLoad()

Utilisation

```
-- Syntaxe Lingo
_movie.cancelIdleLoad(entBaliseDeChargement)

// Syntaxe JavaScript
_movie.cancelIdleLoad(entBaliseDeChargement);
```

Description

Méthode d'animation ; annule le chargement de tous les acteurs portant la balise de chargement spécifiée.

Paramètres

entBaliseDeChargement Requis. Nombre entier qui spécifie un groupe d'acteurs mis en file d'attente pour être chargé lorsque l'ordinateur est inactif.

Exemple

L'instruction suivante annule le chargement des acteurs portant la balise de chargement en période d'inactivité numéro 20 :

```
-- Syntaxe Lingo
_movie.cancelIdleLoad(20)

// Syntaxe JavaScript
_movie.cancelIdleLoad(20);
```

Voir aussi

idleLoadTag, Animation

castLib()

Utilisation

```
-- Syntaxe Lingo
castLib(nomOuNumDeDistribution)

// Syntaxe JavaScript
castLib(nomOuNumDeDistribution);
```

Description

Fonction du niveau supérieur ; renvoie une référence à une distribution spécifiée.

La distribution par défaut est la distribution numéro 1. Si vous voulez spécifier un acteur dans une distribution autre que la distribution numéro 1, définissez `castLib()` pour spécifier l'autre distribution.

Paramètres

`nomOuNumDeDistribution` Requis. Chaîne qui spécifie le nom de la distribution ou nombre entier qui spécifie son numéro.

Exemple

L'instruction suivante affecte à la variable `parties` la deuxième distribution :

```
-- Syntaxe Lingo
parties = castLib(2)

// Syntaxe JavaScript
var parties = castLib(2);
```

Voir aussi

`Bibliothèque de distribution`, `castLibNum`

channel() (niveau supérieur)

Utilisation

```
-- Syntaxe Lingo
channel(nomOuNumDePisteAudio)

// Syntaxe JavaScript
channel(nomOuNumDePisteAudio);
```

Description

Fonction du niveau supérieur ; renvoie une référence à un objet Piste audio.

Paramètres

`nomOuNumDePisteAudio` Requis. Chaîne qui spécifie le nom d'une piste audio ou nombre entier qui spécifie sa position d'index.

Exemple

Cette instruction affecte à la variable `nouvellePiste` la piste audio 9 :

```
-- Syntaxe Lingo
nouvellePiste = channel(9)

// Syntaxe JavaScript
var nouvellePiste = channel(9);
```

Voir aussi

Piste audio

channel() (audio)

Utilisation

```
-- Syntaxe Lingo
_sound.channel(entNumDePiste)

// Syntaxe JavaScript
_sound.channel(entNumDePiste);
```

Description

Méthode audio ; renvoie une référence à une piste audio spécifiée.

La fonctionnalité de cette méthode est identique à celle de la méthode `sound()` du niveau supérieur.

Paramètres

`entNumDePiste` Requis. Nombre entier qui spécifie la piste audio à référencer.

Exemple

Cette instruction affecte à la variable maPiste la piste audio 2 :

```
-- Syntaxe Lingo
maPiste = _sound.channel(2)

// Syntaxe JavaScript
var maPiste = _sound.channel(2);
```

Voir aussi

Son, sound(), Piste audio

chapterCount()

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.chapterCount({entTitre})

// Syntaxe JavaScript
réfObjDvd.chapterCount({entTitre});
```

Description

Méthode de DVD ; indique le nombre de chapitres disponibles dans un titre.

Paramètres

entTitre Facultatif. Nombre entier qui spécifie le titre contenant les chapitres à dénombrer. Si ce paramètre est omis, `chapterCount()` renvoie le nombre de chapitres disponibles dans le titre courant.

Voir aussi

`chapterCount`, `DVD`

charPosToLoc()

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.charPosToLoc(énièmeCaractère)

// Syntaxe JavaScript
réfObjActeur.charPosToLoc(énièmeCaractère);
```

Description

Fonction de champ ; renvoie le point de l'acteur champ entier (et pas seulement la partie affichée sur la scène) qui est le plus proche d'un caractère spécifié. Elle est utile pour déterminer l'emplacement précis de caractères individuels.

Les valeurs de `charPosToLoc` sont exprimées en pixels et commencent à partir du coin supérieur gauche de l'acteur champ. Le paramètre *énièmeCaractère* est 1 pour le premier caractère du champ, 2 pour le deuxième, etc.

Paramètres

énièmeCaractère Requis. Le caractère à tester.

Exemple

L'instruction suivante détermine le point auquel apparaît le cinquantième caractère de l'acteur Titre et affecte le résultat à la variable emplacement :

```
-- Syntaxe Lingo
emplacement = member("Titre").charPosToLoc(50)

// Syntaxe JavaScript
var emplacement = member("Titre").charPosToLoc(50);
```

chars()

Utilisation

`chars(`*expressionChaîne*`, `*premierCaractère*`, `*dernierCaractère*`)`

Description

Fonction (Lingo uniquement) ; identifie une sous-chaîne de caractères dans une expression.

Les expressions *premierCaractère* et *dernierCaractère* doivent spécifier une position dans la chaîne.

Si les expressions *premierCaractère* et *dernierCaractère* sont égales, la chaîne ne renvoie qu'un seul caractère. Si *dernierCaractère* est plus grand que la longueur de la chaîne, seule une sous-chaîne allant jusqu'à la longueur de la chaîne est identifiée. Si *dernierCaractère* est placé avant *premierCaractère*, cette fonction renvoie la valeur `EMPTY`.

Pour un exemple d'utilisation de `chars()` dans une animation, reportez-vous à l'animation Text du dossier Learning/Lingo, lui-même inclus dans le dossier de Director.

Dans la syntaxe JavaScript, utilisez la fonction `substr()` de l'objet Chaîne.

Paramètres

expressionChaîne Requis. Chaîne qui spécifie l'expression à partir de laquelle une sous-chaîne est renvoyée.

premierCaractère Requis. Nombre entier qui spécifie le point où la sous-chaîne commence.

dernierCaractère Requis. Nombre entier qui spécifie le point où la sous-chaîne se termine.

Exemple

L'instruction suivante identifie le sixième caractère du mot *Macromedia* :

```
put chars ("Macromedia", 6, 6)
-- "m"
```

L'instruction suivante identifie les caractères compris entre les sixième et dixième caractères du mot *Macromedia* :

```
put chars ("Macromedia", 6, 10)
-- "media"
```

L'instruction suivante essaie d'identifier les caractères compris entre le sixième et le vingtième caractère du mot Macromedia. Puisque ce mot ne contient que 10 caractères, le résultat ne contient que les caractères compris ente le sixième et le dixième caractère.

```
put chars ("Macromedia", 6, 20)
-- "media"
```

Voir aussi

`char...of`, `length()`, `offset() (fonction de chaîne)`, `number (caractères)`

charToNum()

Utilisation

```
(expressionChaîne).charToNum
charToNum(expressionChaîne)
```

Description

Fonction (Lingo uniquement) ; renvoie le code ASCII correspondant au premier caractère d'une expression.

La fonction `charToNum()` est particulièrement utile pour tester la valeur des caractères ASCII créés avec des combinaisons de touches telles que la touche Ctrl et une autre touche alpha-numérique.

Director ne fait pas la différence entre les caractères en majuscules et en minuscules si vous utilisez l'opérateur de comparaison égal (=) ; par exemple, l'instruction `put ("M" = "m")` donne le résultat 1 ou `TRUE`.

Evitez tout problème en utilisant `charToNum()` pour renvoyer le code ASCII d'un caractère et utilisez ensuite le code ASCII pour faire référence à ce caractère.

Dans la syntaxe JavaScript, utilisez la fonction `charCodeAt()` de l'objet Chaîne.

Paramètres

`expressionChaîne` Requis. Chaîne qui spécifie l'expression à tester.

Exemple

L'instruction suivante affiche le code ASCII de la lettre A :

```
put ("A").charToNum
-- 65
```

La comparaison suivante détermine si la lettre saisie est un A majuscule, puis passe à une séquence correcte ou incorrecte du scénario :

```
-- Syntaxe Lingo
on CheckKeyHit theKey
  if (theKey).charToNum = 65 then
    go "Réponse correcte"
  else
    go "Réponse incorrecte"
  end if
end

// Syntaxe JavaScript
function CheckKeyHit(theKey) {
  if (theKey.charToNum() == 65) {
    go ("Réponse correcte");
  } else {
    go ("Réponse incorrecte");
  }
}
```

Voir aussi

`numToChar()`

clearAsObjects()

Utilisation

```
-- Syntaxe Lingo
clearAsObjects()

// Syntaxe JavaScript
clearAsObjects();
```

Description

Commande ; reconfigure le lecteur Flash global utilisé pour les objets ActionScript et supprime tous les objets ActionScript de la mémoire. Cette commande n'efface ni ne reconfigure les références à ces objets stockées dans Lingo. Les références Lingo resteront présentes mais feront référence à des objets inexistants. Chaque référence doit être définie sur `VOID` individuellement.

La commande `clearAsObjects()` n'affecte que les objets globaux, tels que le tableau créé dans cette instruction :

```
-- Syntaxe Lingo
monTableauGlobal = newObject(#array)

// Syntaxe JavaScript
monTableauGlobal = new Array();
```

La commande `clearAsObjects()` n'a aucun effet sur les objets créés dans les références d'images-objets, tel que :

```
monTableau = sprite(2).newObject(#array)
```

Paramètres

Aucun.

Exemple

Cette instruction supprime de la mémoire tous les objets ActionScript créés globalement :

```
-- Syntaxe Lingo
clearAsObjects()

// Syntaxe JavaScript
clearAsObjects();
```

Voir aussi

`newObject()`, `setCallback()`

clearCache

Utilisation

```
clearCache
```

Description

Commande ; vide la mémoire cache réseau de Director.

La commande `clearCache` vide uniquement la mémoire cache, qui est distincte de celle du navigateur.

Les fichiers en cours d'utilisation ne sont pas supprimés de la mémoire cache (jusqu'à leur inactivité).

Paramètres

Aucun.

Exemple

Le gestionnaire suivant vide la mémoire cache au lancement de l'animation :

```
-- Syntaxe Lingo
on startMovie
  clearCache
end

// Syntaxe JavaScript
function startMovie() {
  clearCache();
}
```

Voir aussi

`cacheDocVerify()`, `cacheSize()`

clearError()

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.clearError()

// Syntaxe JavaScript
réfObjActeur.clearError();
```

Description

Commande Flash ; remet à zéro l'état d'erreur d'un acteur Flash en flux continu.

Si une erreur survient alors qu'un acteur est transféré en flux continu vers la mémoire, Director affecte la valeur -1 à la propriété `state` de cet acteur, afin d'indiquer qu'une erreur s'est produite. Dans ce cas, vous pouvez utiliser la fonction `getError` pour déterminer le type d'erreur survenue, puis la commande `clearError` pour remettre à zéro l'état d'erreur de l'acteur. Une fois que vous avez éliminé l'état d'erreur de l'acteur, Director essaie d'ouvrir ce dernier, si l'animation en a encore besoin. D'autre part, la définition des propriétés `pathName`, `linked` et `preload` d'un acteur élimine automatiquement la condition d'erreur.

Paramètres

Aucun.

Exemple

Le gestionnaire suivant vérifie l'occurrence d'une erreur de type mémoire épuisée pour l'acteur Flash Dali, qui a été transféré en mémoire. Si une telle erreur est survenue, le script utilise la commande `unloadCast` pour essayer de libérer de la mémoire ; il amène ensuite la tête de lecture sur une image de l'animation Artistes de Director, dans laquelle l'image-objet de l'animation Flash apparaît pour la première fois, de sorte que Director puisse relancer la lecture de l'animation Flash. Si un autre type d'erreur est survenu, le script passe à une image appelée Désolé, qui explique que l'animation Flash requise ne peut pas être lue.

```
-- Syntaxe Lingo
on vérifierEtatFlash
  if (member("Dali").getError() = #memory) then
    member("Dali").clearError()
    member("Dali").unload()
    unloadCast
  else
    _movie.go("Désolé")
  end if
end

// Syntaxe JavaScript
function CheckFlashStatus() {
  var ge = member("Dali").getError();
  if (ge = "memory") {
    member("Dali").clearError();
    unloadCast;
    _movie.go("Artistes");
  } else {
    _movie.go("Désolé");
  }
}
```

Voir aussi

state (Flash, SWA), getError() (Flash, SWA)

clearFrame()

Utilisation

```
-- Syntaxe Lingo
_movie.clearFrame()

// Syntaxe JavaScript
_movie.clearFrame();
```

Description

Méthode d'animation ; efface toutes les pistes d'images-objets dans une image pendant l'enregistrement du scénario.

Paramètres

Aucun.

Exemple

Le gestionnaire suivant supprime le contenu de chaque image avant de les modifier pendant la création du scénario :

```
-- Syntaxe Lingo
on newScore
  _movie.beginRecording()
  repeat with compteur = 1 to 50
    _movie.clearFrame()
    _movie.frameScript = 25
    _movie.updateFrame()
  end repeat
  _movie.endRecording()
end
```

```
// Syntaxe JavaScript
function newScore() {
  _movie.beginRecording();
  for (var i = 1; i <= 50; i++) {
    _movie.clearFrame();
    _movie.frameScript = 25;
    _movie.updateFrame();
  }
  _movie.endRecording();
}
```

Voir aussi

beginRecording(), endRecording(), Animation, updateFrame()

clearGlobals()

Utilisation

```
-- Syntaxe Lingo
_global.clearGlobals()

// Syntaxe JavaScript
_global.clearGlobals();
```

Description

Méthode globale ; donne à toutes les variables globales la valeur `VOID` (Lingo) ou `null` (syntaxe JavaScript).

Cette méthode est utile lors de l'initialisation de variables globales ou de l'ouverture d'une nouvelle animation qui exige un nouveau jeu de variables globales.

Paramètres

Aucun.

Exemple

Les gestionnaires suivants donnent à toutes les variables globales la valeur `VOID` (Lingo) ou `null` (JavaScript) :

```
-- Syntaxe Lingo
on mouseDown
  _global.clearGlobals()
end

// Syntaxe JavaScript
function mouseDown() {
  _global.clearGlobals();
}
```

Voir aussi

Global

clone

Utilisation

```
member(quelActeur).model(quelModèle).clone(nomDuClone)
member(quelActeur).group(quelGroupe).clone(nomDuClone)
member(quelActeur).light(quelleLumière).clone(nomDuClone)
member(quelActeur).camera(quelleCaméra).clone(nomDuClone)
```

Description

Commande 3D ; crée une copie du modèle, du groupe, de la lumière ou de la caméra, et de tous ses enfants. Le clone a le même parent que le modèle, le groupe, la lumière ou la caméra à partir duquel il a été cloné.

Le clone d'un modèle utilise la même ressource de modèle et la même liste de matériaux que le modèle d'origine.

Si vous ne spécifiez pas le *nomDuClone* ou si vous spécifiez `""`, le clone ne sera pas pris en compte par la méthode `count`, mais apparaîtra dans la scène.

Paramètres

nomDuClone Requis. Spécifie le nom du nouveau clone.

Exemple

L'instruction suivante crée le clone Théière2 à partir du modèle Théière et renvoie une référence au nouveau modèle.

```
copieDeThéière = member("Univers 3D").model("Théière").clone("Théière2")
```

Voir aussi

`cloneDeep`, `cloneModelFromCastmember`, `cloneMotionFromCastmember`, `loadFile()`

cloneDeep

Utilisation

```
member(quelActeur).model(quelModèle).cloneDeep(nomDuClone)
member(quelActeur).group(quelGroupe).cloneDeep(nomDeClone)
member(quelActeur).light(quelleLumière).cloneDeep(nomDeClone)
member(quelActeur).camera(quelleCaméra).cloneDeep(nomDeClone)
```

Description

Commande 3D ; crée une copie du modèle, du groupe, de la lumière ou de la caméra, plus tous les éléments suivants :

- Les ressources de modèle, matériaux et textures utilisés par le modèle ou groupe d'origine
- Les enfants du modèle, du groupe, de la lumière ou de la caméra
- Les ressources de modèle, matériaux et textures utilisés par les enfants

Cette méthode utilise plus de mémoire et prend plus de temps que la commande `clone`.

Paramètres

nomDuClone Requis. Spécifie le nom du nouveau clone.

Exemple

L'instruction suivante crée une copie du modèle Théière et de ses enfants, et des ressources de modèle, des matériaux et des textures utilisés par Théière et ses enfants. La variable `copieDeThéière` est une référence au modèle cloné.

```
copieDeThéière = member("Univers 3D").model("Théière").cloneDeep("Théière2")
```

Voir aussi

`clone`, `cloneModelFromCastmember`, `cloneMotionFromCastmember`, `loadFile()`

cloneModelFromCastmember

Utilisation

```
member(quelActeur).cloneModelFromCastmember\
  (nomDeNouveauModèle, nomDeModèleSource, acteurSource)
```

Description

Commande 3D ; copie un modèle d'acteur, le renomme et l'insère dans un acteur en tant qu'enfant de son univers 3D.

Cette commande copie également les enfants de *nomDeModèleSource*, ainsi que les ressources de modèle, les matériaux et les textures, utilisés par le modèle et ses enfants.

Le chargement de l'acteur source doit être terminé pour la bonne exécution de cette commande.

Paramètres

nomDeNouveauModèle Requis. Spécifie le nom du modèle qui vient d'être cloné.

nomDeModèleSource Requis. Spécifie le modèle à cloner.

acteurSource Requis. Spécifie l'acteur qui contient le modèle à cloner.

Exemple

L'instruction suivante crée une copie du modèle Pluton de l'acteur Scène et l'insère dans l'acteur Scène2 avec le nouveau nom Planète. Les enfants de Pluton sont également importés, de même que les ressources de modèle, les matériaux et les textures utilisés par Pluton et ses enfants.

```
member("Scène2").cloneModelFromCastmember("Planète", "Pluton", \
  member("Scène"))
```

Voir aussi

`cloneMotionFromCastmember`, `clone`, `cloneDeep`, `loadFile()`

cloneMotionFromCastmember

Utilisation

```
member(quelActeur).cloneMotionFromCastmember(nomDeNouveauMouvement, \
  nomDeMouvementSource, acteurSource)
```

Description

Commande 3D ; copie un mouvement d'un acteur, le renomme et l'insère dans un acteur.

Le chargement de l'acteur source doit être terminé pour la bonne exécution de cette commande.

Paramètres

nomDeNouveauMouvement Requis. Spécifie le nom du mouvement qui vient d'être cloné.

nomDeMouvementSource Requis. Spécifie le mouvement à cloner.

acteurSource Requis. Spécifie l'acteur qui contient le mouvement à cloner.

Exemple

L'instruction suivante copie le mouvement Marche de l'acteur Parc, nomme la copie marcheBizarre, et la place dans l'acteur gbActeur.

```
member("gbActeur").cloneMotionFromCastmember("marcheBizarre", \
  "Marche", member("Parc"))
```

Voir aussi

map (3D), cloneModelFromCastmember, clone, cloneDeep, loadFile()

close()

Utilisation

```
-- Syntaxe Lingo
réfObjFenêtre.close()

// Syntaxe JavaScript
réfObjFenêtre.close();
```

Description

Méthode de fenêtre ; ferme une fenêtre.

Toute tentative visant à fermer une fenêtre déjà fermée n'a aucun effet.

Veuillez noter que la fermeture d'une fenêtre ne termine pas l'exécution de l'animation dans la fenêtre et ne la supprime pas non plus de la mémoire. Cette méthode sert simplement à fermer la fenêtre dans laquelle l'animation est exécutée. Pour la rouvrir rapidement, utilisez la méthode `open() (fenêtre)`. Cette procédure assure un accès rapide aux fenêtres qui doivent rester disponibles.

Pour supprimer totalement une fenêtre et la vider de la mémoire, utilisez la méthode `forget()`. Si vous utilisez la commande `forget()`, assurez-vous qu'aucun élément ne fait référence à l'animation dans cette fenêtre, faute de quoi le système génère des erreurs lorsque les scripts essaient de communiquer ou de dialoguer avec la fenêtre supprimée.

Paramètres

Aucun.

Exemple

L'instruction suivante ferme la fenêtre Panneau, qui se trouve dans le sous-dossier Sources MIAW dans le dossier de l'animation courante :

```
-- Syntaxe Lingo
window(_movie.path & "Sources MIAW\Panneau").close()

// Syntaxe JavaScript
window(_movie.path + "Sources MIAW\\Panneau").close();
```

L'instruction suivante ferme la fenêtre 5 dans la liste `windowList` :

```
-- Syntaxe Lingo
window(5).close()

// Syntaxe JavaScript
window(5).close();
```

Voir aussi

forget() (fenêtre), open() (fenêtre), Fenêtre

closeFile()

Utilisation

```
-- Syntaxe Lingo
réfObjFileio.closeFile()

// Syntaxe JavaScript
réfObjFileio.closeFile();
```

Description

Méthode de Fileio ; ferme un fichier.

Paramètres

Aucun.

Voir aussi

Fileio

closeXlib

Utilisation

```
closeXlib quelFichier
```

Description

Commande ; ferme un fichier Xlibrary.

Les Xtras sont enregistrés dans des fichiers Xlibrary. Les fichiers Xlibrary sont les fichiers des ressources contenant les Xtras. Les XCMD et XFCN HyperCard peuvent également être stockés dans des fichiers Xlibrary.

La commande `closeXlib` ne fonctionne pas avec les URL.

Sous Windows, l'extension DLL des Xtras est facultative.

Nous vous recommandons de fermer tout fichier ouvert dès que vous n'en avez plus besoin.

Remarque : Cette commande n'est pas prise en charge dans Shockwave Player.

Paramètres

`quelFichier` Facultatif. Spécifie le fichier Xlibrary à fermer. Si le fichier Xlibrary se trouve dans un dossier autre que celui de l'animation courante, *`quelFichier`* doit spécifier un chemin d'accès. Si *`quelFichier`* est omis, tous les fichiers Xlibrary ouverts sont fermés.

Exemple

L'instruction suivante ferme tous les fichiers Xlibrary ouverts :

```
closeXlib
```

L'instruction suivante ferme le fichier Xlibrary Disque vidéo qui se trouve dans le même dossier que l'animation courante :

```
closeXlib "Xlibrary Disque vidéo"
```

L'instruction suivante ferme le fichier Xlibrary Transporter Xtras du dossier Nouveaux Xtras, qui se trouve dans le même dossier que l'animation. Le disque est identifié par la variable *lecteurCourant* :

```
closeXlib "@:Nouveaux Xtras:Transporter Xtras"
```

Voir aussi

interface(), openXlib

color()

Utilisation

```
-- Syntaxe Lingo
color(entIndexDePalette)
color(entRouge, entVert, entBleu)

// Syntaxe JavaScript
color(entIndexDePalette);
color(entRouge, entVert, entBleu);
```

Description

Fonction du niveau supérieur et type de données. Renvoie un objet de données Couleur sous la forme de valeurs RVB ou de valeurs d'un index de palette 8 bits.

L'objet Couleur obtenu peut être appliqué à des acteurs, à des images-objets et à la scène si besoin est.

Paramètres

entPaletteIndex Requis si vous utilisez des valeurs de palette 8 bits. Entier qui spécifie la valeur de palette 8 bits à utiliser. Les valeurs correctes sont comprises entre 0 et 255. Toutes les autres valeurs sont tronquées.

entRouge Requis si vous utilisez des valeurs RVB. Entier qui spécifie la composante rouge de la couleur dans la palette courante. Les valeurs correctes sont comprises entre 0 et 255. Toutes les autres valeurs sont tronquées.

entVert Requis si vous utilisez des valeurs RVB. Entier qui spécifie la composante verte de la couleur dans la palette courante. Les valeurs correctes sont comprises entre 0 et 255. Toutes les autres valeurs sont tronquées.

entBleu Requis si vous utilisez des valeurs RVB. Entier qui spécifie la composante bleue de la couleur dans la palette courante. Les valeurs correctes sont comprises entre 0 et 255. Toutes les autres valeurs sont tronquées.

Exemple

Ces instructions affichent la couleur de l'image-objet 6 dans la fenêtre Messages, puis définissent la couleur de l'image-objet 6 sur une nouvelle valeur :

```
-- Syntaxe Lingo
put(sprite(6).color) -- paletteIndex(255)
sprite(6).color = color(137)
put(sprite(6).color) -- paletteIndex(137)

// Syntaxe JavaScript
put(sprite(6).color) // paletteIndex(255);
sprite(6).color = color(137);
put(sprite(6).color) // paletteIndex(137);
```

constrainH()

Utilisation

```
-- Syntaxe Lingo
_movie.constrainH(entNumDimageObjet, entPosn)

// Syntaxe JavaScript
_movie.constrainH(entNumDimageObjet, entPosn);
```

Description

Méthode d'animation ; renvoie un nombre entier dont la valeur dépend des coordonnées horizontales des côtés gauche et droit d'une image-objet.

Le nombre entier renvoyé peut avoir trois valeurs.

- Si le paramètre *entPosn* est compris entre les valeurs des coordonnées gauche et droite de l'image-objet, le nombre entier renvoyé est égale à *entPosn*.
- Si le paramètre *entPosn* est inférieur à la valeur de la coordonnée gauche de l'image-objet, le nombre entier renvoyé prend la valeur de la coordonnée gauche de l'image-objet.
- Si le paramètre *entPosn* est supérieur à la valeur de la coordonnée droite de l'image-objet, le nombre entier renvoyé prend la valeur de la coordonnée droite de l'image-objet.

Cette méthode ne modifie pas les propriétés de l'image-objet.

Les méthodes `constrainH()` et `constrainV()` ne contraignent qu'un seul axe chacune.

Paramètres

entNumDimageObjet Requis. Nombre entier qui spécifie l'image-objet dont les coordonnées horizontales sont évaluées par rapport à *entPosn*.

entPosn Requis. Nombre entier à évaluer par rapport aux coordonnées horizontales des côtés gauche et droit de l'image-objet identifiée par *entNumDimageObjet*.

Exemple

Les instructions suivantes vérifient la fonction `constrainH` sur l'image-objet 1 lorsque ses coordonnées gauche et droite sont 40 et 60 :

```
-- Syntaxe Lingo
put(constrainH(1, 20)) -- 40
put(constrainH(1, 55)) --55
put(constrainH(1, 100)) --60

// Syntaxe JavaScript
put(constrainH(1, 20)); // 40
put(constrainH(1, 55)); // 55
put(constrainH(1, 100)); // 60
```

L'instruction suivante limite les déplacements d'un curseur mobile (image-objet 1) aux bords d'une jauge (image-objet 2) lorsque le pointeur de la souris dépasse les bords de cette dernière :

```
-- Syntaxe Lingo
sprite(1).locH = _movie.constrainH(2, _mouse.mouseH)

// Syntaxe JavaScript
sprite(1).locH = _movie.constrainH(2, _mouse.mouseH);
```

Voir aussi

`constrainV()`, `Animation`

constrainV()

Utilisation

```
-- Syntaxe Lingo
_movie.constrainV(entNumDimageObjet, entPosn)

// Syntaxe JavaScript
_movie.constrainV(entNumDimageObjet, entPosn);
```

Description

Méthode d'animation ; renvoie un nombre entier dont la valeur dépend des coordonnées verticales des côtés supérieur et inférieur d'une image-objet.

Le nombre entier renvoyé peut avoir trois valeurs.

- Si le paramètre *entPosn* est compris entre les valeurs des coordonnées supérieure et inférieure de l'image-objet, le nombre entier renvoyé est égale à *entPosn*.
- Si le paramètre *entPosn* est inférieur à la valeur de la coordonnée supérieure de l'image-objet, le nombre entier renvoyé prend la valeur de la coordonnée supérieure de l'image-objet.
- Si le paramètre *entPosn* est supérieur à la valeur de la coordonnée inférieure de l'image-objet, le nombre entier renvoyé prend la valeur de la coordonnée inférieure de l'image-objet.

Cette méthode ne modifie pas les propriétés de l'image-objet.

Les méthodes `constrainV()` et `constrainH()` ne contraignent qu'un seul axe chacune.

Paramètres

entNumDimageObjet Requis. Nombre entier qui identifie l'image-objet dont les coordonnées verticales sont évaluées par rapport à *entPosn*.

entPosn Requis. Nombre entier à évaluer par rapport aux coordonnées verticales des côtés gauche et droit de l'image-objet identifiée par *entNumDimageObjet*.

Exemple

Les instructions suivantes vérifient la fonction `constrainV` de l'image-objet 1 lorsque ses coordonnées supérieure et inférieure sont 40 et 60 :

```
-- Syntaxe Lingo
put(constrainV(1, 20)) -- 40
put(constrainV(1, 55)) --55
put(constrainV(1, 100)) --60

// Syntaxe JavaScript
put(constrainV(1, 20)); // 40
put(constrainV(1, 55)); // 55
put(constrainV(1, 100)); // 60
```

L'instruction suivante limite les déplacements d'un curseur mobile (image-objet 1) aux bords d'une jauge (image-objet 2) lorsque le pointeur de la souris dépasse les bords de cette dernière :

```
-- Syntaxe Lingo
sprite(1).locV = _movie.constrainV(2, _mouse.mouseH)

// Syntaxe JavaScript
sprite(1).locV = _movie.constrainV(2, _mouse.mouseH);
```

Voir aussi

`constrainH()`, `Animation`

copyPixels()

Utilisation

```
-- Syntaxe Lingo
réfObjImage.copyPixels(objImageSource, rectOuQuadDeDest, rectSource {,
  listeDeParamètres})

// Syntaxe JavaScript
réfObjImage.copyPixels(objImageSource, rectOuQuadDeDest, rectSource {,
  listeDeParamètres});
```

Description

Méthode d'image. Copie le contenu d'un rectangle inclus dans un objet image existant à l'intérieur d'un nouvel objet image.

Lors de la copie de pixels d'une zone d'un acteur vers une autre zone du même acteur, il est recommandé de copier d'abord les pixels dans un objet image dupliqué avant de les recopier dans l'acteur d'origine. Copier directement d'une zone à l'autre dans la même image est déconseillé.

Pour simuler l'encre Dessin seul avec `copyPixels()`, créez un objet Dessin seul avec `createMatte()`, puis transmettez cet objet en tant que paramètre `#maskImage` de `copyPixels()`.

Pour un exemple d'utilisation de quadrilatère dans une animation, reportez-vous à l'animation Quad du dossier Learning/Lingo, lui-même inclus dans le dossier de Director.

Paramètres

objImageSource Requis. Référence à l'objet image source à partir duquel les pixels sont copiés.

rectOuQuadDeDest Requis si vous copiez les pixels dans un rectangle à coordonnée d'écran ou un quadrilatère à virgule flottante. Le rectangle ou le quadrilatère dans lequel les pixels sont copiés.

rectSource Requis. Rectangle source à partir duquel les pixels sont copiés.

listeDeParamètres Facultatif. Liste de paramètres qui peuvent être utilisés pour manipuler les pixels copiés avant de les placer dans *rectDeDest* ou *quadDeDest*. La liste de propriétés peut contenir l'intégralité ou une partie des paramètres suivants.

Propriété	Usage et effet
#color	Couleur de premier plan à appliquer pour les effets de colorisation. La valeur par défaut est noir.
#bgColor	Couleur d'arrière-plan à appliquer pour les effets de colorisation ou comme transparence d'arrière-plan. La couleur par défaut est le blanc.
#ink	Type d'encre à appliquer aux pixels copiés. Il peut s'agir d'un symbole d'encre ou de la valeur numérique correspondante. L'encre par défaut est #copy.
#blendLevel	Degré d'opacité (transparence) à appliquer aux pixels copiés. La plage de valeurs est comprise entre 0 et 225. La valeur par défaut est 255 (opaque). L'utilisation d'une valeur inférieure à 255 définit le paramètre #ink sur #blend, ou sur #blendTransparent s'il était initialement défini sur #backgroundTransparent. #blendLevel peut également être remplacé par #blend ; dans ce cas, utilisez une plage de valeurs comprise entre 0 et 100.
#dither	Valeur TRUE ou FALSE qui détermine si les pixels copiés seront tramés lorsqu'ils seront placés dans le *rectDeDestination* des images 8 et 16 bits. La valeur par défaut est FALSE, qui convertit directement les pixels copiés dans la palette de couleurs de la *réfObjImage*.
#useFastQuads	Valeur TRUE ou FALSE qui détermine si les calculs de quadrilatères sont effectués à l'aide de la méthode de Director, plus rapide mais moins précise, lors de la copie de pixels dans un *quadDeDestination*. Définissez ce paramètre sur TRUE si vous utilisez les quadrilatères pour des opérations simples de rotation et d'inclinaison. Définissez ce paramètre sur FALSE pour des quadrilatères aléatoires, tels que ceux utilisés pour les transformations de perspective. La valeur par défaut est FALSE.

Propriété	Usage et effet
#maskImage	Spécifie un objet Masque ou Dessin seul, créé avec les fonctions createMask() ou createMatte(), qui sera utilisé en tant que masque pour les pixels à copier. Ceci permet de reproduire les effets des encres d'images-objets Masque et Dessin seul. Si l'image source possède une couche alpha et que sa propriété useAlpha est TRUE, la couche alpha sera utilisée et l'objet Masque ou Dessin seul spécifié sera ignoré. La valeur par défaut est pas de masque.
#maskOffset	Point indiquant la valeur de décalage x et y à appliquer au masque spécifié par #maskImage. Le décalage est calculé par rapport au coin supérieur gauche de l'image source. Le décalage par défaut est (0, 0).

Exemple

L'instruction suivante copie toute l'image de l'acteur Joyeux dans le rectangle de l'acteur Fleur. Si les acteurs sont de tailles différentes, l'image de l'acteur Joyeux sera redimensionnée pour s'ajuster au rectangle de l'acteur Fleur.

L'instruction suivante copie une portion de l'image de l'acteur Joyeux dans une portion de l'acteur Fleur. La portion de l'image copiée de l'acteur Joyeux est située dans le rectangle (0, 0, 200, 90). Elle est collée dans le rectangle (20, 20, 100, 40) à l'intérieur de l'image de l'acteur Fleur. La portion copiée de l'acteur Joyeux est redimensionnée pour s'adapter aux dimensions du rectangle dans lequel elle est collée.

L'instruction suivante copie entièrement l'image de l'acteur Joyeux dans un rectangle à l'intérieur de l'acteur Fleur. Le rectangle dans lequel l'image copiée de l'acteur Joyeux est collée a la même taille identique que le rectangle de l'acteur Joyeux, de sorte que l'image copiée n'est pas redimensionnée. Le niveau d'opacité de l'image copiée est de 50, elle est donc semi-tranparente, révélant la portion de l'acteur Fleur sur laquelle elle est collée.

Voir aussi

color(), image()

copyToClipBoard()

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.copyToClipBoard()

// Syntaxe JavaScript
réfObjActeur.copyToClipBoard();
```

Description

Méthode d'acteur ; copie un acteur désigné dans le Presse-papiers.

Il est possible d'appeler cette méthode sans que la fenêtre Distribution soit active.

Vous pouvez utiliser cette méthode pour copier des acteurs entre plusieurs animations ou applications.

Paramètres

Aucun.

Exemple

L'instruction suivante copie l'acteur `Chaise` dans le Presse-papiers :

```
-- Syntaxe Lingo
member("Chaise").copyToClipBoard()

// Syntaxe JavaScript
member("Chaise").copyToClipBoard();
```

L'instruction suivante copie l'acteur numéro 5 dans le Presse-papiers :

```
-- Syntaxe Lingo
member(5).copyToClipBoard()

// Syntaxe JavaScript
member(5).copyToClipBoard();
```

Voir aussi

`Acteur`, `pasteClipBoardInto()`

cos()

Utilisation

```
(angle).cos
cos (angle)
```

Description

Fonction (Lingo uniquement) ; calcule le cosinus de l'angle spécifié, exprimé en radians.

Dans la syntaxe JavaScript, utilisez la fonction `cos()` de l'objet Math.

Paramètres

`angle` Requis. Nombre entier qui spécifie l'angle à tester.

Exemple

L'instruction suivante calcule le cosinus de `pi` sur 2 et l'affiche dans la fenêtre Messages :

```
put (PI/2).cos
```

Voir aussi

`atan()`, `PI`, `sin()`

count()

Utilisation

```
-- Syntaxe Lingo
liste.count
quelObjet.count

// Syntaxe JavaScript
liste.count;
quelObjet.count;
```

Description

Fonction ; renvoie le nombre d'entrées d'une liste linéaire ou de propriétés, le nombre de propriétés d'un script parent sans compter les propriétés d'un script ancêtre ou les sous-chaînes d'une expression texte telles que caractères, lignes ou mots.

La commande `count` fonctionne avec les listes linéaires et de propriétés, les objets créés avec des scripts parents et la propriété `the globals`.

Pour un exemple d'utilisation de `count()` dans une animation, reportez-vous à l'animation Text du dossier Learning/Lingo, lui-même inclus dans le dossier de Director.

Paramètres

Aucun.

Exemple

L'instruction suivante affiche la valeur 3, qui correspond au nombre d'entrées :

```
-- Syntaxe Lingo
put([10,20,30].count) -- 3

// Syntaxe JavaScript
put(list(10,20,30).count); // 3
```

Voir aussi

`globals`

createFile()

Utilisation

```
-- Syntaxe Lingo
réfObjFileio.createFile(chaîneNomDeFichier)

// Syntaxe JavaScript
réfObjFileio.createFile(chaîneNomDeFichier);
```

Description

Méthode de Fileio ; crée un fichier spécifié.

Paramètres

`chaîneNomDeFichier` Requis. Chaîne qui spécifie le chemin et le nom du fichier à créer.

Voir aussi

`Fileio`

createMask()

Utilisation

objetImage.createMask()

Description

La fonction suivante crée et renvoie un objet masque à utiliser avec la fonction copyPixels().

Les objets masque ne sont pas des objets image. Ils ne sont utilisés qu'avec la fonction copyPixels() pour la reproduction de l'effet d'encre Masque. Afin de gagner du temps, si vous envisagez d'utiliser plusieurs fois la même image en tant que masque, il est conseillé de créer l'objet masque et de l'enregistrer dans une variable pour une utilisation ultérieure.

Exemple

L'instruction suivante copie entièrement l'image de l'acteur Joyeux dans un rectangle à l'intérieur de l'acteur Carré marron. L'acteur Dégradé2 est utilisé en tant que masque avec l'image copiée. Le masque est décalé de 10 pixels vers le haut et vers la gauche du rectangle dans lequel l'image Joyeux est collée.

```
member("Carré marron").image.copyPixels(member("Joyeux").image, \
rect(20, 20, 150, 108), member("Joyeux").rect, \
[#maskImage:member("Dégrad2").image.createMask(), maskOffset:point(-10, -10)])
```

Voir aussi

copyPixels(), createMatte(), ink

createMatte()

Utilisation

objetImage.createMatte({*seuilAlpha*})

Description

Cette fonction crée et renvoie un objet Dessin seul que vous pouvez utiliser avec copyPixels() pour reproduire l'effet d'encre Dessin seul. L'objet Dessin seul est créé à partir de la couche alpha de l'objet image spécifiée. Le paramètre facultatif *seuilAlpha* exclut de l'encre Dessin seul tous les pixels dont la valeur de couche alpha est inférieure à ce seuil. Il n'est utilisé qu'avec les images 32 bits contenant une couche alpha. La valeur de *seuilAlpha* doit être comprise entre 0 et 255.

Les objets Dessin seul ne sont pas des objets image. Ils ne sont utilisés qu'avec la fonction copyPixels(). Afin de gagner du temps, si vous envisagez d'utiliser plusieurs fois la même image en tant que Dessin seul, il est conseillé de créer l'objet Dessin seul et de l'enregistrer dans une variable pour une utilisation ultérieure.

Exemple

L'instruction suivante crée un nouvel objet Dessin seul à partir de la couche alpha de l'objet image imageTest et ignore les pixels dont les valeurs alpha sont inférieures à 50 %.

```
nouveauDessinSeul = imageTest.createMatte(128)
```

Voir aussi

copyPixels(), createMask()

crop() (Image)

Utilisation

```
-- Syntaxe Lingo
réfObjImage.crop(rectDeRecadrage)

// Syntaxe JavaScript
réfObjImage.crop(rectDeRecadrage);
```

Description

Méthode d'image. Renvoie un nouvel objet image qui contient une copie d'un objet image source recadré par rapport à un rectangle donné.

L'appel de `crop()` ne modifie pas l'objet image source.

Le nouvel objet image n'appartient plus à aucun acteur et n'est plus associé à la scène. Pour affecter la nouvelle image à un acteur, définissez la propriété `image` de cet acteur.

Paramètres

`rectDeRecadrage` Requis. Rectangle par rapport auquel la nouvelle image est recadrée.

Exemple

L'instruction Lingo suivante prend un cliché de la scène et la redimensionne dans le `rect` de l'image-objet 10, capturant l'apparence courante de cette image-objet sur la scène :

L'instruction suivante utilise le rectangle de l'acteur Joyeux pour recadrer l'image de l'acteur Fleur, puis applique l'image de l'acteur Joyeux au résultat :

```
member("Joyeux").image = member("Fleur").image.crop(member("Joyeux").rect)
```

Voir aussi

`image (image)`, `image()`, `rect (image)`

crop() (Bitmap)

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.crop()

// Syntaxe JavaScript
réfObjActeur.crop();
```

Description

Commande bitmap ; permet de recadrer un acteur bitmap à une taille spécifique.

La commande `crop` permet de recadrer des acteurs existants ou, en combinaison avec l'image de la scène, de saisir un instantané, puis de le recadrer à la taille souhaitée pour l'afficher.

Le point d'alignement reste inchangé de sorte que le bitmap n'est pas déplacé par rapport à sa position d'origine.

Paramètres

rectDeRecadrage Requis. Spécifie le rectangle par rapport auquel la nouvelle image est recadrée.

Exemple

L'instruction suivante affecte un acteur bitmap existant à un instantané de la scène, puis recadre l'image résultante dans un rectangle égal à l'image-objet 10 :

```
-- Syntaxe Lingo
imageDeLaScène = (_movie.stage).image
imageDeLimageObjet = imageDeLaScène.crop(sprite(10).rect)
member("instantané de l'image-objet").image = imageDeLimageObjet

// Syntaxe JavaScript
var imageDeLaScène = (_movie.stage).image
var imageDeLimageObjet = imageDeLaScène.crop(sprite(10).rect);
member("instantané de l'image-objet").image = imageDeLimageObjet;
```

Voir aussi

picture (acteur)

cross

Utilisation

vecteur1.cross(*vecteur2*)

Description

Méthode 3D de vecteur ; renvoie un vecteur perpendiculaire à *vecteur1* et *vecteur2*.

Exemple

Dans l'exemple suivant, pos1 est un vecteur sur l'axe des x et pos2 est un vecteur sur l'axe des y. La valeur renvoyée par `pos1.cross(pos2)` est `vector( 0.0000, 0.0000, 1.00000e4 )`, qui est perpendiculaire à pos1 et pos2.

```
pos1 = vector(100, 0, 0)
pos2 = vector(0, 100, 0)
put pos1.cross(pos2)
-- vector( 0.0000, 0.0000, 1.00000e4 )
```

Voir aussi

crossProduct(), perpendicularTo

crossProduct()

Utilisation

```
vecteur1.crossProduct(vecteur2)
```

Description

Méthode 3D de vecteur ; renvoie un vecteur perpendiculaire à *vecteur1* et *vecteur2*.

Exemple

Dans l'exemple suivant, `pos1` est un vecteur sur l'axe des x et `pos2` est un vecteur sur l'axe des y. La valeur renvoyée par `pos1.crossProduct(pos2)` est `vector( 0.0000, 0.0000, 1.00000e4 )`, qui est perpendiculaire à pos1 et pos2.

```
pos1 = vector(100, 0, 0)
pos2 = vector(0, 100, 0)
put pos1.crossProduct(pos2)
-- vector( 0.0000, 0.0000, 1.00000e4 )
```

Voir aussi

perpendicularTo, cross

cursor()

Utilisation

```
-- Syntaxe Lingo
_player.cursor(entNumDeCurseur)
_player.cursor(numDacteurCurseur, numDacteurMasque)
_player.cursor(réfDacteurCurseur)

// Syntaxe JavaScript
_player.cursor(entNumDeCurseur);
_player.cursor(numDacteurCurseur, numDacteurMasque);
_player.cursor(réfDacteurCurseur);
```

Description

Méthode de lecteur ; change l'acteur ou le curseur intégré utilisé comme pointeur et reste en vigueur jusqu'à ce que vous la désactiviez en affectant au curseur la valeur 0.

- Utilisez la syntaxe `_player.cursor(numDacteurCurseur, numDacteurMasque)` pour spécifier le numéro d'acteur à utiliser comme curseur et son masque facultatif. La zone référencée du curseur correspond au point d'alignement de l'acteur.

  L'acteur spécifié doit être un acteur 1 bit. Si l'acteur est plus grand que 16 x 16 pixels, Director le recadre pour le transformer en carré de 16 x 16 pixels, à partir de l'angle supérieur gauche de l'image. La zone référencée du curseur correspond toujours au point d'alignement de l'acteur.
- Utilisez la syntaxe `_player.cursor(réfDacteurCurseur)` pour les curseurs personnalisés proposés par l'Xtra Cursor.

  Remarque : Bien que l'Xtra Cursor accepte les curseurs de différents types de distributions, les acteurs texte ne peuvent pas être utilisés comme curseurs.

- Utilisez la syntaxe `_player.cursor(entNumDeCurseur)` pour spécifier les curseurs par défaut fournis par le système. Le terme *`entNumDeCurseur`* doit être l'un des nombres entiers suivants :

Valeur	Description
-1, 0	Flèche
1	Curseur en I
2	Croix
3	Croix épaisse
4	Montre (Macintosh) ou sablier (Windows)
5	Nord Sud Est Ouest (NSEO)
6	Nord Sud (NS)
200	Vide (masque le curseur)
254	Aide
256	Crayon
257	Gomme
258	Sélection
259	Pot de peinture
260	Main
261	Outil Rectangle
262	Outil Rectangle arrondi
263	Outil Cercle
264	Outil Ligne
265	Outil RTF
266	Outil Champ de texte
267	Outil Bouton
268	Outil Case à cocher
269	Outil Bouton radio
270	Outil Position
271	Outil Point d'alignement
272	Lasso
280	Doigt
281	Pipette
282	Attente bouton de souris enfoncé 1
283	Attente bouton de souris enfoncé 2
284	Taille verticale

Valeur	Description
285	Taille horizontale
286	Taille diagonale
290	Main fermée
291	Main sans déposer
292	Copie (main fermée)
293	Flèche inversée
294	Rotation
295	Inclinaison
296	Double flèche horizontale
297	Double flèche verticale
298	Double flèche sud-ouest nord-est
299	Double flèche nord-ouest sud-est
300	Pinceau pour enduire/estomper
301	Aérographe
302	Zoom avant
303	Zoom arrière
304	Annulation du zoom
305	Démarrer la forme
306	Ajouter un point
307	Fermer la forme
308	Zoom de la caméra
309	Déplacement de la caméra
310	Rotation de la caméra
457	Personnalisé

Pendant les événements système tels que le chargement d'un fichier, le système d'exploitation peut afficher le curseur montre, puis revenir au curseur pointeur en rendant le contrôle à l'application. Cela supplante les paramètres de la commande `cursor` de l'animation précédente. Pour utiliser la commande `cursor()` au début d'une nouvelle animation chargée dans une présentation utilisant un curseur personnalisé pour plusieurs animations, stockez tout numéro de ressource de curseur spécial sous la forme d'une variable globale qui reste en mémoire entre les animations.

Les commandes de curseur peuvent être interrompues par un Xtra ou tout autre élément externe. Lorsque le curseur a une valeur dans Director et qu'un Xtra ou un élément externe en prend le contrôle, la restitution de sa valeur d'origine n'a aucun effet, Director ne percevant pas que sa valeur a changé. Pour résoudre ce problème, attribuez explicitement au curseur une troisième valeur, puis rendez-lui sa valeur d'origine.

Paramètres

entNumDeCurseur Requis lorsqu'un nombre entier est utilisé pour identifier un curseur. Nombre entier qui spécifie le curseur intégré à utiliser en tant que curseur.

numDacteurCurseur Requis lorsqu'un numéro d'acteur et son masque facultatif sont utilisés pour identifier le curseur. Nombre entier qui spécifie le numéro de l'acteur à utiliser en tant que curseur.

numDacteurMasque Requis lorsqu'un numéro d'acteur et son masque facultatif sont utilisés pour identifier le curseur. Nombre entier qui spécifie le numéro de masque de *numDacteurCurseur*.

réfDacteurCurseur Requis lorsqu'une référence d'acteur est utilisée pour identifier le curseur. Référence à l'acteur à utiliser en tant que curseur.

Exemple

L'instruction suivante change le curseur en curseur montre sur un Macintosh et en sablier sous Windows, si la valeur de la variable status est égale à 1 :

```
-- Syntaxe Lingo
if (status = 1) then
  _player.cursor(4)
end if

// Syntaxe JavaScript
if (status == 1) {
  _player.cursor(4);
}
```

Le gestionnaire suivant vérifie que l'acteur affecté à la variable est un acteur 1 bit et, le cas échéant, l'utilise comme curseur :

```
-- Syntaxe Lingo
on monCurseur(unActeur)
  if (member(UnActeur).depth = 1) then
    _player.cursor(unActeur)
  else
    _sound.beep()
  end if
end

// Syntaxe JavaScript
function monCurseur(unActeur) {
  if (member(unActeur).depth == 1) {
    _player.cursor(unActeur);
  }
  else {
    _sound.beep();
  }
}
```

Voir aussi

Lecteur

date() (formats)

Utilisation

```
-- Syntaxe Lingo
date({chaîneFormat})
date({entFormat})
date({entFormatAnnée, entFormatMois, entFormatJour})

// Syntaxe JavaScript
Date({"mois jj, aaaa hh:mm:ss"});
Date({"mois jj, aaaa"});
Date({aa,mm,jj,hh,mm,ss});
Date({aa,mm,jj});
Date({millisecondes});
```

Description

Fonction du niveau supérieur et type de données. Crée une instance d'objet Date au format standard que vous pouvez utiliser avec d'autres instances de date dans des opérations arithmétiques ou pour manipuler les dates d'une plate-forme à l'autre, ainsi que dans des formats de pays différents.

Les objet Date Lingo et ceux de la syntaxe JavaScript sont différents ; par conséquent, il n'est pas possible de créer des objets Date Lingo avec la syntaxe JavaScript et, inversement, de créer des objets Date JavaScript avec la syntaxe Lingo.

Créez un nouvel objet Date dans la syntaxe JavaScript en utilisant la syntaxe `new Date()`. La syntaxe JavaScript tient compte de la casse. Par exemple, `new date()` provoque une erreur d'exécution.

Lorsque vous créez une date en Lingo, utilisez quatre chiffres pour l'année, deux chiffres pour le mois et deux chiffres pour le jour. Les expressions suivantes renvoient un objet Date correspondant au 21 octobre 2004.

Format de date	Emploi
chaîne	`date("20041021")`
entier	`date(20041021)`
séparation par virgules	`date(2004, 10, 21)`

Les différentes propriétés de l'objet Date renvoyé sont les suivantes :

Propriété	Description
#year	Nombre entier représentant l'année
#month	Nombre entier représentant le mois de l'année
#day	Nombre entier représentant le jour du mois

Les opérations d'addition et de soustraction sur la date sont interprétées comme l'addition et la soustraction de jours.

Paramètres

chaîneFormat Facultatif lors de la création d'un objet Date Lingo. Chaîne qui spécifie le nouvel objet Date.

entFormat Facultatif lors de la création d'un objet Date Lingo. Nombre entier qui spécifie le nouvel objet Date.

entFormatAnnée Facultatif lors de la création d'un objet Date Lingo. Nombre entier qui spécifie les quatre chiffres de l'année du nouvel objet Date.

entFormatMois Facultatif lors de la création d'un objet Date Lingo. Nombre entier qui spécifie les deux chiffres du mois du nouvel objet Date.

entFormatJour Facultatif lors de la création d'un objet Date Lingo. Nombre entier qui spécifie les deux chiffres du jour du nouvel objet Date.

mois Facultatif lors de la création d'un objet Date dans la syntaxe JavaScript. Chaîne qui spécifie le mois du nouvel objet Date. Les valeurs correctes sont comprises entre 0 (janvier) et 11 (décembre).

jj Facultatif lors de la création d'un objet Date dans la syntaxe JavaScript. Nombre entier de deux chiffres qui spécifie le jour du nouvel objet Date. Les valeurs correctes sont comprises entre 0 (dimanche) et 6 (samedi).

aaaa Facultatif lors de la création d'un objet Date dans la syntaxe JavaScript. Nombre entier de quatre chiffres qui spécifie l'année du nouvel objet Date.

hh Facultatif lors de la création d'un objet Date dans la syntaxe JavaScript. Nombre entier de deux chiffres qui spécifie l'heure du nouvel objet Date. Les valeurs correctes sont comprises entre 0 (12:00) et 23 (23:00).

mm Facultatif lors de la création d'un objet Date dans la syntaxe JavaScript. Nombre entier de deux chiffres qui spécifie les minutes du nouvel objet Date. Les valeurs correctes sont comprises entre 0 et 59.

ss Facultatif lors de la création d'un objet Date dans la syntaxe JavaScript. Nombre entier de deux chiffres qui spécifie les secondes du nouvel objet Date. Les valeurs correctes sont comprises entre 0 et 59.

aa Facultatif lors de la création d'un objet Date dans la syntaxe JavaScript. Nombre entier de deux chiffres qui spécifie l'année du nouvel objet Date. Les valeurs correctes sont comprises entre 0 et 99.

millisecondes Facultatif lors de la création d'un objet Date dans la syntaxe JavaScript. Nombre entier qui spécifie les millisecondes du nouvel objet Date. Les valeurs correctes sont comprises entre 0 et 999.

Exemple

Les instructions suivantes créent et déterminent le nombre de jours séparant deux dates :

```
-- Syntaxe Lingo
maDateDeNaissance = date(19650712)
taDateDeNaissance = date(19450529)
put("Il y a" && abs(taDateDeNaissance - maDateDeNaissance) && "jours entre nos
  \ dates de naissances.")

// Syntaxe JavaScript
var maDateDeNaissance = new Date(1965, 07, 12);
var taDateDeNaissance = new Date(1945, 05, 29);
put("Il y a " + Math.abs(((taDateDeNaissance - maDateDeNaissance)/1000/60/60/
  24)) +
  " jours entre nos dates de naissance.");
```

Les instructions suivantes permettent d'accéder à la propriété individuelle d'une date :

```
-- Syntaxe Lingo
maDateDeNaissance = date(19650712)
put("Je suis né au mois numéro"&&maDateDeNaissance.month)

// Syntaxe JavaScript
var maDateDeNaissance = new Date(1965, 07, 12);
put("Je suis né au mois numéro" + maDateDeNaissance.getMonth());
```

date() (Système)

Utilisation

```
-- Syntaxe Lingo
_system.date({aaaammjj})

// Syntaxe JavaScript
_system.date({aaaammjj});
```

Description

Méthode du système ; renvoie la date courante fournie par l'horloge du système.

Le format de date utilisé par Director varie suivant le format défini sur l'ordinateur.

- Sous Windows, vous pouvez personnaliser l'affichage de la date dans le Panneau de configuration Paramètres régionaux. (Windows enregistre le format de date court dans le fichier System.ini. Utilisez cette valeur pour déterminer ce qu'indiquent les parties de la date au format court.)
- Sur Macintosh, vous pouvez personnaliser l'affichage de la date dans le tableau de bord Date et heure.

Paramètres

aaaammjj Facultatif. Nombre qui spécifie les quatre chiffres de l'année (*aaaa*), les deux chiffres du mois (*mm*) et les deux chiffres du jour (*jj*) de la date renvoyée.

Exemple

L'instruction suivante détermine si la date en cours est le 1er janvier en vérifiant si les quatre premiers caractères de la date sont 1/1/. Si c'est le 1er janvier, le message d'alerte Bonne année ! apparaît :

```
-- Syntaxe Lingo
if (_system.date().char[1..4] = "1/1/") then
  _player.alert("Bonne année !")
end if

// Syntaxe JavaScript
if (_system.date().toString().substr(0, 4) == "1/1/") {
  _player.alert("Bonne année !");
}
```

Voir aussi

Système

delay()

Utilisation

```
-- Syntaxe Lingo
_movie.delay(entBattements)

// Syntaxe JavaScript
_movie.delay(entBattements);
```

Description

Commande d'animation ; immobilise la tête de lecture pendant une durée donnée.

La seule activité de la souris et du clavier possible à ce moment-là consiste à arrêter l'animation en appuyant sur Ctrl+Alt+point (Windows) ou Cmd+point (Macintosh). Dans la mesure où elle augmente la durée des différentes images, la commande `delay()` est utile pour contrôler la cadence de lecture d'une séquence d'images.

La méthode `delay()` peut être appliquée seulement lorsque la tête de lecture est en mouvement. Toutefois, les gestionnaires peuvent toujours être exécutés lorsque la méthode `delay()` est active. Seule la tête de lecture s'arrête ; l'exécution des scripts, elle, se poursuit. Placez les scripts utilisant la méthode `delay()` dans un gestionnaire `enterFrame` ou `exitFrame`.

Pour simuler une interruption dans un gestionnaire lorsque la tête de lecture est immobile, utilisez la propriété `milliseconds` de l'objet Système et patientez jusqu'à la fin du délai spécifié avant de sortir de l'image.

Paramètres

`entBattements` Requis. Nombre entier qui spécifie la durée d'immobilisation de la tête de lecture en nombre de battements. Un battement correspond à 1/60ème de seconde.

Exemple

Le gestionnaire suivant interrompt l'animation pendant 2 secondes lorsque la tête de lecture quitte l'image en cours :

```
-- Syntaxe Lingo
on keyDown
  _movie.delay(2*60)
end

// Syntaxe JavaScript
function keyDown() {
  _movie.delay(2*60);
}
```

Le gestionnaire suivant, qui peut être placé dans un script d'image, retarde l'animation d'un nombre aléatoire de battements :

```
-- Syntaxe Lingo
on keyDown
  if (_key.key = "x") then
    _movie.delay(random(180))
  end if
end

// Syntaxe JavaScript
function keyDown() {
  if (_key.key == "x") {
    _movie.delay(random(180));
  }
}
```

Voir aussi

endFrame, milliseconds, Animation

delete()

Utilisation

```
-- Syntaxe Lingo
réfObjFileio.delete()

// Syntaxe JavaScript
réfObjFileio.delete();
```

Description

Méthode de Fileio ; supprime un fichier.

Paramètres

Aucun.

Voir aussi

Fileio

deleteAt

Utilisation

```
liste.deleteAt(nombre)
deleteAt liste, nombre
```

Description

Commande de liste ; supprime un élément d'une liste linéaire ou de propriétés.

La commande `deleteAt` vérifie que l'élément se trouve dans la liste et, si vous essayez de supprimer un objet qui n'est pas dans la liste, Director affiche un message d'erreur.

Paramètres

`nombre` Requis. Spécifie la position de l'élément à supprimer dans la liste.

Exemple

L'instruction suivante supprime le deuxième élément de la liste designers, qui contient [dupont, avatar, soldes] :

```
designers = ["dupont", "avatar", "soldes"]
designers.deleteAt(2)
```

Le résultat est la liste [dupont, soldes].

Le gestionnaire suivant vérifie si un objet se trouve dans la liste avant d'essayer de le supprimer :

```
on monDeleteAt laListe, Lindex
  if laListe.count < Lindex then
    beep
  else
    laListe.deleteAt(Lindex)
  end if
end
```

Voir aussi

`addAt`

deleteCamera

Utilisation

```
member(quelActeur).deleteCamera(nomDeCaméra)
member(quelActeur).deleteCamera(index)
sprite(quelleImageObjet).deleteCamera(caméraOuIndex)
```

Description

Commande 3D ; dans un acteur, cette commande supprime la caméra de l'acteur et de l'univers 3D. Les enfants de la caméra sont retirés de l'univers 3D mais ne sont pas supprimés.

Il est impossible de supprimer la caméra par défaut de l'acteur.

Dans une image-objet, cette commande supprime la caméra de la liste des caméras de l'image-objet. La caméra n'est pas supprimée de l'acteur.

Paramètres

`nomDeCaméra`, *`index`* ou *`caméraOuIndex`*. Requis. Chaîne ou nombre entier qui spécifie le nom ou la position d'index de la caméra à supprimer.

Exemple

L'instruction suivante supprime deux caméras de l'acteur Pièce : tout d'abord la caméra Caméra06, puis la caméra 1.

```
member("Pièce").deleteCamera("Caméra06")
member("Pièce").deleteCamera(1)
```

L'instruction suivante supprime deux caméras de la liste des caméras de l'image-objet 5 : tout d'abord la deuxième caméra de la liste, puis la caméra Caméra06.

```
sprite(5).deleteCamera(2)
sprite(5).deleteCamera(member("Pièce").camera("Caméra06"))
```

Voir aussi

`newCamera`, `addCamera`, `cameraCount()`

deleteFrame()

Utilisation

```
-- Syntaxe Lingo
_movie.deleteFrame()

// Syntaxe JavaScript
_movie.deleteFrame();
```

Description

Méthode d'animation ; supprime l'image courante et fait de l'image suivante la nouvelle image courante pendant une session de création du scénario uniquement.

Paramètres

Aucun.

Exemple

Le gestionnaire suivant vérifie si l'image-objet de la piste 10 de l'image courante a dépassé le bord droit d'une scène de 640 x 480 pixels et, le cas échéant, supprime l'image :

```
-- Syntaxe Lingo
on testSprite
  _movie.beginRecording()
  if (sprite(10).locH > 640) then
    _movie.deleteFrame()
  end if
  _movie.endRecording()
end

// Syntaxe JavaScript
function testSprite() {
  _movie.beginRecording();
  if (sprite(10).locH > 640) {
    _movie.deleteFrame();
  }
  _movie.endRecording();
}
```

Voir aussi

`beginRecording()`, `endRecording()`, `Animation`, `updateFrame()`

deleteGroup

Utilisation

```
member(quelActeur).deleteGroup(quelGroupe)
member(quelActeur).deleteGroup(index)
```

Description

Commande 3D ; supprime le groupe de l'acteur et de l'univers 3D. Les enfants du groupe sont retirés de l'univers 3D mais ne sont pas supprimés.

Il est impossible de supprimer le groupe World, qui est le groupe par défaut.

Paramètres

quelGroupe ou *index*. Requis. Chaîne ou nombre entier qui spécifie le nom ou la position d'index du groupe à supprimer.

Exemple

La première ligne de cet exemple supprime le groupe Factice16 de l'acteur Scène. La deuxième ligne supprime le troisième groupe de Scène.

```
member("Scène").deleteGroup("Factice16")
member("Scène").deleteGroup(3)
```

Voir aussi

newGroup, child (3D), parent

deleteLight

Utilisation

```
member(quelActeur).deleteLight(quelleLumière)
member(quelActeur).deleteLight(index)
```

Description

Commande 3D ; supprime la lumière de l'acteur et de l'univers 3D. Les enfants de la lumière sont retirés de l'univers 3D mais ne sont pas supprimés.

Paramètres

quelleLumière ou *index*. Requis. Chaîne ou nombre entier qui spécifie le nom ou la position d'index de la lumière à supprimer.

Exemple

Les exemples suivants suppriment les lumières de l'acteur Pièce.

```
member("Pièce").deleteLight("lumièreAmbiante")
member("Pièce").deleteLight(6)
```

Voir aussi

newLight

deleteModel

Utilisation

```
member(quelActeur).deleteModel(quelModèle)
member(quelActeur).deleteModel(index)
```

Description

Commande 3D ; supprime le modèle de l'acteur et de l'univers 3D. Les enfants du modèle sont retirés de l'univers 3D mais ne sont pas supprimés.

Paramètres

quelModèle ou *index*. Requis. Chaîne ou nombre entier qui spécifie le nom ou la position d'index du modèle à supprimer.

Exemple

La première ligne de cet exemple supprime le modèle Lecteur3 de l'acteur gbUnivers. La deuxième ligne supprime le neuvième modèle de gbUnivers.

```
member("gbUnivers").deleteModel("Lecteur3")
member("gbUnivers").deleteModel(9)
```

Voir aussi

newModel

deleteModelResource

Utilisation

```
member(quelActeur).deleteModelResource(quelleRessDeMod)
member(quelActeur).deleteModelResource(index)
```

Description

Commande 3D ; supprime la ressource de modèle de l'acteur et de l'univers 3D.

Les modèles qui utilisent la ressource de modèle supprimée deviennent invisibles car ils perdent leur géométrie, mais ne sont ni supprimés ni retirés de l'univers.

Paramètres

quelleRessDeMod ou *index*. Requis. Chaîne ou nombre entier qui spécifie le nom ou la position d'index de la ressource de modèle à supprimer.

Exemple

Les exemples suivants suppriment deux ressources de modèle de l'acteur ScèneDeRue.

```
member("ScèneDeRue").deleteModelResource("MaisonB")
member("ScèneDeRue").deleteModelResource(3)
```

Voir aussi

newModelResource, newMesh

deleteMotion

Utilisation

```
member(quelActeur).deleteMotion(quelMouvement)
member(quelActeur).deleteMotion(index)
```

Description

Commande 3D ; supprime le mouvement de l'acteur.

Paramètres

quelMouvement ou *index*. Requis. Chaîne ou nombre entier qui spécifie le nom ou la position d'index du mouvement à supprimer.

Exemple

La première ligne de l'exemple suivant supprime le mouvement Saut de l'acteur ScèneDePiqueNique. La deuxième ligne supprime le cinquième mouvement de ScèneDePiqueNique.

```
member("ScèneDePiqueNique").deleteMotion("Saut")
member("ScèneDePiqueNique").deleteMotion(5)
```

Voir aussi

`newMotion()`, `removeLast()`

deleteOne

Utilisation

```
liste.deleteOne(valeur)
deleteOne liste, valeur
```

Description

Commande de liste ; supprime une valeur d'une liste linéaire ou de propriétés. Pour une liste de propriétés, `deleteOne` supprime également la propriété associée à la valeur supprimée. Si la valeur se trouve à plusieurs endroits dans la liste, `deleteOne` ne supprime que la première occurrence.

L'utilisation de cette commande pour supprimer une propriété reste sans effet.

Paramètres

valeur Requis. Valeur à supprimer de la liste.

Exemple

La première instruction suivante crée une liste contenant les jours mardi, mercredi et vendredi. La seconde instruction suivante supprime le nom mercredi de la liste.

```
jours = ["mardi", "mercredi", "vendredi"]
jours.deleteOne("mercredi")
put jours
```

L'instruction `put jours` entraîne l'affichage du résultat dans la fenêtre Messages :

```
-- ["mardi", "vendredi"].
```

deleteProp

Utilisation

```
liste.deleteProp(élément)
deleteProp liste, élément
```

Description

Commande de liste ; supprime l'élément spécifié de la liste indiquée.

- Pour les listes linéaires, remplacez *élément* par le numéro correspondant à la position de l'élément à supprimer dans la liste. La commande `deleteProp` a le même effet (dans les listes linéaires) que la commande `deleteAt`. Un chiffre supérieur au nombre d'éléments de la liste entraîne une erreur de script.
- Pour les listes de propriétés, remplacez *élément* par le nom de la propriété à supprimer. La suppression d'une propriété supprime également la valeur qui lui est associée. Si la liste contient la même propriété à plusieurs endroits, seule la première occurrence de la propriété est supprimée.

Paramètres

élément Requis. Elément à supprimer de la liste.

Exemple

L'instruction suivante supprime la propriété color de la liste `[#height:100, #width: 200, #color: 34, #ink: 15]`, appelée attributsDimageObjet :

```
attributsDimageObjet.deleteProp(#color)
```

Le résultat est la liste `[#height:100, #width: 200, #ink: 15]`.

Voir aussi

`deleteAt`

deleteShader

Utilisation

```
member(quelActeur).deleteShader(quelMatériau)
member(quelActeur).deleteShader(index)
```

Description

Commande 3D ; retire le matériau de l'acteur.

Paramètres

quelMatériau ou *index*. Requis. Chaîne ou nombre entier qui spécifie le nom ou la position d'index du matériau à supprimer.

Exemple

La première ligne de cet exemple supprime le matériau Route de l'acteur ScèneDeRue. La deuxième ligne supprime le troisième matériau de ScèneDeRue.

```
member("ScèneDeRue").deleteShader("Route")
member("ScèneDeRue").deleteShader(3)
```

Voir aussi

newShader, shaderList

deleteTexture

Utilisation

```
member(quelActeur).deleteTexture(quelleTexture)
member(quelActeur).deleteTexture(index)
```

Description

Commande 3D ; retire la texture de l'acteur.

Paramètres

quelleTexture ou *index*. Requis. Chaîne ou nombre entier qui spécifie le nom ou la position d'index de la texture à supprimer.

Exemple

La première ligne de cet exemple supprime la texture Ciel de l'acteur ScèneDePiqueNique. La deuxième ligne supprime la cinquième texture de ScèneDePiqueNique.

```
member("ScèneDePiqueNique).deleteTexture("Ciel")
member("ScèneDePiqueNique").deleteTexture(5)
```

Voir aussi

newTexture

deleteVertex()

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.deleteVertex(indexASupprimer)

// Syntaxe JavaScript
réfObjActeur.deleteVertex(indexASupprimer);
```

Description

Commande de forme vectorielle ; supprime un sommet existant d'un acteur forme vectorielle à la position d'index spécifiée.

Paramètres

indexASupprimer Requis. Nombre entier qui spécifie la position d'index du sommet à supprimer.

Exemple

La ligne suivante supprime le deuxième point de sommet de la forme vectorielle Archie :

```
-- Syntaxe Lingo
member("Archie").deleteVertex(2)

// Syntaxe JavaScript
member("Archie").deleteVertex(2);
```

Voir aussi

addVertex(), moveVertex(), originMode, vertexList

displayOpen()

Utilisation

```
-- Syntaxe Lingo
réfObjFileio.displayOpen()

// Syntaxe JavaScript
réfObjFileio.displayOpen();
```

Description

Méthode de Fileio ; affiche une boîte de dialogue d'ouverture.

Cette méthode renvoie au script le chemin d'accès complet et le nom du fichier sélectionné.

Paramètres

Aucun.

Voir aussi

Fileio

displaySave()

Utilisation

```
-- Syntaxe Lingo
réfObjFileio.displaySave(chaîneTitre, chaîneNomDeFichier)

// Syntaxe JavaScript
réfObjFileio.displaySave(chaîneTitre, chaîneNomDeFichier);
```

Description

Méthode de Fileio ; affiche une boîte de dialogue d'enregistrement.

Cette méthode renvoie au script le chemin d'accès complet et le nom du fichier enregistré.

Paramètres

chaîneTitre. Requis. Chaîne qui représente le titre affiché dans la boîte de dialogue d'enregistrement.

chaîneNomDeFichier. Requis. Chaîne qui spécifie le chemin complet et le nom du fichier à enregistrer.

Voir aussi

Fileio

do

Utilisation

```
do expressionChaîne
```

Description

Commande ; évalue une chaîne et exécute le résultat sous la forme d'une instruction de script. Cette commande est pratique pour évaluer des expressions saisies par l'utilisateur et pour exécuter des commandes placées dans des variables chaînes, des champs, des listes et des fichiers.

L'utilisation de variables locales non initialisées dans une commande `do` entraîne une erreur de compilation. Il est conseillé d'initialiser à l'avance les variables locales.

Remarque : Cette commande ne permet pas la déclaration de variables globales, celles-ci devant toujours être déclarées en avance.

La commande `do` fonctionne avec des chaînes à plusieurs lignes ainsi qu'avec les chaînes n'en contenant qu'une seule.

Paramètres

expressionChaîne Requis. Chaîne à évaluer.

Exemple

L'instruction suivante exécute l'instruction placée entre guillemets :

```
do "beep 2"
do listeDeCommandes[3]
```

doneParsing()

Utilisation

```
ObjetDanalyse.doneParsing()
```

Description

Fonction ; renvoie `1 (TRUE)` lorsque l'objet d'analyse a terminé l'analyse du document avec `parseURL()`. La valeur renvoyée est `0 (FALSE)` jusqu'à la fin de l'analyse.

Paramètres

Aucun.

Voir aussi

parseURL()

dot()

Utilisation

vecteur1.dot(*vecteur2*)

Description

Méthode 3D de vecteur ; renvoie la somme des produits des composants x, y et z de deux vecteurs. Si les deux vecteurs sont normalisés, le produit est le cosinus de l'angle entre les deux vecteurs.

Pour manuellement arriver au dot de deux vecteurs, multipliez le composant x de `vecteur1` par le composant x de `vecteur2`, puis multipliez le composant y de `vecteur1` par le composant y de `vecteur2`, puis multipliez le composant z de `vecteur1` par le composant z de `vecteur2`, avant de finalement ajouter les trois produits.

Cette méthode est identique à la fonction `dotProduct()`.

Paramètres

vecteur2 Requis. Le second des vecteurs dont la somme est renvoyée.

Exemple

Dans l'exemple suivant, l'angle séparant les vecteurs pos5 et pos6 est de 45°. La fonction `getNormalized` renvoie les valeurs normalisées de pos5 et pos6 et les enregistre dans les variables norm1 et norm2. Le produit `dot` de norm1 et norm2 est 0,7071, ce qui correspond au cosinus de 45°.

```
pos5 = vector(100, 100, 0)
pos6 = vector(0, 100, 0)
put pos5.angleBetween(pos6)
-- 45.0000
norm1 = pos5.getNormalized()
put norm1
-- vector( 0.7071, 0.7071, 0.0000 )
norm2 = pos6.getNormalized()
put norm2
-- vector( 0.0000, 1.0000, 0.0000 )
put norm1.dot(norm2)
-- 0.7071
```

Voir aussi

`dotProduct()`, `getNormalized`, `normalize`

dotProduct()

Utilisation

vecteur1.dotProduct(*vecteur2*)

Description

Méthode 3D de vecteur ; renvoie la somme des produits des composants x, y et z de deux vecteurs. Si les deux vecteurs sont normalisés, `dotproduct` renvoie le cosinus de l'angle entre les deux vecteurs.

Pour manuellement arriver au dot de deux vecteurs, multipliez le composant x de `vecteur1` par le composant x de `vecteur2`, puis multipliez le composant y de `vecteur1` par le composant y de `vecteur2`, puis multipliez le composant z de `vecteur1` par le composant z de `vecteur2`, avant de finalement ajouter les trois produits.

Cette méthode est identique à la fonction `dot()`.

Paramètres

`vecteur2` Requis. Le second des vecteurs dont la somme est renvoyée.

Exemple

Dans l'exemple suivant, l'angle séparant les vecteurs `pos5` et `pos6` est de 45°. La fonction `getNormalized` renvoie les valeurs normalisées de `pos5` et `pos6` et les enregistre dans les variables `norm1` et `norm2`. Le `dotProduct` de `norm1` et `norm2` est 0.7071, ce qui est le cosinus de 45°.

```
pos5 = vector(100, 100, 0)
pos6 = vector(0, 100, 0)
put pos5.angleBetween(pos6)
-- 45.0000
norm1 = pos5.getNormalized()
put norm1
-- vector( 0.7071, 0.7071, 0.0000 )
norm2 = pos6.getNormalized()
put norm2
-- vector( 0.0000, 1.0000, 0.0000 )
put norm1.dotProduct(norm2)
-- 0.7071
```

Voir aussi

`dot()`, `getNormalized`, `normalize`

downloadNetThing

Utilisation

`downloadNetThing` *`URL`*, *`fichierLocal`*

Description

Commande ; copie un fichier depuis Internet vers un fichier placé sur le disque local, tandis que la lecture de l'animation courante continue. Utilisez `netDone` pour vérifier si le téléchargement est terminé.

Les animations Director en mode auteur et les projections prennent en charge la commande `downLoadNetThing`, mais celle-ci n'est pas prise en charge par Shockwave Player. Cela empêche les utilisateurs de copier accidentellement des fichiers à partir d'Internet.

Bien que plusieurs opérations réseau puissent avoir lieu en même temps, l'exécution de plus de quatre opérations simultanées réduit généralement les performances de façon inacceptable.

Ni la taille de la mémoire cache de l'animation Director, ni le paramètre de l'option Vérifier les documents n'affectent le comportement de la commande `downloadNetThing`.

Paramètres

URL Requis. URL de n'importe quel objet pouvant être téléchargé : par exemple, un serveur FTP ou HTTP, une page HTML, un acteur externe, une animation Director ou un graphique.

fichierLocal Requis. Chemin d'accès et nom du fichier sur le disque local.

Exemple

Les instructions suivantes téléchargent un acteur externe depuis une URL vers le dossier de Director et font de ce fichier l'acteur externe intitulé Distribution importante :

```
downLoadNetThing("http://www.cbDeMille.com/Milliers.cst", the \
  applicationPath&"Milliers.Cst")
castLib("Distribution importante").fileName = the
  applicationPath&"Milliers.Cst"
```

Voir aussi

importFileInto(), netDone(), preloadNetThing()

draw()

Utilisation

```
-- Syntaxe Lingo
réfObjImage.draw(x1, y1, x2, y2, objetCouleurOuListedeParamètres)
réfObjImage.draw(point(x, y), point(x, y), objetCouleurOuListedeParamètres)
réfObjImage.draw(rect, objetCouleurOuListedeParamètres)

// Syntaxe JavaScript
réfObjImage.draw(x1, y1, x2, y2, objetCouleurOuListedeParamètres);
réfObjImage.draw(point(x, y), point(x, y), objetCouleurOuListedeParamètres);
réfObjImage.draw(rect, objetCouleurOuListedeParamètres);
```

Description

Méthode d'image. Dessine une ligne ou une forme vide avec une couleur spécifiée dans une zone rectangulaire d'un objet image donné.

Cette méthode renvoie une valeur égale à 1 si aucune erreur n'est détectée.

Si la liste des paramètres facultatifs n'est pas fournie, `draw()` dessine une ligne de 1 pixel entre le premier et le second point donnés ou entre l'angle supérieur gauche et l'angle inférieur droit du rectangle donné.

Pour optimiser les performances avec des images 8 bits (ou d'une qualité inférieure), l'objet Couleur doit contenir une valeur de couleur indexée. Pour les images 16 ou 32 bits, utilisez une valeur de couleur rvb.

Pour remplir une zone unie, utilisez la méthode `fill()`.

Paramètres

x1 Requis si vous dessinez une ligne avec des coordonnées `x` et `y`. Entier qui spécifie la coordonnée `x` du début de la ligne.

y1 Requis si vous dessinez une ligne avec des coordonnées `x` et `y`. Entier qui spécifie la coordonnée `y` du début de la ligne.

x2 Requis si vous dessinez une ligne avec des coordonnées `x` et `y`. Entier qui spécifie la coordonnée `x` de la fin de la ligne.

y2 Requis si vous dessinez une ligne avec des coordonnées `x` et `y`. Entier qui spécifie la coordonnée `y` de la fin de la ligne.

objCouleurOuListeDeParamètres Requis. Objet couleur ou liste de paramètres qui spécifie la couleur de la ligne ou de la bordure de la forme. Il est possible d'utiliser la liste de paramètres à la place d'un simple objet Couleur pour spécifier les propriétés suivantes.

Propriété	Description
`#shapeType`	Une valeur de symbole de `#oval`, `#rect`, `#roundRect` ou `#line`. La valeur par défaut est #line.
`#lineSize`	Epaisseur du trait à utiliser pour le dessin de la forme.
`#color`	Un objet couleur, qui détermine la couleur de la bordure de la forme.

`point(x, y), point(x, y)` Requis si vous dessinez une ligne avec des points. Deux points qui spécifient le début et la fin de la ligne.

rect Requis si vous dessinez une forme. Rectangle qui spécifie la zone rectangulaire dans laquelle la forme est dessinée.

Exemple

L'instruction suivante dessine une ligne diagonale de 1 pixel, de couleur rouge foncé, du point (0, 0) au point (128, 86) dans l'image de l'acteur Joyeux.

L'instruction suivante dessine un ovale vide de 3 pixels, rouge foncé, dans l'image de l'acteur Joyeux. L'ovale est dessiné à l'intérieur du rectangle (0, 0, 128, 86).

Voir aussi

`color()`, `copyPixels()`, `fill()`, `image()`, `setPixel()`

duplicate() (image)

Utilisation

```
-- Syntaxe Lingo
réfObjImage.duplicate()

// Syntaxe JavaScript
réfObjImage.duplicate();
```

Description

Méthode d'image. Crée et renvoie une copie d'une image donnée.

La nouvelle image est entièrement indépendante de l'image d'origine et n'est liée à aucun acteur. Si vous projetez d'apporter un grand nombre de modifications à une image, il est recommandé d'en faire une copie indépendante d'un acteur.

Paramètres

Aucun.

Exemple

L'instruction suivante crée un nouvel objet image à partir de l'image de l'acteur Surface lunaire et place le nouvel objet image dans la variable `imageTemporaire` :

```
imageTemporaire = member("Surface lunaire").image.duplicate()
```

Voir aussi

`image()`

duplicate() (fonction de liste)

Utilisation

```
(ancienneListe).duplicate()
duplicate(ancienneListe)
```

Description

Fonction de liste ; renvoie la copie d'une liste et copie des listes imbriquées (éléments de listes qui sont eux-mêmes des listes) et leur contenu. Cette fonction est pratique pour enregistrer le contenu courant d'une liste.

Lorsque vous affectez une liste à une variable, la variable contient une référence à la liste et non la liste même. Cela signifie que les modifications apportées à la copie affectent également l'original.

Pour un exemple d'utilisation de `duplicate() (fonction de liste)` dans une animation, reportez-vous à l'animation Vector Shapes du dossier Learning/Lingo, lui-même inclus dans le dossier de Director.

Paramètres

`ancienneListe` Requis. Spécifie la liste à dupliquer.

Exemple

Cette instruction copie la liste ClientsDuJour et l'affecte à la variable `ListeDesClients` :

```
ListeDesClients = ClientsDuJour.duplicate()
```

Voir aussi

`image()`

duplicate() (acteur)

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.duplicate({entPosn})

// Syntaxe JavaScript
réfObjActeur.duplicate({entPosn});
```

Description

Méthode d'acteur ; crée une copie d'un acteur spécifié.

Cette méthode est destinée à être utilisée pendant la programmation plutôt que pendant l'exécution, car elle crée un autre acteur en mémoire, ce qui risque d'entraîner des problèmes de mémoire.

Utilisez cette méthode pour enregistrer définitivement les modifications de l'acteur avec le fichier.

Paramètres

entPosn Facultatif. Nombre entier qui spécifie la fenêtre Distribution pour l'acteur copié. Si le paramètre est omis, la copie de l'acteur est placée dans la première position disponible de la fenêtre Distribution.

Exemple

L'instruction suivante copie l'acteur Bureau et le place dans la première position vide de la fenêtre Distribution :

```
-- Syntaxe Lingo
member("Bureau").duplicate()

// Syntaxe JavaScript
member("Bureau").duplicate();
```

L'instruction suivante copie l'acteur Bureau et le place à la position 125 :

```
-- Syntaxe Lingo
member("Bureau").duplicate(125)

// Syntaxe JavaScript
member("Bureau").duplicate(125);
```

Voir aussi

Acteur

duplicateFrame()

Utilisation

```
-- Syntaxe Lingo
_movie.duplicateFrame()

// Syntaxe JavaScript
_movie.duplicateFrame();
```

Description

Méthode d'animation ; copie l'image courante et son contenu, insère la copie après l'image courante et fait de la copie l'image courante. Cette méthode peut être utilisée pendant la création du scénario uniquement.

Cette méthode a la même fonction que la méthode `insertFrame()`.

Paramètres

Aucun.

Exemple

Lorsqu'elle est utilisée dans le gestionnaire suivant, la commande `duplicateFrame` crée une série d'images dans lesquelles l'acteur Balle de la distribution externe Jouets est affecté à la piste d'image-objet 20. Le nombre d'images est déterminé par l'argument `nombreDimages`.

```
-- Syntaxe Lingo
on animBalle(nombreDimages)
  _movie.beginRecording()
  sprite(20).member = member("Balle", "Jouets")
  repeat with i = 0 to nombreDimages
    _movie.duplicateFrame()
  end repeat
  _movie.endRecording()
end animBalle

// Syntaxe JavaScript
function animBalle(nombreDimages) {
  _movie.beginRecording();
  sprite(20).member = member("Balle", "Jouets");
  for (var i = 0; i <= nombreDimages; i++) {
    _movie.duplicateFrame();
  }
  _movie.endRecording();
}
```

Voir aussi

`insertFrame()`, `Animation`

enableHotSpot()

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.enableHotSpot(idDeZoneRéférencée, trueOrFalse)

// Syntaxe JavaScript
réfObjImageObjet.enableHotSpot(idDeZoneRéférencée, trueOrFalse);
```

Description

Commande QTVR (QuickTime VR) ; détermine si une zone référencée dans une image-objet QTVR spécifiée est activée (`TRUE`) ou désactivée (`FALSE`).

Paramètres

`idDeZoneRéférencée` Requis. Spécifie la zone référencée dans l'image-objet QTVR à tester.

`trueOuFalse` Requis. Valeur `TRUE` ou `FALSE` qui spécifie si l'image-objet QTVR est activée.

endRecording()

Utilisation

```
-- Syntaxe Lingo
_movie.endRecording()

// Syntaxe JavaScript
_movie.endRecording();
```

Description

Méthode d'animation ; termine une session de mise à jour du scénario.

Vous pouvez reprendre le contrôle des pistes du scénario en créant des scripts après avoir appelé `endRecording()`.

Paramètres

Aucun.

Exemple

Lorsque utilisé dans le gestionnaire suivant, le mot-clé `endRecording` termine la session de création de scénario :

```
-- Syntaxe Lingo
on animBalle(nombreDimages)
  _movie.beginRecording()
  horizontal = 0
  vertical = 100
  repeat with i = 1 to nombreDimages
    _movie.go(i)
    sprite(20).member = member("Balle")
    sprite(20).locH = horizontal
    sprite(20).locV = vertical
    sprite(20).forecolor = 255
    horizontal = horizontal + 3
    vertical = vertical + 2
    _movie.updateFrame()
  end repeat
  _movie.endRecording()
end animBalle
```

```
// Syntaxe JavaScript
function animBalle(nombreDimages) {
  _movie.beginRecording();
  var horizontal = 0;
  var vertical = 100;
  for (var i = 1; i <= nombreDimages; i++) {
    _movie.go(1);
    sprite(20).member = member("Balle");
    sprite(20).locH = horizontal;
    sprite(20).locV = vertical;
    sprite(20).foreColor = 255;
    horizontal = horizontal + 3;
    vertical = vertical + 2;
    _movie.updateFrame();
  }
  _movie.endRecording();
}
```

Voir aussi

beginRecording(), Animation, updateFrame()

erase()

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.erase()

// Syntaxe JavaScript
réfObjActeur.erase();
```

Description

Méthode d'acteur ; supprime l'acteur spécifié et laisse son emplacement vide dans la fenêtre Distribution.

Pour de meilleurs résultats, utilisez cette méthode pendant la programmation, et non dans les projections, ce qui pourrait provoquer des problèmes de mémoire.

Paramètres

Aucun.

Exemple

L'instruction suivante supprime l'acteur Engrenage de la distribution Matériel :

```
-- Syntaxe Lingo
member("Engrenage", "Matériel").erase()

// Syntaxe JavaScript
member("Engrenage", "Matériel").erase();
```

Le gestionnaire suivant supprime les acteurs du `début` à la `fin` :

```
-- Syntaxe Lingo
on effacerLesActeurs début, fin
  repeat with i = début to fin
    member(i).erase()
  end repeat
end effacerLesActeurs

// Syntaxe JavaScript
function effacerLesActeurs(début, fin) {
  for (var i=début; i<=fin; i++) {
    member(i).erase();
  }
}
```

Voir aussi

`Acteur`, `new()`

error()

Utilisation

```
-- Syntaxe Lingo
réfObjFileio.error(entErreur)

// Syntaxe JavaScript
réfObjFileio.error(entErreur);
```

Description

Méthode de Fileio ; renvoie un message d'erreur spécifié.

Paramètres

entErreur Requis. Nombre entier qui spécifie l'erreur. Les valeurs correctes incluent 0 ("OK") et 1 ("Memory allocation failure"). Toutes les autres valeurs renvoient "Unknown error".

Voir aussi

`Fileio`

externalEvent()

Utilisation

```
externalEvent "chaîne"
```

Description

Commande ; envoie une chaîne au navigateur web pour qu'il l'interprète comme une instruction de script, permettant la lecture d'une animation ou la communication avec la page HTML dans laquelle elle est intégrée.

Cette commande fonctionne uniquement pour les animations dans des navigateurs.

Remarque : La commande `externalEvent` ne produit pas de valeur de renvoi. Il n'existe aucune façon immédiate permettant de déterminer si le navigateur a traité l'événement ou l'a ignoré. Utilisez `on EvalScript` dans le navigateur pour renvoyer un message à l'animation.

Paramètres

chaîne Requis. Chaîne à envoyer au navigateur. La chaîne doit être dans un langage de programmation pris en charge par le navigateur.

Exemple

Les instructions suivantes utilisent `externalEvent` dans l'environnement script LiveConnect, supporté par Netscape 3.x et les versions plus récentes.

LiveConnect évalue la chaîne passée par `externalEvent` comme un appel de fonction. Les auteurs utilisant JavaScript doivent définir et nommer cette fonction dans l'en-tête HTML. Dans l'animation, le nom et les paramètres de la fonction sont définis comme une chaîne dans `externalEvent`. Les paramètres devant être interprétés par le navigateur web comme des chaînes distinctes, chaque paramètre est encadré de guillemets droits simples.

Instructions dans le code HTML :

```
function maFonction(param1, param2) {
  //script placé ici
}
```

Instructions dans un script dans l'animation :

```
externalEvent ("maFonction ('param1','param2')")
```

Les instructions suivantes utilisent `externalEvent` dans l'environnement script ActiveX utilisé par Internet Explorer sous Windows. ActiveX interprète `externalEvent` comme un événement et traite cet événement et son paramètre de chaîne de la même façon qu'un événement `onClick` dans un objet bouton.

- Instructions dans le code HTML :

Dans l'en-tête HTML, définissez une fonction repérant l'événement (l'exemple suivant est en VBScript) :

```
Sub
nomDeLinstanceShockwave_externalEvent(unParamètre)
  'script placé ici
End Sub
```

Vous pouvez également définir un script pour l'événement :

```
<SCRIPT FOR="nomDeLinstanceShockwave"
EVENT="externalEvent(unParamètre)"
LANGUAGE="VBScript">
  'script placé ici
</SCRIPT>
```

Dans l'animation, incluez la fonction et les paramètres dans la chaîne `externalEvent` :

```
externalEvent ("maFonction ('param1','param2')")
```

Voir aussi

`on EvalScript`

extrude3D

Utilisation

```
member(quelActeurTexte).extrude3D(member(quelActeur3D))
```

Description

Commande 3D ; crée une ressource de modèle `#extruder` dans un acteur 3D à partir du texte d'un acteur texte.

Cela n'équivaut pas à utiliser la propriété 3D `displayMode` d'un acteur texte.

Pour créer un modèle avec extrude3D :

1 Créez une ressource de modèle `#extruder` dans un acteur 3D :

```
ressourceTexte = member("acteurTexte").extrude3D(member\
  ("acteur3D"))
```

2 Créez un modèle à l'aide de la ressource de modèle créée à l'étape 1 :

```
member("acteur3D").newModel("monTexte", ressourceTexte)
```

Paramètres

quelActeur3D Requis. Acteur dans lequel une nouvelle ressource de modèle `#extruder` est créée.

Exemple

Dans l'exemple suivant, Logo est un acteur texte et Scène est un acteur 3D. La première ligne crée une ressource de modèle dans Scène, qui est une version 3D du texte de Logo. La seconde ligne utilise cette ressource de modèle pour créer un modèle nommé logo3D.

```
maRessourceDeModèleTexte = member("Logo").extrude3d(member("Scène"))
member("Scène").newModel("Logo3D", maRessourceDeModèleTexte)
```

Voir aussi

`bevelDepth`, `bevelType`, `displayFace`, `smoothness`, `tunnelDepth`, `displayMode`

externalParamName()

Utilisation

```
-- Syntaxe Lingo
_player.externalParamName(nomOuNumDeParam)

// Syntaxe JavaScript
_player.externalParamName(nomOuNumDeParam);
```

Description

Méthode de lecteur ; renvoie le nom d'un paramètre spécifié dans la liste de paramètres externes d'une balise HTML <EMBED> ou <OBJECT>.

Si vous spécifiez un paramètre par son nom, cette méthode renvoie tous les noms de paramètres qui correspondent à *nomOuNumDeParam*. La correspondance ne fait pas la distinction entre les majuscules et les minuscules. Si aucun paramètre correspondant n'est trouvé, cette méthode renvoie `VOID` (Lingo) ou `null` (syntaxe JavaScript).

Si vous spécifiez un paramètre par son numéro, cette méthode renvoie le nom de paramètre situé à la position *nomOuNumDeParam* dans la liste des paramètres. Si aucun paramètre correspondant n'est trouvé, cette méthode renvoie `VOID` ou `null`.

Cette méthode est valide uniquement pour les animations avec un contenu Shockwave exécutées dans un navigateur. Elle ne peut pas être utilisée avec des animations Director ou des projections.

La liste suivante décrit les paramètres externes prédéfinis qui peuvent être utilisés.

Paramètre	Définition
`swAudio`	Chaîne qui spécifie l'emplacement d'un fichier audio Shockwave à lire avec l'animation. Cette valeur est une URL complète.
`swBackColor`	Valeur de couleur destinée à modifier la propriété de couleur associée à la scène de l'animation. Cette valeur est un nombre entier compris entre 0 et 255. Utilisez la plage 0-255 pour les animations en couleur 8 bits et la plage 0-15 pour les animations en couleur 4 bits.
`swBanner`	Chaîne qui spécifie le texte à utiliser en tant que bandeau dans l'animation.
`swColor`	Valeur de couleur à utiliser pour modifier la couleur d'un objet spécifique. Cette valeur est un nombre entier compris entre 0 et 255. Utilisez la plage 0-255 pour les animations en couleur 8 bits et la plage 0-15 pour les animations en couleur 4 bits.
`swForeColor`	Nouvelle valeur de couleur de premier plan. Le texte écrit dans les acteurs champ est rendu dans la couleur de premier plan actuellement active. Cette valeur est un nombre entier compris entre 0 et 255. Utilisez la plage 0-255 pour les animations en couleur 8 bits et la plage 0-15 pour les animations en couleur 4 bits.
`swFrame`	Valeur de chaîne correspondant au nom attribué à une image donnée de l'animation.
`swList`	Liste d'éléments séparés par des virgules qui peut être analysée avec un script. Les valeurs de la liste peuvent être des paires clé/valeur, des éléments booléens, des nombres entiers ou des chaînes.
`swName`	Nom (par exemple, un nom d'utilisateur) à afficher ou utiliser dans l'animation.
`swPassword`	Mot de passe, qui doit éventuellement être combiné avec la propriété `swName`, à utiliser dans l'animation.
`swPreloadTime`	Nombre entier qui indique combien de secondes doivent s'écouler entre le préchargement d'un fichier audio et le début de la lecture du son. Utilisé avec Shockwave Audio pour améliorer les performances de la lecture en augmentant la quantité de son déjà téléchargée avant qu'elle ne commence.
`swSound`	Chaîne qui peut soit spécifier le nom d'un son à lire dans l'animation Director soit indiquer si un son doit ou non être lu.
`swText`	Valeur (chaîne) qui spécifie le texte à utiliser dans l'animation.
`swURL`	URL (chaîne) qui peut spécifier l'emplacement d'une autre animation avec un contenu Shockwave ou d'un autre fichier Shockwave Audio.

Paramètre	Définition
`swVolume`	Nombre entier (valeur comprise entre 0 et 10 recommandée) utilisé pour contrôler le niveau du volume de la sortie audio de l'animation. Une valeur égale à 0 désactive le son, une valeur égale à 10 correspond au volume maximum.
`sw1` à `sw9`	Neuf propriétés supplémentaires pour les paramètres définis par l'auteur.

Paramètres

`nomOuNumDeParam` Requis. Chaîne qui spécifie le nom du paramètre à renvoyer ou nombre entier qui spécifie sa position d'index.

Exemple

L'instruction suivante place la valeur d'un paramètre externe donné dans la variable `maVariable` :

```
-- Syntaxe Lingo
if (_player.externalParamName("swURL") = "URLsw") then
  maVariable = _player.externalParamName("URLsw")
end if

// Syntaxe JavaScript
if (_player.externalParamName("URLsw") == "URLsw") {
  var maVariable = _player.externalParamName("URLsw");
}
```

Voir aussi

`externalParamValue()`, `Animation`

externalParamValue()

Utilisation

```
-- Syntaxe Lingo
_player.externalParamValue(nomOuNumDeParam)

// Syntaxe JavaScript
_player.externalParamValue(nomOuNumDeParam);
```

Description

Renvoie la valeur d'un paramètre spécifié dans la liste de paramètres externes d'une balise HTML <EMBED> ou <OBJECT>.

Si vous spécifiez un paramètre par son nom, cette méthode renvoie la valeur du premier paramètre dont le nom correspond à *`nomOuNumDeParam`*. La correspondance ne fait pas la distinction entre les majuscules et les minuscules. Si aucun paramètre correspondant n'est trouvé, cette méthode renvoie `VOID` (Lingo) ou `null` (syntaxe JavaScript).

Si vous spécifiez un paramètre par son numéro d'index, cette méthode renvoie le nom de paramètre situé à la position *`nomOuNumDeParam`* dans la liste des paramètres. Si aucun paramètre correspondant n'est trouvé, cette méthode renvoie `VOID` ou `null`.

Cette méthode est valide uniquement pour les animations avec un contenu Shockwave exécutées dans un navigateur. Elle ne peut pas être utilisée avec des animations Director ou des projections.

La liste suivante décrit les paramètres externes prédéfinis qui peuvent être utilisés.

Paramètre	Définition
`swAudio`	Chaîne qui spécifie l'emplacement d'un fichier audio Shockwave à lire avec l'animation. Cette valeur est une URL complète.
`swBackColor`	Valeur de couleur destinée à modifier la propriété de couleur associée à la scène de l'animation. Cette valeur est un nombre entier compris entre 0 et 255. Utilisez la plage 0-255 pour les animations en couleur 8 bits et la plage 0-15 pour les animations en couleur 4 bits.
`swBanner`	Chaîne qui spécifie le texte à utiliser en tant que bandeau dans l'animation.
`swColor`	Valeur de couleur à utiliser pour modifier la couleur d'un objet spécifique. Cette valeur est un nombre entier compris entre 0 et 255. Utilisez la plage 0-255 pour les animations en couleur 8 bits et la plage 0-15 pour les animations en couleur 4 bits.
`swForeColor`	Nouvelle valeur de couleur de premier plan. Le texte écrit dans les acteurs champ est rendu dans la couleur de premier plan actuellement active. Cette valeur est un nombre entier compris entre 0 et 255. Utilisez la plage 0-255 pour les animations en couleur 8 bits et la plage 0-15 pour les animations en couleur 4 bits.
`swFrame`	Valeur de chaîne correspondant au nom attribué à une image donnée de l'animation.
`swList`	Liste d'éléments séparés par des virgules qui peut être analysée avec un script. Les valeurs de la liste peuvent être des paires clé/valeur, des éléments booléens, des nombres entiers ou des chaînes.
`swName`	Nom (par exemple, un nom d'utilisateur) à afficher ou utiliser dans l'animation.
`swPassword`	Mot de passe, qui doit éventuellement être combiné avec la propriété `swName`, à utiliser dans l'animation.
`swPreloadTime`	Nombre entier qui indique combien de secondes doivent s'écouler entre le préchargement d'un fichier audio et le début de la lecture du son. Utilisé avec Shockwave Audio pour améliorer les performances de la lecture en augmentant la quantité de son déjà téléchargée avant qu'elle ne commence.
`swSound`	Chaîne qui peut soit spécifier le nom d'un son à lire dans l'animation Director soit indiquer si un son doit ou non être lu.
`swText`	Valeur (chaîne) qui spécifie le texte à utiliser dans l'animation.
`swURL`	URL (chaîne) qui peut spécifier l'emplacement d'une autre animation Shockwave ou d'un autre fichier Shockwave Audio.
`swVolume`	Nombre entier (valeur comprise entre 0 et 10 recommandée) utilisé pour contrôler le niveau du volume de la sortie audio de l'animation. Une valeur égale à 0 désactive le son, une valeur égale à 10 correspond au volume maximum.
`sw1` à `sw9`	Neuf propriétés supplémentaires pour les paramètres définis par l'auteur.

Paramètres

nomOuNumDeParam Requis. Chaîne qui spécifie le nom de la valeur de paramètre à renvoyer ou nombre entier qui spécifie sa position d'index.

Exemple

L'instruction suivante place la valeur d'un paramètre externe dans la variable `maVariable` :

```
-- Syntaxe Lingo
if (_player.externalParamName("swURL") = "URLsw") then
  maVariable = _player.externalParamValue("URLsw")
end if

// Syntaxe JavaScript
if (_player.externalParamName("URLsw") == "URLsw") {
  var maVariable = _player.externalParamValue("URLsw");
}
```

Voir aussi

externalParamName(), Animation

extractAlpha()

Utilisation

objetImage.extractAlpha()

Description

Cette fonction copie la couche alpha de l'image 32 bits donnée et la renvoie sous forme d'un nouvel objet image. Il en résulte une image 8 bits faite de niveaux de gris, représentant la couche alpha.

Cette fonction s'avère pratique pour le sous-échantillonnage des images 32 bits avec des couches alpha.

Exemple

L'instruction suivante place la couche alpha de l'image de l'acteur 1 dans la variable `alphaPrincipal` :

```
alphaPrincipal = member(1).image.extractAlpha()
```

setAlpha(), useAlpha

fadeIn()

Utilisation

```
-- Syntaxe Lingo
réfObjPisteAudio.fadeOut({entMillisecondes})

// Syntaxe JavaScript
réfObjPisteAudio.fadeOut({entMillisecondes});
```

Description

Méthode de piste audio ; définit immédiatement le `volume` de la piste audio sur zéro pour le ramener ensuite sur le volume courant en fonction du nombre de millisecondes défini.

Le paramètre de balance courant est conservé pour l'ensemble du fondu.

Paramètres

entMillisecondes Facultatif. Nombre entier qui spécifie en combien de millisecondes le volume augmente de nouveau jusqu'à sa valeur d'origine. Si aucune valeur n'est fournie, la valeur par défaut est 1 000 millisecondes (1 seconde).

Exemple

L'instruction Lingo suivante amplifie le son de la piste 3 pendant une durée de 3 secondes à partir du début de l'acteur `intro2` :

```
-- Syntaxe Lingo
sound(3).play(member("intro2"))
sound(3).fadeIn(3000)

// Syntaxe JavaScript
sound(3).play(member("intro2"));
sound(3).fadeIn(3000);
```

Voir aussi

fadeOut(), fadeTo(), pan, Piste audio, volume (Windows Media)

fadeOut()

Utilisation

```
-- Syntaxe Lingo
réfObjPisteAudio.fadeOut({entMillisecondes})

// Syntaxe JavaScript
réfObjPisteAudio.fadeOut({entMillisecondes});
```

Description

Méthode de piste audio ; diminue progressivement le `volume` d'une piste audio jusqu'à zéro dans un délai donné, défini en millisecondes.

Le paramètre de balance courant est conservé pour l'ensemble du fondu.

Paramètres

entMillisecondes Facultatif. Nombre entier qui spécifie en combien de millisecondes le volume est réduit à zéro. Si aucune valeur n'est fournie, la valeur par défaut est 1 000 millisecondes (1 seconde).

Exemple

L'instruction suivante diminue le son de la piste 3 sur de période de 5 secondes :

```
-- Syntaxe Lingo
sound(3).fadeOut(5000)

// Syntaxe JavaScript
sound(3).fadeOut(5000);
```

Voir aussi

fadeIn(), fadeTo(), pan, Piste audio, volume (Windows Media)

fadeTo()

Utilisation

```
-- Syntaxe Lingo
réfObjPisteAudio.fadeTo(entVolume {, entMillisecondes})

// Syntaxe JavaScript
réfObjPisteAudio.fadeTo(entVolume {, entMillisecondes});
```

Description

Méthode de piste audio ; modifie progressivement le `volume` d'une piste audio jusqu'à un volume spécifié dans un délai donné, défini en millisecondes.

Le paramètre de balance courant est conservé pour l'ensemble du fondu.

Pour un exemple d'utilisation de `fadeTo()` dans une animation, reportez-vous à l'animation Sound Control du dossier Learning/Lingo, lui-même inclus dans le dossier de Director.

Paramètres

`entVolume` Requis. Nombre entier qui spécifie le nouveau niveau de volume souhaité. La plage de valeurs du paramètre *`entVolume`* est comprise entre 0 et 255.

`entMillisecondes` Facultatif. Nombre entier qui spécifie en combien de millisecondes le niveau du volume passe à *`entVolume`*. Si aucune valeur n'est fournie, la valeur par défaut est 1 000 millisecondes (1 seconde).

Exemple

L'instruction suivante fait passer le volume de la piste audio 4 à 150, sur une période de 2 secondes. Il peut s'agir d'une augmentation ou d'une diminution de volume, en fonction du volume initial de la piste audio 4 au moment de départ du fondu.

```
-- Syntaxe Lingo
sound(4).fadeTo(150, 2000)

// Syntaxe JavaScript
sound(4).fadeTo(150, 2000);
```

Voir aussi

`fadeIn()`, `fadeOut()`, `pan`, `Piste audio`, `volume (Windows Media)`

fileName()

Utilisation

```
-- Syntaxe Lingo
réfObjFileio.fileName()

// Syntaxe JavaScript
 réfObjFileio.fileName();
```

Description

Méthode de Fileio ; renvoie le chemin complet et le nom d'un fichier ouvert.

Vous devez d'abord ouvrir un fichier en appelant `openFile()` avant d'utiliser `fileName()` pour renvoyer le nom du fichier.

Paramètres

Aucun.

Voir aussi

Fileio , openFile()

fill()

Utilisation

```
-- Syntaxe Lingo
réfObjImage.fill(gauche, haut, droite, bas, objetCouleurOuListedeParamètres)
réfObjImage.fill(point(x, y), point(x, y), objetCouleurOuListedeParamètres)
réfObjImage.fill(rect, objetCouleurOuListedeParamètres)

// Syntaxe JavaScript
réfObjImage.fill(gauche, haut, droite, bas, objetCouleurOuListedeParamètres);
réfObjImage.fill(point(x, y), point(x, y), objetCouleurOuListedeParamètres);
réfObjImage.fill(rect, objetCouleurOuListedeParamètres);
```

Description

Méthode d'image. Remplit une zone rectangulaire avec une couleur spécifiée dans un objet image donné.

Cette méthode renvoie la valeur 1 si aucune erreur n'est détectée et zéro en cas d'erreur.

Pour optimiser les performances avec des images 8 bits (ou d'une qualité inférieure), l'objet Couleur doit contenir une valeur de couleur indexée. Pour les images 16 ou 32 bits, utilisez une valeur de couleur RVB.

Paramètres

gauche Requis si vous remplissez une zone spécifiée par des coordonnées. Nombre entier qui spécifie le côté gauche de la zone à remplir.

haut Requis si vous remplissez une zone spécifiée par des coordonnées. Nombre entier qui spécifie le côté supérieur de la zone à remplir.

droite Requis si vous remplissez une zone spécifiée par des coordonnées. Nombre entier qui spécifie le côté droit de la zone à remplir.

bas Requis si vous remplissez une zone spécifiée par des coordonnées. Nombre entier qui spécifie le côté inférieur de la zone à remplir.

objCouleurOuListeDeParamètres Requis. Objet Couleur ou liste de paramètres qui spécifie la couleur de remplissage de la zone. Il est possible d'utiliser la liste de paramètres à la place d'un simple objet Couleur pour spécifier les propriétés suivantes.

Propriété	Description
#shapeType	Une valeur de symbole de #oval, #rect, #roundRect ou #line. La valeur par défaut est #line.
#lineSize	Epaisseur du trait à utiliser pour le dessin de la forme.
#color	Objet Couleur qui détermine la couleur de remplissage de la zone.
#bgColor	Objet Couleur qui détermine la couleur de la bordure de la zone.

`point(x, y), point(x, y)` Requis si vous remplissez une zone avec des points. Deux points qui spécifient les angles supérieur gauche et inférieur droit de la zone à remplir, par rapport à l'angle supérieur gauche de l'objet image donné.

rect Requis si vous remplissez une zone avec un rectangle. Rectangle qui spécifie la zone rectangulaire à remplir.

Exemple

L'instruction suivante rend l'objet image dans la variable *monImage* complètement noir :

L'instruction suivante dessine un ovale rempli dans l'objet image imageTest. L'ovale a un remplissage vert et une bordure rouge d'une épaisseur de 5 pixels.

Voir aussi

color(), draw(), image()

findLabel()

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.findLabel(quelNomDeLibellé)

// Syntaxe JavaScript
réfObjImageObjet.findLabel(quelNomDeLibellé);
```

Description

Fonction : cette fonction renvoie le numéro d'image (dans l'animation Flash) associée au nom demandé.

Un 0 est renvoyé si le nom n'existe pas ou si cette portion de l'animation Flash n'a pas encore été transférée en mémoire.

Paramètres

quelNomDeLibellé Requis. Spécifie le libellé de l'image à rechercher.

findEmpty()

Utilisation

```
-- Syntaxe Lingo
réfObjDistribution.findEmpty({réfObjActeur})

// Syntaxe JavaScript
réfObjDistribution.findEmpty({réfObjActeur});
```

Description

Méthode de distribution ; affiche la prochaine position d'acteur vide ou la position après un acteur spécifié.

Cette méthode est disponible uniquement sur la distribution courante.

Paramètres

réfObjActeur Facultatif. Référence à l'acteur après lequel la prochaine position d'acteur vide est affichée. Si ce paramètre est omis, la prochaine position d'acteur vide est affichée.

Exemple

L'instruction suivante recherche le premier acteur vide à partir de l'acteur 100 :

```
-- Syntaxe Lingo
trace(castLib(1).findEmpty(member(100)))

// Syntaxe JavaScript
trace(castLib(1).findEmpty(member(100)));
```

Voir aussi

Bibliothèque de distribution, Acteur

findPos

Utilisation

```
liste.findPos(propriété)
findPos(liste, propriété)
```

Description

Commande de liste ; identifie la position d'une propriété dans une liste de propriétés.

L'utilisation de `findPos` avec des listes linéaires renvoie un nombre fictif si la valeur de *propriété* est un nombre et une erreur de script si la valeur de *propriété* est une chaîne.

La commande `findPos` a la même fonction que la commande `findPosNear`, excepté que `findPos` a la valeur `VOID` lorsque la propriété spécifiée ne figure pas dans la liste.

Paramètres

propriété Requis. Propriété dont la position est identifiée.

Exemple

L'instruction suivante identifie la position de la propriété c dans la liste Réponses, composée de [#a:10, #b:12, #c:15, #d:22] :

```
Réponses.findPos(#c)
```

Le résultat est 3, c étant la troisième propriété de la liste.

Voir aussi

findPosNear, sort

findPosNear

Utilisation

```
listeTriée.findPosNear(valeurOuPropriété)
findPosNear(listeTriée, valeurOuPropriété)
```

Description

Commande de liste ; pour les listes triées uniquement, identifie la position d'un élément dans une liste triée spécifiée.

La commande `findPosNear` fonctionne uniquement avec les listes triées. Remplacez *valeurOuPropriété* par une valeur pour les listes linéaires triées et par une propriété pour les listes de propriétés triées.

La commande `findPosNear` est semblable à la commande `findPos` excepté que, lorsque la propriété spécifiée n'est pas dans la liste, la commande `findPosNear` identifie la position de la valeur ayant le nom alphanumérique le plus similaire. Cette commande est pratique pour trouver le nom le plus proche dans un répertoire de noms triés.

Paramètres

`valeurOuPropriété` Requis. Valeur ou propriété dont la position est identifiée.

Exemple

L'instruction suivante identifie la position d'une propriété dans la liste triée Réponses, composée de [#Nil:2, #Pharaon:4, #Raja:0] :

```
Réponses.findPosNear(#Ni)
```

Le résultat est 1, Ni étant le plus proche de Nil, la première propriété de la liste.

Voir aussi

`findPos`

finishIdleLoad()

Utilisation

```
-- Syntaxe Lingo
_movie.finishIdleLoad(entBaliseDeChargement)

// Syntaxe JavaScript
_movie.finishIdleLoad(entBaliseDeChargement);
```

Description

Méthode d'animation ; force le chargement de tous les acteurs possédant la balise de chargement spécifiée.

Paramètres

`entBaliseDeChargement` Requis. Nombre entier qui spécifie la balise de chargement des acteurs à charger.

Exemple

L'instruction suivante termine le chargement de tous les acteurs possédant la balise de chargement 20 :

```
-- Syntaxe Lingo
_movie.finishIdleLoad(20)

// Syntaxe JavaScript
_movie.finishIdleLoad(20);
```

Voir aussi

`idleHandlerPeriod`, `idleLoadDone()`, `idleLoadMode`, `idleLoadPeriod`, `idleLoadTag`, `idleReadChunkSize`, `Animation`

flashToStage()

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.flashToStage(pointDeLanimationFlash)

// Syntaxe JavaScript
réfObjImageObjet.flashToStage(pointDeLanimationFlash);
```

Description

Fonction ; renvoie la coordonnée de la scène Director correspondant à une coordonnée spécifiée dans une image-objet d'animation Flash. Cette fonction accepte la coordonnée de piste et d'animation Flash et renvoie la coordonnée de scène Director en valeurs de points Director : par exemple, point (300,300).

Les coordonnées d'animation Flash sont mesurées en pixels d'animation Flash, qui sont déterminés par la taille d'origine d'une animation lors de sa création dans Flash. Afin de calculer les coordonnées de l'animation Flash, le point (0,0) d'une animation Flash est toujours son angle supérieur gauche. La propriété `originPoint` de l'acteur est utilisée uniquement pour la rotation et la mise à l'échelle, mais pas pour le calcul des coordonnées d'une animation.

La fonction `flashToStage` et la fonction `stageToFlash` correspondante sont pratiques pour déterminer la coordonnée d'une animation Flash se trouvant à une coordonnée spécifique de la scène Director. Pour Flash et Director, le point (0,0) est le coin supérieur gauche de la scène Flash ou Director. Ces coordonnées peuvent ne pas coïncider sur la scène Director si une image-objet Flash est étirée, mise à l'échelle ou a pivoté.

Paramètres

`pointDeLanimationFlash` Requis. Point dans l'image-objet de l'animation Flash dont les coordonnées sont renvoyées.

Exemple

Le gestionnaire suivant accepte une valeur de point et une référence d'image-objet comme paramètre, puis définit la coordonnée supérieure gauche de l'image-objet spécifiée en fonction du point indiqué dans une image-objet d'animation Flash de la piste 10 :

```
-- Syntaxe Lingo
on déplacImObj(quelPointFlash, quelleImageObjet)
  sprite(quelleImageObjet).loc = sprite(1).FlashToStage(quelPointFlash)
  _movie.updatestage()
end

// Syntaxe JavaScript
function déplacImObj(quelPointFlash, quelleImageObjet) {
  sprite(quelleImageObjet).loc = sprite(1).FlashToStage(quelPointFlash);
  _movie.updateStage();
}
```

Voir aussi

`stageToFlash()`

float()

Utilisation

```
(expression).float
float (expression)
```

Description

Fonction (Lingo uniquement) ; convertit une expression en un nombre à virgule flottante. Le nombre de chiffres après la virgule (pour l'affichage uniquement, les calculs n'étant pas affectés) est défini à l'aide de la propriété `floatPrecision`.

Dans la syntaxe JavaScript, utilisez la fonction `parseFloat()`.

Paramètres

`expression` Requis. Expression à convertir en nombre à virgule flottante.

Exemple

L'instruction suivante convertit le nombre entier 1 en 1 à virgule flottante :

```
put (1).float
-- 1.0
```

Les opérations mathématiques peuvent être réalisées avec `float`. Si l'un des termes a une valeur `float`, toute l'opération est réalisée avec `float` :

```
"the floatPrecision = 1
put 2 +2
-- 4
put (2).float + 2
-- 4.0
the floatPrecision = 4
put 22 / 7
-- 3
put (22).float / 7
-- 3.1429"
```

Voir aussi

`floatPrecision`, `ilk()`

floatP()

Utilisation

```
(expression).floatP
floatP(expression)
```

Description

Fonction (Lingo uniquement) ; indique si une expression est un nombre à virgule flottante (1 ou `TRUE`) ou non (0 ou `FALSE`).

Le *P* dans `floatP` signifie *prédicat*.

Paramètres

`expression` Requis. Expression à tester.

Exemple

L'instruction suivante vérifie si 3.0 est un nombre à virgule flottante. La fenêtre Messages affiche le nombre 1, indiquant que c'est le cas (TRUE).

```
put (3.0).floatP
-- 1
```

L'instruction suivante vérifie si 3 est un nombre à virgule flottante. La fenêtre Messages affiche le nombre 0, indiquant que ce n'est pas le cas (FALSE).

```
put (3).floatP
-- 0
```

Voir aussi

float(), ilk(), integerP(), objectP(), stringP(), symbolP()

flushInputEvents()

Utilisation

```
-- Syntaxe Lingo
_player.flushInputEvents()

// Syntaxe JavaScript
_player.flushInputEvents();
```

Description

Méthode de lecteur ; purge les événements clavier ou souris de la file d'attente des messages de Director.

En règle générale, cette commande est pratique lorsque le script exécute une boucle assez longue et que vous souhaitez vous assurer que les commandes effectuées au clavier ou à l'aide de la souris ne seront pas transmises.

Cette méthode n'est effective qu'en cours d'exécution et n'a aucun effet pendant la programmation.

Paramètres

Aucun.

Exemple

L'instruction suivante désactive les événements souris et clavier lors de l'exécution d'une boucle de répétition :

```
-- Syntaxe Lingo
repeat with i = 1 to 10000
  _player.flushInputEvents()
  sprite(1).loc = sprite(1).loc + point(1, 1)
end repeat
```

```
// Syntaxe JavaScript
for (var i = 1; i <= 10000; i++) {
  _player.flushInputEvents();
  sprite(1).loc = sprite(1).loc + point(1, 1);
}
```

Voir aussi

on keyDown, on keyUp, on mouseDown (gestionnaire d'événement), on mouseUp (gestionnaire d'événement), Lecteur

forget() (fenêtre)

Utilisation

```
-- Syntaxe Lingo
réfObjFenêtre.forget()

// Syntaxe JavaScript
réfObjFenêtre.forget();
```

Description

Méthode de fenêtre ; ordonne au script de fermer la fenêtre et d'arrêter sa lecture lorsqu'elle n'est plus utilisée et qu'aucune autre variable n'y fait référence.

L'appel de la méthode `forget()` sur une fenêtre supprime également la référence à cette fenêtre de la liste `windowList`.

Lorsque la méthode `forget window` est appelée, la fenêtre et l'animation dans une fenêtre (MIAW) disparaissent sans appeler les gestionnaires `stopMovie`, `closeWindow` ou `deactivateWindow`.

S'il existe plusieurs références globales à l'animation dans une fenêtre, la fenêtre ne répond pas à la méthode `forget()`.

Paramètres

Aucun.

Exemple

L'instruction suivante ordonne à Lingo de supprimer la fenêtre Tableau de commande lorsque l'animation ne l'utilise plus :

```
-- Syntaxe Lingo
window("Tableau de commande").forget()

// Syntaxe JavaScript
window("Tableau de commande").forget();
```

Voir aussi

close(), open() (fenêtre), Fenêtre, windowList

forget() (temporisation)

Utilisation

```
timeout("nomDeTemporisation").forget()
forget(timeout("nomDeTemporisation"))
```

Description

Cette fonction d'objet de temporisation supprime un objet de temporisation de la liste `timeoutList` et l'empêche d'envoyer d'autres événements de temporisation.

Paramètres

Aucun.

Exemple

L'instruction suivante supprime l'objet de temporisation Réveil de la liste `timeoutList` :

```
timeout("Réveil").forget()
```

Voir aussi

`timeout()`, `timeoutHandler`, `timeoutList`, `new()`

framesToHMS()

Utilisation

```
framesToHMS(images, cadence, compensé, fractions)
```

Description

Fonction ; convertit le nombre d'images spécifié en durée équivalente en heures, minutes et secondes. Cette fonction est pratique pour prévoir la durée de lecture réelle d'une animation ou pour contrôler un appareil de lecture vidéo.

Le résultat est une chaîne sous la forme `sHH:MM:SS.FFD`, où :

s	Un caractère est utilisé si le temps est inférieur à zéro ou un espace si le temps est supérieur ou égal à zéro.
HH	Heures.
MM	Minutes.
SS	Secondes.
FF	Indique une fraction de seconde si *`fractions`* a la valeur `TRUE` ou des `images` si *`fractions`* a la valeur `FALSE`.
D	Un `"d"` est utilisé si *`compensé`* a la valeur `TRUE` ou un espace si *`compensé`* a la valeur `FALSE`.

Paramètres

`images` Requis. Expression entière qui spécifie le nombre d'images.

`cadence` Requis. Expression entière qui spécifie la cadence en nombre d'images par seconde.

`compensé` Requis. Compense la cadence d'image NTSC couleur, qui n'est pas exactement de 30 images par seconde, et n'est utile que si la cadence est définie sur 30 images par seconde. Normalement, ce paramètre a pour valeur `FALSE`.

fractions Requis. Détermine si les images résiduelles sont converties au centième de seconde le plus proche (TRUE) ou renvoyées sous la forme d'un nombre entier d'images (FALSE).

Exemple

L'instruction suivante convertit une animation de 2710 images, 30 images par seconde. Les arguments *compensé* et *fractions* sont tous les deux désactivés :

```
put framesToHMS(2710, 30, FALSE, FALSE)
-- " 00:01:30.10 "
```

Voir aussi

HMStoFrames()

frameReady() (animation)

Utilisation

```
-- Syntaxe Lingo
_movie.frameReady({entNumDimage})
_movie.frameReady(numDimageA, numDimageB)

// Syntaxe JavaScript
_movie.frameReady({entNumDimage});
_movie.frameReady(numDimageA, numDimageB);
```

Description

Méthode d'animation ; pour les animations Director, les projections et les animations avec un contenu Shockwave, détermine si les acteurs d'une image ou d'une plage d'images ont été téléchargés.

Cette méthode renvoie TRUE si les acteurs spécifiés ont été téléchargés et FALSE dans le cas contraire.

Pour un exemple d'utilisation de la méthode frameReady() dans une animation Director, consultez l'animation « Shockwave et flux continu » dans l'Aide de Director.

Paramètres

entNumDimage Facultatif si le test porte sur le téléchargement des acteurs d'une image unique. Nombre entier qui spécifie l'image à tester. Si ce paramètres est omis, frameReady() détermine si les acteurs utilisés dans n'importe quelle image du scénario ont été téléchargés.

numDimageA Requis si le test porte sur le téléchargement des acteurs d'une plage d'images. Nombre entier qui spécifie la première image de la plage.

numDimageB Requis si le test porte sur le téléchargement des acteurs d'une plage d'images. Nombre entier qui spécifie la dernière image de la plage.

Exemple

L'instruction suivante détermine si les acteurs de l'image 20 sont téléchargés et prêts à être affichés :

```
-- Syntaxe Lingo
on exitFrame
  if (_movie.frameReady(20)) then
    _movie.go(20)
  else
    _movie.go(1)
  end if
end

// Syntaxe JavaScript
function exitFrame() {
  if (_movie.frameReady(20)) {
    _movie.go(20);
  }
  else {
    _movie.go(1);
  }
}
```

Le script d'image suivant vérifie si l'image 25 d'une image-objet animation Flash dans la piste 5 peut être affichée. Dans le cas contraire, le script maintient la tête de lecture en boucle sur l'image courante de l'animation Director. Lorsque l'image 25 peut être affichée, le script démarre l'animation et laisse la tête de lecture passer à l'image suivante de l'animation Director.

Voir aussi

`mediaReady`, `Animation`

frameStep()

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.frameStep(entImages)

// Syntaxe JavaScript
réfObjDvd.frameStep(entImages);
```

Description

Méthode de DVD ; avance d'un nombre d'images spécifié à partir de la position courante.

Paramètres

`entImages` Requis. Nombre entier qui indique de combien d'images vous voulez avancer.

Exemple

Cette instruction fait avancer de 100 images :

```
-- Syntaxe Lingo
member("drame").frameStep(100)

// Syntaxe JavaScript
member("drame").frameStep(100);
```

Voir aussi

DVD

freeBlock()

Utilisation

```
the freeBlock
```

Description

Fonction ; indique la taille du plus grand bloc de mémoire disponible, en octets. Un kilo-octet (ko) correspond à 1 024 octets. Un méga-octet (Mo) correspond à 1 024 kilo-octets. Le chargement d'un acteur nécessite un bloc libre au moins aussi grand que l'acteur.

Paramètres

Aucun.

Exemple

L'instruction suivante détermine si le plus grand bloc de mémoire disponible est inférieur à 10 ko et affiche un message d'alerte si c'est le cas :

```
-- Syntaxe Lingo
if (the freeBlock < (10 * 1024)) then alert "Mémoire insuffisante !"

// Syntaxe JavaScript
if (freeBlock < (10 * 1024)) {
  alert("Mémoire insuffisante !")
}
```

Voir aussi

freeBytes(), memorySize, ramNeeded(), size

freeBytes()

Utilisation

```
the freeBytes
```

Description

Fonction ; indique le nombre total d'octets de mémoire disponible, qui ne forme pas forcément un bloc continu. Un kilo-octet (ko) correspond à 1 024 octets. Un méga-octet (Mo) correspond à 1 024 kilo-octets.

Cette fonction est différente de `freeBlock`, puisqu'elle indique toute la mémoire disponible et pas seulement la mémoire contiguë.

Sur Macintosh, la sélection de l'option Utiliser la mémoire temporaire du système dans les préférences générales de Director ou dans la boîte de dialogue Options d'une projection indique à la fonction `freeBytes` de renvoyer toute la mémoire disponible pour l'application. Cette quantité est égale à la quantité affectée à l'application, affichée dans sa boîte de dialogue Lire les informations, et à la valeur Mémoire disponible affichée dans la boîte de dialogue A propos de votre Macintosh.

Paramètres

Aucun.

Exemple

L'instruction suivante vérifie si plus de 200 ko de mémoire son disponibles et lit une animation couleur si c'est le cas :

```
if (the freeBytes > (200 * 1024)) then play movie "animationCouleur"
```

Voir aussi

`freeBlock()`, `memorySize`, `objectP()`, `ramNeeded()`, `size`

generateNormals()

Utilisation

```
member(quelActeur).modelResource(quelleRessDeMod).
  generateNormals(style)
```

Description

Commande 3D de ressource de modèle `#mesh` ; calcule les vecteurs `normal` pour chaque sommet de la maille.

Si le paramètre *`style`* a pour valeur `#flat`, chaque sommet reçoit une normale pour chaque face à laquelle il appartient. Les trois sommets d'une face partagent la même normale. Par exemple, si les sommets de `face[1]` reçoivent tous `normal[1]` et les sommets de `face[2]` reçoivent tous `normal[2]`, et que les deux faces partagent `vertex[8]`, la normale de `vertex[8]` est `normal[1]` pour `face[1]` et `normal[2]` pour `face[2]`. L'utilisation du paramètre `#flat` résulte en une délimitation très claire des faces d'une maille.

Si le paramètre *`style`* a pour valeur `#smooth`, chaque sommet ne reçoit qu'une seule normale, quel que soit le nombre de faces auquel il appartient, les trois sommets d'une face pouvant avoir différente normales. Chaque normale de sommet est la moyenne des normales de toutes les faces le partageant. L'utilisation du paramètre `#smooth` résulte en une apparence plus adoucie des faces d'une maille, à l'exception des bords extérieurs, qui restent nets.

Une normale de sommet est le vecteur de direction indiquant la direction « vers l'avant » d'un sommet. Si la normale de sommet pointe vers la caméra, les couleurs affichées dans la région de la maille contrôlée par cette normale sont déterminées par le matériau. Si la normale de sommet pointe en direction opposée à la caméra, la région de la maille contrôlée par cette normale ne sera pas visible.

Si vous utilisez la commande `generateNormals()`, vous devrez utiliser la commande `build()` pour reconstruire la maille.

Paramètres

`style` Requis. Symbole qui spécifie le style du sommet.

Exemple

L'instruction suivante calcule les normales de sommet de la ressource de modèle mailleDeSol. Le paramètre *style* a pour valeur #smooth, tel que chaque sommet de la maille ne reçoive qu'une seule normale.

```
member("Pièce").modelResource("mailleDeSol").generateNormals(#smooth)
```

Voir aussi

build(), face[ ], normalList, normals, flat

getaProp

Utilisation

```
listeDePropriétés.nomDePropriété
getaProp(liste, élément)
liste[positionDansLaListe]
listeDePropriétés [ #nomDePropriété ]
listeDePropriétés [ "nomDePropriété" ]
```

Description

Commande de liste ; pour les listes linéaires et de propriétés, identifie la valeur associée à l'élément spécifié par *élément*, *positionDansLaListe* ou *nomDePropriété* dans la liste spécifiée par *liste*.

- Lorsque la liste est linéaire, remplacez *élément* par le numéro correspondant à la position de l'élément dans cette liste, indiquée par *positionDansLaListe*. Le résultat est la valeur située à cette position.
- Lorsque la liste est une liste de propriétés, remplacez *élément* par une propriété de la liste comme dans *nomDePropriété*. Le résultat est la valeur associée à cette propriété.

La commande getaProp renvoie VOID si la valeur spécifiée n'est pas dans la liste.

Lorsque utilisée avec des listes linéaires, la commande getaProp a la même fonction que la commande getAt.

Paramètres

nomOuNumDélément Requis. Pour les listes linéaires, nombre entier qui spécifie la position d'index de la valeur dans la liste à renvoyer ; pour les listes de propriétés, symbole (Lingo) ou chaîne (syntaxe JavaScript) qui spécifie la propriété dont la valeur est renvoyée.

Exemple

L'instruction suivante identifie la valeur associée à la propriété #pierre dans la liste de propriétés âges, composée de [#jean:10, #pierre:12, #sophie:15, #barbara:22] :

```
put getaProp(âges, #pierre)
```

Le résultat est 12, car il s'agit de la valeur associée à la propriété #pierre.

Le même résultat peut être obtenu avec des crochets d'accès dans la même liste :

```
put âges[#jean]
```

Le résultat est à nouveau 12.

Pour obtenir la valeur située à une certaine position dans la liste, vous pouvez également utiliser des crochets d'accès. Pour obtenir la troisième valeur de la liste associée à la troisième propriété, utilisez la syntaxe suivante :

```
put âges[3]
-- 15
```

Remarque : Contrairement à la commande `getAProp` dans laquelle la valeur `VOID` est renvoyée lorsqu'une propriété n'existe pas, une erreur de script a lieu si la propriété n'existe pas et que des crochets d'accès sont utilisés.

Voir aussi

`getAt`, `getOne()`, `getProp()`, `setaProp`, `setAt`

getAt

Utilisation

```
getAt(liste, position)
liste [position]
```

Description

Commande de liste ; identifie l'élément situé à une position donnée dans une liste spécifiée. Si la liste contient moins d'éléments que la position spécifiée, une erreur de script a lieu.

La commande `getAt` fonctionne avec les listes linéaires et de propriétés. Elle a la même fonction que la commande `getaProp` pour les listes linéaires.

Elle est pratique pour extraire une liste d'une autre liste, comme `deskTopRectList`.

Paramètres

`liste` Requis. Spécifie la liste dans laquelle l'élément existe.

`position` Requis. Spécifie la position d'index de l'élément dans la liste.

Exemple

L'instruction suivante entraîne l'affichage dans la fenêtre Messages du troisième élément de la liste Réponses, composée de [10, 12, 15, 22] :

```
put getAt(Réponses, 3)
-- 15
```

Le même résultat peut être renvoyé à l'aide de crochets d'accès :

```
put Réponses[3]
-- 15
```

L'exemple suivant extrait la première entrée d'une liste de deux entrées indiquant les noms, services et numéros d'identification des employés. Le second élément de la liste extraite est ensuite renvoyé, identifiant le service dans lequel la première personne de la liste est employée. Le format de la liste est [["Denis", "Conseil", 510], ["Sophie", "Distribution", 973]] et la liste est appelée listeDinfosDesEmployés.

```
première̲Personne = getAt(listeDinfosDesEmployés, 1)
put première̲Personne
-- ["Denis", "Conseil", 510]
serviceDeLaPremièrePersonne = getAt(première̲Personne, 2)
put serviceDeLaPremièrePersonne
-- "Conseil"
```

Il est également possible d'imbriquer des commandes `getAt` sans affecter de valeurs aux variables dans les étapes intermédiaires. Ce format peut être plus difficile à lire et rédiger, mais contient moins de texte.

```
serviceDeLaPremièrePersonne = getAt(getAt(listeDinfosDesEmployés, 1), 2)
put serviceDeLaPremièrePersonne
 -- "Conseil"
```

Vous pouvez également utiliser les crochets d'accès :

```
première̲Personne = listeDinfosDesEmployés[1]
put première̲Personne
-- ["Denis", "Conseil", 510]
serviceDeLaPremièrePersonne = première̲Personne[2]
put serviceDeLaPremièrePersonne
-- "Conseil"
```

De même qu'avec `getAt`, les crochets peuvent être imbriqués :

```
serviceDeLaPremièrePersonne = listeDinfosDesEmployés[1][2]
```

Voir aussi

`getaProp`, `setaProp`, `setAt`

getError() (Flash, SWA)

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.getError()

// Syntaxe JavaScript
réfObjActeur.getError();
```

Description

Fonction ; pour les acteurs Shockwave Audio (SWA) ou Flash, indique si une erreur est survenue pendant le transfert de l'acteur en mémoire et renvoie une valeur.

Les acteurs Shockwave Audio ont les valeurs entières `getError()` suivantes possibles et les messages `getErrorString()` correspondants :

valeur getError()	message getErrorString()
0	OK
1	memory

valeur getError()	message getErrorString()
2	network
3	playback device
99	other

Les valeurs `getError` possibles des acteurs animation Flash sont les suivantes :

- `FALSE` – Aucune erreur n'est survenue.
- `#memory` – La mémoire est insuffisante pour le chargement de l'acteur.
- `#fileNotFound` – Le fichier contenant les éléments de l'acteur est introuvable.
- `#network` – Une erreur réseau a empêché le chargement de l'acteur.
- `#fileFormat` – Le fichier a été trouvé, mais semble être d'un type incorrect ou une erreur est survenue à la lecture.
- `#other` – Une autre erreur est survenue.

Lorsqu'une erreur survient pendant le transfert de l'acteur en mémoire, Director affecte la valeur –1 à la propriété d'état de l'acteur. Utilisez la fonction `getError` pour déterminer le type d'erreur.

Paramètres

Aucun.

Exemple

Le gestionnaire suivant utilise `getError` pour déterminer si une erreur impliquant l'acteur Shockwave Audio Norma Desmond parle est survenue et, le cas échéant, affiche la chaîne d'erreur appropriée dans un champ :

```
-- Syntaxe Lingo
on exitFrame
  if member("Norma Desmond parle").getError() <> 0 then
    member("Affichage de l'erreur").text = member("Norma Desmond \
  parle").getErrorString()
  end if
end

// Syntaxe JavaScript
function exitFrame() {
  var memNor = member("Norma Desmond parle").getError();
  if (memNor != 0) {
    member("Affichage de l'erreur").text =
  member("Norma Desmond parle").getErrorString();
  }
}
```

Le gestionnaire suivant vérifie si une erreur est survenue pour un acteur Flash intitulé Dali, lors de son transfert en mémoire. Si une erreur est survenue et qu'il s'agit d'une erreur de mémoire, le script utilise la commande `unloadCast` pour essayer de libérer de la mémoire et fait ensuite passer la tête de lecture à une image appelée Artistes de l'animation Director, dans laquelle l'image-objet d'animation Flash apparaît en premier, de façon à ce que Director puisse encore essayer de charger et lire l'animation Flash. Si un autre type d'erreur est survenu, le script passe à une image appelée Désolé, qui explique que l'animation Flash requise ne peut pas être lue.

```
-- Syntaxe Lingo
on CheckFlashStatus
  testErreur = member("Dali").getError()
  if testErreur <> 0 then
    if testErreur = #memory then
      member("Dali").clearError()
      unloadCast()
      _movie.go("Artistes")
    else
      _movie.go("Désolé")
    end if
  end if
end

// Syntaxe JavaScript
function CheckFlashStatus() {
  var testErreur = member("Dali").getError();
  if (testErreur != 0) {
    if (testErreur = "memory") {
      member("Dali").clearError();
      unloadCast();
      _movie.go("Artistes");
    } else {
      _movie.go("Désolé");
    }
  }
}
```

Voir aussi

clearError(), getErrorString(), state (Flash, SWA)

getError() (XML)

Utilisation

gObjetDanalyse.getError()

Description

Fonction ; renvoie une chaîne d'erreur descriptive associée à un numéro d'erreur (comprenant le numéro de la ligne et de la colonne de l'emplacement de l'erreur dans le code XML). Cette fonction renvoie <VOID> lorsqu'il n'existe aucune erreur.

Paramètres

Aucun.

Exemple

Les instructions suivantes vérifient une erreur après l'analyse d'une chaîne contenant des données XML :

```
errCode = ObjetDanalyse.parseString(member("texteXML").text)
chaîneDerreur = ObjetDanalyse.getError()
if voidP(chaîneDerreur) then
  -- continuer et utiliser le code XML
else
  alert "Désolé. Une erreur est survenue " & errorString
  -- sortie du gestionnaire
  exit
end if
```

getErrorString()

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.getErrorString()

// Syntaxe JavaScript
réfObjActeur.getErrorString();
```

Description

Fonction ; pour les acteurs Shockwave Audio (SWA), renvoie la chaîne de message d'erreur correspondant à la valeur de l'erreur renvoyée par la fonction `getError()`.

Les valeurs entières `getError()` possibles et les messages `getErrorString()` correspondants sont :

valeur getError()	message getErrorString()
0	OK
1	memory
2	network
3	playback device
99	other

Paramètres

Aucun.

Exemple

Le gestionnaire suivant utilise `getError()` pour déterminer si une erreur est survenue pour l'acteur Shockwave Audio Norma Desmond parle et, si c'est le cas, utilise `getErrorString` pour obtenir le message d'erreur et l'affecter à un acteur champ :

```
-- Syntaxe Lingo
on exitFrame
  if member("Norma Desmond parle").getError() <> 0 then
    member("Affichage de l'erreur").text = member("Norma Desmond \
  parle").getErrorString()
  end if
end

// Syntaxe JavaScript
function exitFrame() {
  var memNor = member("Norma Desmond parle").getError();
  if (memNor != 0) {
    member("Affichage de l'erreur").text =
  member("Norma Desmond parle").getErrorString();
  }
}
```

Voir aussi

`getError() (Flash, SWA)`

getFinderInfo()

Utilisation

```
-- Syntaxe Lingo
réfObjFileio.getFinderInfo()

// Syntaxe JavaScript
réfObjFileio.getFinderInfo();
```

Description

Méthode de Fileio (Macintosh uniquement) ; renvoie les informations du Finder pour un fichier ouvert.

Vous devez d'abord ouvrir un fichier en appelant `openFile()` avant d'utiliser `getFinderInfo()` pour renvoyer les informations du Finder relatives au fichier.

Paramètres

Aucun.

Voir aussi

`Fileio`, `openFile()`

getFlashProperty()

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.getFlashProperty(NomDeCible, symProp)

// Syntaxe JavaScript
réfObjImageObjet.getFlashProperty(NomDeCible, symProp);
```

Description

Cette fonction permet à Lingo d'invoquer la fonction script d'action Flash `getProperty()` dans l'image-objet Flash donnée. Cette fonction script d'action Flash est utilisée pour obtenir la valeur des propriétés des recadrages ou des niveaux dans une animation Flash. Cette fonction est similaire au test des propriétés d'images-objets dans Director.

Pour obtenir une propriété globale de l'image-objet Flash, passez une ligne vide comme *`nomDeCible`*. Ces propriétés globales Flash peuvent être testées : `#focusRect` et `#spriteSoundBufferTime`.

Consultez la documentation de Flash pour plus d'informations sur ces propriétés.

Paramètres

`nomDeCible` Requis. Chaîne qui spécifie le nom du clip ou du niveau de l'animation dont vous souhaitez obtenir la propriété dans l'image-objet Flash donnée.

`symProp` Requis. Symbole qui spécifie le nom de la propriété à obtenir. Les valeurs correctes sont : `#posX`, `#posY`, `#scaleX`, `#scaleY`, `#visible`, `#rotate`, `#alpha`, `#name`, `#width`, `#height`, `#target`, `#url`, `#dropTarget`, `#totalFrames`, `#currentFrame`, `#cursor` et `#lastframeLoaded`.

Exemple

L'instruction suivante obtient la valeur de la propriété `#rotate` du clip Etoile dans l'acteur Flash de l'image-objet 3 :

```
-- Syntaxe Lingo
sprite(3).setFlashProperty("Etoile", #rotate)
sprite(3).getFlashProperty()

// Syntaxe JavaScript
sprite(3).setFlashProperty("Etoile", symbol("rotate"));
sprite(3).getFlashProperty();
```

getFrameLabel()

Utilisation

```
sprite(quelleImageObjetFlash).getFrameLabel(quelNuméroDimageFlash)
getFrameLabel(sprite quelleImageObjetFlash, quelNuméroDimageFlash)
```

Description

Fonction ; renvoie le libellé d'image d'une animation Flash associé au nom de numéro demandé. Si le libellé n'existe pas ou si cette portion de l'animation Flash n'a pas encore été transférée en mémoire, cette fonction renvoie une chaîne vide.

Paramètres

`quelNuméroDimageFlash` Requis. Spécifie le numéro d'image associé au libellé d'image.

Exemple

Le gestionnaire suivant vérifie si le repère de l'image 15 de l'animation Flash lue dans l'image-objet 1 porte le nom « Lions ». Le cas échéant, l'animation Director passe à l'image Lions. Dans le cas contraire, l'animation Director reste dans l'image courante et la lecture de l'animation Flash se poursuit.

```
-- Syntaxe Lingo
on exitFrame
  if sprite(1).getFrameLabel(15) = "Lions" then
    go "Lions"
  else
    go the frame
  end if
end

// Syntaxe JavaScript
function exitFrame() {
  if (sprite(1).getFrameLabel(15) == "Lions") {
    _movie.go("Lions");
  } else {
    _movie.go(_movie.frame);
  }
}
```

getHardwareInfo()

Utilisation

```
getRendererServices().getHardwareInfo()
```

Description

Méthode 3D `rendererServices` ; renvoie une liste de propriétés contenant les informations relatives à la carte vidéo de l'utilisateur. Cette liste contient les propriétés suivantes :

`#present` est une valeur booléenne indiquant si l'ordinateur est équipé de matériel d'accélération vidéo.

`#vendor` indique le nom du fabricant de la carte vidéo.

`#model` indique le modèle de la carte vidéo.

`#version` indique la version du pilote vidéo.

`#maxTextureSize` est une liste linéaire contenant la largeur et hauteur maximum d'une texture, en pixels. Les textures dépassant cette taille sont sous-échantillonnées à la taille maximum. Vous pourrez éviter des erreurs de sous-échantillonnage en créant des textures de tailles variées et en choisissant celles qui ne dépassent pas la valeur `#maxTextureSize` à l'exécution.

`#supportedTextureRenderFormats` est une liste linéaire des formats supportés par la carte vidéo. Pour plus d'informations, consultez `textureRenderFormat`.

`#textureUnits` indique le nombre d'unités de texture disponibles pour la carte.

`#depthBufferRange` est une liste linéaire de résolutions binaires associées à la propriété `depthBufferDepth`.

`#colorBufferRange` est une liste linéaire de résolutions binaires associées à la propriété `colorBufferDepth`.

Exemple

L'instruction suivante affiche une liste de propriétés détaillée concernant le matériel de l'utilisateur.

```
put getRendererServices().getHardwareInfo()
-- [#present: 1, #vendor: "NVIDIA Corporation", #model: \
  "32MB DDR NVIDIA GeForce2 GTS (Dell)", #version: "4.12.01.0532", \
  #maxTextureSize: [2048, 2048], #supportedTextureRenderFormats: \
  [#rgba8888, #rgba8880, #rgba5650, #rgba5551, #rgba5550, \
  #rgba4444], #textureUnits: 2, #depthBufferRange: [16, 24], \
  #colorBufferRange: [16, 32]]
```

Voir aussi

`getRendererServices()`

getHotSpotRect()

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.getHotSpotRect(idDeZoneRéférencée)

// Syntaxe JavaScript
réfObjImageObjet.getHotSpotRect(idDeZoneRéférencée);
```

Description

Fonction QuickTime VR ; renvoie un rectangle de délimitation approximative pour une zone référencée. Si la zone référencée n'existe pas ou n'est pas visible sur la scène, cette fonction renvoie rect(0, 0, 0, 0). Si la zone référencée est partiellement visible, cette fonction renvoie le rectangle de délimitation pour la partie visible.

Paramètres

idDeZoneRéférencée Requis. Spécifie la zone référencée pour laquelle un rectangle de délimitation est renvoyé.

getLast()

Utilisation

```
liste.getLast()
getLast(liste)
```

Description

Fonction de liste ; identifie la dernière valeur d'une liste linéaire ou de propriétés spécifiée par *liste*.

Paramètres

Aucun.

Exemple

L'instruction suivante identifie le dernier élément (22) de la liste Réponses, composée de [10, 12, 15, 22] :

```
put Réponses.getLast()
```

L'instruction suivante identifie le dernier élément (850) de la liste Devis, composée de [#avatar:750, #dupont:600, #soldes:850] :

```
put Devis.getLast()
```

getLatestNetID

Utilisation

```
getLatestNetID
```

Description

Cette fonction renvoie un identifiant pour la dernière opération réseau entamée.

L'identificateur renvoyé par `getLatestNetID` peut être utilisé en tant que paramètre dans les fonctions `netDone`, `netError` et `netAbort` pour identifier la dernière opération réseau.

Remarque : Cette fonction est destinée à la compatibilité en amont. Il est recommandé d'utiliser l'ID Réseau renvoyé par une fonction réseau de Lingo plutôt que par `getLatestNetID`. Cependant, si vous utilisez `getLatestNetID`, faites-le immédiatement après la commande netLingo.

Paramètres

Aucun.

Exemple

Le script suivant affecte l'identifiant réseau d'une opération `getNetText` à l'acteur champ Résultat, de manière à pouvoir récupérer ultérieurement les résultats de cette opération :

```
on débutDopération
  global gIDréseau
  getNetText("url")
  set gIDréseau = getLatestNetID()
end
on vérifierLopération
  global gIDréseau
  if netDone(gIDréseau) then
    put netTextResult into member "Résultat"
  end if
end
```

Voir aussi

`netAbort`, `netDone()`, `netError()`

getLength()

Utilisation

```
-- Syntaxe Lingo
réfObjFileio.getLength()

// Syntaxe JavaScript
réfObjFileio.getLength();
```

Description

Méthode de Fileio ; renvoie la longueur d'un fichier ouvert.

Vous devez d'abord ouvrir un fichier en appelant `openFile()` avant d'utiliser `getLength()` pour renvoyer la longueur du fichier.

Paramètres

Aucun.

Voir aussi

`Fileio`, `openFile()`

getNetText()

Utilisation

```
getNetText(URL {, chaîneOSduServeur} {, jeuDeCaractères})
getNetText(URL, listeDePropriétés {, chaîneOSduServeur} {, jeuDeCaractères})
```

Description

Fonction ; démarre la récupération de texte à partir d'un fichier placé sur un serveur HTTP ou FTP ou lance une requête CGI.

La première syntaxe présentée entame la récupération du texte. Vous pouvez soumettre des requêtes CGI HTTP de cette manière et devez les encoder correctement dans l'adresse URL. La seconde syntaxe comprend une liste de propriétés et soumet une requête CGI, fournissant le codage URL correct.

Utilisez le paramètre facultatif *listeDePropriétés* pour utiliser une liste de propriétés pour les requêtes CGI. La liste de propriétés est encodée dans l'URL et l'URL envoyée est `(chaîneURL & "?" & listedePropriétésCodée)`.

Utilisez le paramètre facultatif *chaîneOSduServeur* pour encoder tout caractère renvoyé dans *listeDePropriétés*. La valeur est par défaut `UNIX` mais peut recevoir `Win` ou `Mac` comme valeur et convertit les retours chariot de l'argument *listeDePropriétés* en ceux utilisés par le serveur. Pour la plupart des applications, ce paramètre n'est pas nécessaire, les ruptures de ligne n'étant généralement pas utilisées dans les réponses de formulaires.

Le paramètre facultatif *jeuDeCaractères* ne s'applique que si l'utilisateur exécute Director sur un système `shift-JIS` (Japonais). Les paramètres des jeux de caractères possibles sont `JIS`, `EUC`, `ASCII` et `AUTO`. Lingo convertit les données récupérées de `shift-JIS` dans le jeu de caractères spécifié. Avec le paramètre AUTO, le jeu de caractères essaie de déterminer le jeu de caractères du texte récupéré et de le convertir au jeu de caractères de la machine locale. Le paramètre par défaut est ASCII.

Utilisez `netDone` pour savoir quand l'opération `getNetText` est terminée et `netError` pour savoir si elle a abouti. Utilisez `netTextResult` pour renvoyer le texte récupéré par `getNetText`.

Cette fonction est utilisable avec des URL relatives.

Pour un exemple d'utilisation de `getNextText()` dans une animation, reportez-vous à l'animation Form and Post du dossier Learning/Lingo, lui-même inclus dans le dossier de Director.

Paramètres

URL Requis. URL du fichier qui contient le texte à récupérer.

listeDePropriétés Facultatif. Spécifie une liste de propriétés utilisée pour les requêtes CGI.

chaîneOSduServeur Facultatif. Spécifie le codage des caractères de retour chariot dans la liste *listeDePropriétés*.

jeuDeCaractères Facultatif. Spécifie les paramètres des caractères.

Exemple

Le script suivant récupère du texte à partir de l'URL http://grandServeur.fr/exemple.txt et met à jour l'acteur champ sur lequel la souris se trouve lorsque l'utilisateur clique dessus :

```
property spriteNum
property IDréseau

on mouseUp me
  IDréseau = getNetText ("http://grandServeur.fr/exemple.txt")
end

on exitFrame me
  if netDone(IDréseau) then
    sprite(spriteNum).member.text = netTextResult(IDréseau)
  end if
end
```

L'exemple suivant récupère les résultats d'une requête CGI :

```
getNetText("http://www.votreServeur.fr/cgi-bin/requête.cgi?nom=Jean")
```

Cet exemple est similaire au précédent, mais utilise une liste de propriétés pour soumettre une requête CGI et effectue automatiquement le codage URL :

```
getNetText("http://www.votreServeur.fr/cgi-bin/requête.cgi", [#nom:"Jean"])
```

Voir aussi

`netDone()`, `netError()`, `netTextResult()`

getNormalized

Utilisation

```
getNormalized(vecteur)
vecteur.getNormalized()
```

Description

Méthode 3D de vecteur ; copie le vecteur et divise les composants x, y et z de la copie par la longueur du vecteur d'origine. Le vecteur résultant a une longueur correspondant à une unité de l'univers.

Cette méthode renvoie la copie et n'affecte pas le vecteur d'origine. Pour normaliser le vecteur d'origine, utilisez la commande `normalize`.

Exemple

L'instruction suivante enregistre la valeur normalisée du vecteur monVecteur dans la variable Norm. Norm a pour valeur vector(-0.1199, 0.9928, 0.0000) et 1 pour magnitude.

```
monVecteur = vector(-209.9019, 1737.5126, 0.0000)
Norm = monVecteur.getNormalized()
put Norm
-- vector( -0.1199, 0.9928, 0.0000 )
put Norm.magnitude
-- 1.0000
```

Voir aussi

`normalize`

getNthFileNameInFolder()

Utilisation

```
getNthFileNameInFolder(cheminDeDossier, numéroDeFichier)
```

Description

Méthode d'animation ; renvoie un nom de fichier du dossier en fonction du chemin d'accès et du numéro spécifiés dans ce dossier. Pour être détectées par la fonction `getNthFileNameInFolder` les animations Director doivent être définies comme visibles dans la structure des dossiers. Sur Macintosh, d'autres types de fichiers peuvent être détectés, qu'ils soient visibles ou pas. Si cette fonction renvoie une chaîne vide, vous avez spécifié un nombre supérieur au nombre de fichiers dans le dossier.

La fonction `getNthFileNameInFolder` ne peut pas être utilisée avec des URL.

Pour spécifier d'autres noms de dossiers, utilisez l'opérateur `@ pathname` ou le chemin d'accès complet défini dans le format de la plate-forme sur laquelle l'animation est exécutée.
Par exemple :

- Sous Windows, utilisez un chemin tel que C:\Director\Animations.
- Sur Macintosh, utilisez un chemin tel que DisqueDur:Director:Animations. Pour chercher des fichiers se trouvant sur le bureau du Macintosh, utilisez le chemin DisqueDur:Desktop Folder.
- Cette fonction n'est pas disponible dans Shockwave Player.

Paramètres

`cheminDeDossier` Requis. Spécifie le chemin d'accès du dossier qui contient le fichier.

`numéroDeFichier` Requis. Spécifie la position d'index du fichier dans le dossier.

Exemple

Le gestionnaire suivant renvoie une liste de noms de fichiers dans le dossier du chemin courant. Pour appeler cette fonction, utilisez des parenthèses, comme dans `put dossierCourant()`.

```
on dossierCourant
  listeDeFichiers = [ ]
  repeat with i = 1 to 100
    n = getNthFileNameInFolder(the moviePath, i)
    if n = EMPTY then exit repeat
    listeDeFichiers.append(n)
  end repeat
  return listeDeFichiers
end dossierCourant
```

Voir aussi

`@ (chemin d'accès)`, `Animation`

getOne()

Utilisation

```
liste.getOne(valeur)
getOne(liste, valeur)
```

Description

Fonction de liste ; identifie la position (liste linéaire) ou la propriété (liste de propriétés) associée à une valeur dans une liste.

Pour les valeurs contenues plus d'une fois dans la liste, seule la première occurrence est affichée. La commande `getOne` renvoie le résultat 0 lorsque la valeur spécifiée n'est pas dans la liste.

Lorsque utilisée avec des listes linéaires, la commande `getOne` réalise les mêmes fonctions que la commande `getPos`.

Paramètres

`valeur` Requis. Spécifie la valeur associée à la position ou la propriété.

Exemple

L'instruction suivante identifie la position de la valeur 12 dans la liste linéaire Réponses, composée de [10, 12, 15, 22] :

```
put Réponses.getOne(12)
```

Le résultat est 2, 12 étant la seconde valeur de la liste.

L'instruction suivante identifie la propriété associée à la valeur 12 dans la liste de propriétés Réponses, composée de [#a:10, #b:12, #c:15, #d:22] :

```
put Réponses.getOne(12)
```

Le résultat est #b, correspondant à la propriété associée à la valeur 12.

Voir aussi

`getPos()`

getOSDirectory()

Utilisation

```
-- Syntaxe Lingo
réfObjFileio.getOSDirectory()

// Syntaxe JavaScript
réfObjFileio.getOSDirectory();
```

Description

Méthode de Fileio ; renvoie le chemin complet du dossier System (Macintosh) ou du répertoire Windows (Windows).

Paramètres

Aucun.

Voir aussi

`Fileio`

getPixel()

Utilisation

```
-- Syntaxe Lingo
réfObjImage.getPixel(x, y {, #entier})
réfObjImage.getPixel(point(x, y) {, #entier})

// Syntaxe JavaScript
réfObjImage.getPixel(x, y {, #entier});
réfObjImage.getPixel(point(x, y) {, #entier});
```

Description

Méthode d'image. Renvoie une couleur indexée ou RVB du pixel en un point spécifié d'une image donnée.

L'index des lignes et des colonnes de l'image renvoyée commence à 0. Par conséquent, pour accéder au pixel situé dans le coin supérieur gauche d'une image, spécifiez l'emplacement `(0,0)`, et non `(1,1)`. Si une image donnée a une hauteur de *h* pixels et une largeur de *w* pixels, pour accéder au pixels situé dans le coin inférieur droit de l'image, spécifiez l'emplacement `(w,1)`, `(h,1)`.

Cette méthode renvoie une valeur égale à 0 si le pixel spécifié se trouve en dehors de l'image donnée.

Pour affecter à un grand nombre de pixels la couleur d'un autre pixel, la méthode la plus rapide consiste à les définir en tant que nombres bruts (en utilisant le paramètre facultatif `#entier`). Les valeurs de couleur brutes sont également pratiques en ce sens qu'elles contiennent à la fois des informations de couche alpha et de couleurs dans les images 32 bits. Les informations de couche alpha peuvent être extraites du nombre brut en divisant le nombre par 2^8+8+8.

Paramètres

x Requis si vous spécifiez un pixel avec des coordonnées `x` et `y`. Entier qui spécifie la coordonnée `x` du pixel.

y1 Requis si vous spécifiez un pixel avec des coordonnées `x` et `y`. Entier qui spécifie la coordonnée `y` du pixel.

#entier Facultatif. Symbole qui spécifie le nombre brut de la valeur de couleur renvoyée.

`point(x, y)` Requis si vous spécifiez un pixel en utilisant un point. Point qui spécifie le point du pixel.

Exemple

Les instructions suivantes définissent la couleur du pixel au point (90, 20) de l'acteur Joyeux et définissent l'image-objet 2 sur cette couleur.

L'instruction suivante définit la variable alpha en fonction de la valeur de couche alpha du point (25, 33) dans l'objet image 32 bits monImage.

Voir aussi

`color()`, `image()`, `power()`, `setPixel()`

getPlayList()

Utilisation

```
-- Syntaxe Lingo
réfObjPisteAudio.getPlayList()

// Syntaxe JavaScript
réfObjPisteAudio.getPlayList();
```

Description

Méthode de piste audio ; renvoie une copie de la liste des sons mis en file d'attente pour une piste audio.

La liste obtenue n'inclut pas le son en cours de lecture et ne peut pas être éditée directement. Vous devez utiliser `setPlayList()`.

La liste de lecture est une liste linéaire de listes de propriétés. Chaque liste de propriétés correspond à un acteur son placé en file d'attente. Chaque son placé en file d'attente peut spécifier ces propriétés :

Propriété	Description
`#member`	L'acteur à placer en file d'attente. Cette propriété doit être spécifiée. Toutes les autres sont facultatives.
`#startTime`	Position temporelle de départ de la lecture du son, en millisecondes. La valeur par défaut est le début du son. Pour plus d'informations, consultez `startTime`.
`#endTime`	Position temporelle de fin de la lecture du son, en millisecondes. La valeur par défaut est la fin du son. Pour plus d'informations, consultez `endTime`.
`#loopCount`	Nombre de répétitions d'une boucle défini avec `#loopStartTime` et `#loopEndTime`. La valeur par défaut est 1. Voir `loopCount`.
`#loopStartTime`	Position temporelle de départ de la boucle, en millisecondes. Pour plus d'informations, consultez `loopStartTime`.
`#loopEndTime`	Position temporelle de fin de la boucle, en millisecondes. Pour plus d'informations, consultez `loopEndTime`.
`#preloadTime`	Quantité de son à placer en mémoire tampon avant la lecture, en millisecondes. Pour plus d'informations, consultez `preloadTime`.

Paramètres

Aucun.

Exemple

Le gestionnaire suivant place deux sons en file d'attente dans la piste audio 2, démarre leur lecture, puis affiche la liste de lecture dans la fenêtre Messages. La liste de lecture ne contient que le second son placé en file d'attente, le premier son étant déjà en cours de lecture.

```
-- Syntaxe Lingo
on playMusic
  sound(2).queue(member("Carillon"))
  sound(2).queue([#member:member("intro"), #startTime:3000, \
  #endTime:10000, #loopCount:5, #loopStartTime:8000, #loopEndTime:8900])
  put(sound(2).getPlayList())
  sound(2).play()
end playMusic

// Syntaxe JavaScript
function playMusic() {
  sound(2).queue(member("Carillon"));
  sound(2).queue(propList("member",member("intro"), "startTime",3000,
  "endTime",10000, "loopCount",5, "loopStartTime",8000, "loopEndTime",8900));
  put(sound(2).getPlayList());
  sound(2).play();
}
```

Voir aussi

endTime, loopCount, loopEndTime, loopStartTime, Acteur, member, preLoadTime, queue(), setPlayList(), Piste audio, startTime

getPosition()

Utilisation

```
-- Syntaxe Lingo
réfObjFileio.getPosition()

// Syntaxe JavaScript
réfObjFileio.getPosition();
```

Description

Méthode de Fileio ; renvoie la position d'un fichier.

Paramètres

Aucun.

Voir aussi

Fileio

getPref()

Utilisation

```
-- Syntaxe Lingo
_player.getPref(chaîneNomDePréf)

// Syntaxe JavaScript
_player.getPref(chaîneNomDePréf);
```

Description

Méthode de lecteur ; récupère le contenu du fichier spécifié.

Lorsque vous utilisez cette méthode, remplacez *chaîneNomDePréf* par le nom d'un fichier créé par la méthode `setPref()`. Si un tel fichier n'existe pas, `getPref()` renvoie `VOID` (Lingo) ou `null` (syntaxe JavaScript).

Le nom de fichier utilisé pour *chaîneNomDePréf* doit être un nom de fichier valide, et non un chemin d'accès complet ; le chemin est fourni par Director. Le chemin vers le fichier est géré par Director. Les seules extensions de fichiers valides pour *chaîneNomDePréf* sont .txt et .htm ; toute autre extension est rejetée.

N'utilisez pas cette commande pour accéder à des médias en lecture seule ou verrouillés.

Remarque : Dans un navigateur, les données écrites par `setPref()` ne sont pas confidentielles. Toute animation avec un contenu Shockwave est en mesure de lire ces informations et de les télécharger vers un serveur. Les informations confidentielles ne doivent donc pas être stockées à l'aide de la commande `setPref()`.

Pour un exemple d'utilisation de `getPref()` dans une animation, reportez-vous à l'animation Read and Write Text du dossier Learning/Lingo, lui-même inclus dans le dossier de Director.

Paramètres

chaîneNomDePréf Requis. Chaîne qui spécifie le fichier dont le contenu est récupéré.

Exemple

Le gestionnaire suivant récupère le contenu du fichier Test et en affecte le texte au champ Score :

```
-- Syntaxe Lingo
on mouseUp
  leTexte = _player.getPref("Test")
  member("Score").text = leTexte
end

// Syntaxe JavaScript
function mouseUp() {
  var leTexte = _player.getPref("Test");
  member("Score").text = leTexte;
}
```

Voir aussi

Lecteur, setPref()

getPos()

Utilisation

```
liste.getPos(valeur)
getPos(liste, valeur)
```

Description

Fonction de liste ; identifie la position d'une valeur dans une liste. Lorsque la valeur spécifiée n'est pas dans la liste, la commande `getPos` renvoie la valeur 0.

Pour les valeurs contenues plus d'une fois dans la liste, seule la première occurrence est affichée. Cette commande réalise la même fonction que la commande `getOne` lorsque utilisée pour les listes linéaires.

Paramètres

`valeur` Requis. Spécifie la valeur associée à la position.

Exemple

L'instruction suivante identifie la position de la valeur 12 dans la liste Réponses, composée de [#a:10, #b:12, #c:15, #d:22] :

```
put Réponses.getPos(12)
```

Le résultat est 2, 12 étant la seconde valeur de la liste.

Voir aussi

`getOne()`

getPref()

Utilisation

```
getPref(nomDeFichierDePréférences)
```

Description

Fonction ; récupère le contenu du fichier spécifié.

Lorsque vous utilisez cette fonction, remplacez *`nomDeFichierDePréférences`* par le nom du fichier créé par la fonction `setPref`. Si un tel fichier n'existe pas, `getPref` renvoie `VOID`.

Le nom de fichier utilisé pour *`nomDeFichierDePréférences`* doit être un nom de fichier valide et non un chemin d'accès complet ; Director fournit le chemin. Le chemin vers le fichier est géré par Director. Les seules extensions de fichiers valides pour *`nomDeFichierDePréférences`* sont .txt et .ht, toute autre extension étant rejetée.

N'utilisez pas cette commande pour accéder à des médias en lecture seule ou verrouillés.

Remarque : Dans un navigateur web, les données écrites par `setPref` ne sont pas confidentielles. Toute animation avec un contenu Shockwave est en mesure de lire ces informations et de les télécharger vers un serveur. Les informations confidentielles ne doivent donc pas être stockées à l'aide de la commande `setPref`.

Pour un exemple d'utilisation de `getPref()` dans une animation, reportez-vous à l'animation Read and Write Text du dossier Learning/Lingo, lui-même inclus dans le dossier de Director.

Paramètres

nomDeFichierDePréférences Requis. Spécifie le fichier à partir duquel le contenu est récupéré.

Exemple

Le gestionnaire suivant récupère le contenu du fichier Test et en affecte le texte au champ Score :

```
on mouseUp
  leTexte = getPref("Test")
  member("Score").text = leTexte
end
```

Voir aussi

setPref()

getProp()

Utilisation

```
getProp(liste, propriété)
liste.propriété
```

Description

Fonction de liste de propriétés ; identifie la valeur associée à une propriété dans une liste de propriétés.

Presque identique à la commande getaProp, la commande getProp affiche un message d'erreur si la propriété spécifiée n'est pas dans la liste ou si vous spécifiez une liste linéaire.

Paramètres

liste Requis. Spécifie la liste de propriétés dans laquelle la *propriété* est récupérée.

propriété Requis. Spécifie la propriété à laquelle la valeur identifiée est associée.

Exemple

L'instruction suivante identifie la valeur associée à la propriété #c dans la liste de propriétés Réponses, composée de [#a:10, #b:12, #c:15, #d:22] :

```
getProp(Réponses, #c)
```

Le résultat est 15, qui est la valeur associée à #c.

Voir aussi

getOne()

getPropAt()

Utilisation

```
liste.getPropAt(index)
getPropAt(liste, index)
```

Description

Fonction de liste de propriétés ; pour les listes de propriétés uniquement, identifie le nom de la propriété associée à une position spécifiée dans une liste de propriétés. Si l'élément spécifié n'est pas dans la liste ou si vous utilisez getPropAt() avec une liste linéaire, une erreur de script a lieu.

Paramètres

index Requis. Spécifie la position d'index de la propriété dans la liste de propriétés.

Exemple

L'instruction suivante affiche la seconde propriété de la liste donnée :

```
put Réponses.getPropAt(2)
-- #b
```

Le résultat est 20, qui est la valeur associée à #b.

getRendererServices()

Utilisation

```
getRendererServices()
getRendererServices().quellePropriétéDeGetRendererServices
```

Description

Commande 3D ; renvoie l'objet rendererServices. Cet objet contient les informations sur le matériel et les propriétés qui affectent tous les acteurs et images-objets 3D.

L'objet rendererServices a les propriétés suivantes :

- renderer indique le rasteriseur logiciel utilisé pour le rendu des images-objets 3D.
- rendererDeviceList renvoie une liste des rasteriseurs logiciels présents sur le système de l'utilisateur. Les valeurs possibles sont #openGL, #directX5_2, #directX7_0 et #software. La valeur de renderer doit être une de celles-ci. Cette propriété peut être testée, mais pas définie.
- textureRenderFormat indique le format de pixel utilisé par le moteur de rendu. Les valeurs possibles sont #rgba8888, #rgba8880, #rgba5650, #rgba5550, #rgba5551 et #rgba4444. Les quatre chiffres de chaque symbole indiquent le nombre de bits utilisés pour chaque composante rouge, verte, bleue et alpha.
- depthBufferDepth indique le codage binaire du tampon de sortie matériel.
- colorBufferDepth indique le codage binaire du tampon des couleurs. Cette propriété peut être testée, mais pas définie.
- modifiers est une liste linéaire des modificateurs disponibles et pouvant être utilisées par les modèles des acteurs 3D. Les valeurs possibles sont #collision, #bonesPlayer, #keyframePlayer, #toon, #lod, #meshDeform, #sds, #inker et des modificateurs basés sur des Xtras publiés par des tiers. Cette propriété peut être testée, mais pas définie.
- primitives est une liste linéaire des types de primitives disponibles et pouvant être utilisés dans la création de ressources de modèle. Les valeurs possibles sont #sphere, #box, #cylinder, #plane, #particle et des types de primitives basés sur des Xtras publiés par d'autres éditeurs. Cette propriété peut être testée, mais pas définie.

Remarque : Pour plus d'informations, consultez les entrées des différentes propriétés.

Paramètres

Aucun.

Voir aussi

renderer, preferred3dRenderer, active3dRenderer, rendererDeviceList

getStreamStatus()

Utilisation

```
getStreamStatus(IDréseau)
getStreamStatus(chaîneURL)
```

Description

Fonction ; renvoie une liste de propriétés correspondant au format utilisé pour la fonction `tellStreamStatus` disponible globalement et pouvant être utilisée avec des appels d'images-objets ou d'objets. La liste contient les chaînes suivantes :

#URL	Chaîne contenant l'emplacement de l'URL utilisée pour le début des opérations réseau.
#state	Chaîne composée de `Connecting` (connexion), `Started` (démarrée), `InProgress` (en cours), `Complete` (terminée), `Error` (erreur) ou `NoInformation` (aucune information) ; cette dernière chaîne sert lorsque l'ID réseau est tellement ancienne que les informations d'état n'existent plus ou que l'URL spécifiée dans `chaîneURL` n'a pas été trouvée dans la mémoire cache.
#bytesSoFar	Nombre d'octets récupérés jusqu'à maintenant à partir du réseau.
#bytesTotal	Nombre total d'octets dans le flux, s'il est connu. Cette valeur peut être zéro si le serveur HTTP n'indique pas la longueur du contenu dans l'en-tête MIME.
#error	Chaîne contenant "" (`EMPTY`) si le téléchargement n'est pas terminé, OK s'il a abouti ou un code d'erreur si le téléchargement s'est terminé par une erreur.

Par exemple, vous pouvez démarrer une opération réseau avec `getNetText( )` et suivre sa progression avec `getStreamStatus()`.

Paramètres

`IDréseau` Facultatif. Opération réseau qui représente le flux de texte sur lequel intervenir.

Exemple

L'instruction suivante affiche dans la fenêtre Messages l'état courant d'un téléchargement démarré avec `getNetText()` et l'identifiant réseau obtenu, placé dans la variable `IDréseau` :

```
put getStreamStatus(IDréseau)
-- [#URL: "www.macromedia.com", #state: "InProgress", #bytesSoFar: 250, \
  #bytesTotal: 50000, #error: EMPTY]
```

Voir aussi

`on streamStatus`, `tellStreamStatus()`

getVariable()

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet).getVariable(nomDeVariable {, valeurOuRéfRenvoyée})

// Syntaxe JavaScript
réfObjImageObjet.getVariable(nomDeVariable {, valeurOuRéfRenvoyée});
```

Description

Cette fonction renvoie la valeur courante de la variable donnée de l'image-objet donnée. Les variables Flash ont été introduites dans la version 4 de Flash.

Cette fonction peut être utilisée de deux façons.

La définition du paramètre optionnel *valeurOuRéfRenvoyée* sur TRUE (valeur par défaut) renvoie la valeur courante de la variable en tant que chaîne. La définition du paramètre *valeurOuRéfRenvoyée* sur FALSE renvoie la valeur littérale courante de la variable Flash.

Si la valeur de la variable Flash est une référence d'objet, vous devez définir le paramètre *valeurOuRéfRenvoyée* sur FALSE pour que la valeur renvoyée soit traitée en tant que référence d'objet. Si elle est renvoyée en tant que chaîne, la chaîne ne constituera pas une référence d'objet valide.

Paramètres

nomDeVariable Requis. Spécifie le nom de la variable dont la valeur est renvoyée.

valeurOuRéfRenvoyée Facultatif. Spécifie si la valeur renvoyée est une chaîne (TRUE) ou une référence d'objet (FALSE).

Exemple

L'instruction suivante définit la variable valeurT sur la valeur de la chaîne de la variable Flash appelée gAutreVar dans l'animation Flash, au niveau de l'image-objet 3 :

```
-- Syntaxe Lingo
valeurT = sprite(3).getVariable("gAutreVar", TRUE)
put(valeurT) -- "5"
// Syntaxe JavaScript
var valeurT = sprite(3).getVariable("gAutreVar", true);
trace(valeurT); // 5
```

L'instruction suivante définit la variable objetT de manière à référencer le même objet que la variable appelée gVar dans l'animation Flash, au niveau de l'image-objet 3 :

```
-- Syntaxe Lingo
objetT = sprite(3).getVariable("gVar",FALSE)
// Syntaxe JavaScript
var objetT = sprite(3).getVariable("gVar",0);
```

L'instruction suivante renvoie la valeur de la variable `URLactuelle` de l'acteur Flash de l'image-objet 3 et l'affiche dans la fenêtre Messages.

```
-- Syntaxe Lingo
put(sprite(3).getVariable("URLactuelle"))
// Syntaxe JavaScript
trace(sprite(3).getVariable("URLactuelle"));
```

Voir aussi

`setVariable()`

getWorldTransform()

Utilisation

```
member(quelActeur).node(quelNœud).getWorldTransform()
member(quelActeur).node(quelNoeud).getWorldTransform().\
  position
member(quelActeur).node(quelNœud).getWorldTransform().\
  rotation
member(quelActeur).node(quelNœud).getWorldTransform().scale
```

Description

Commande 3D ; renvoie une transformation relative à l'univers du modèle, du groupe, de la caméra ou de la lumière, représenté par `nœud`.

La propriété `transform` du nœud est calculée en fonction de la transformation du parent du nœud et est donc relative au parent. La commande `getWorldTransform()` calcule la transformation du nœud par rapport à l'origine de l'univers 3D ; elle est donc relative à l'univers.

Utilisez `member(quelActeur).node(quelNœud).getWorldTransform().position` pour rechercher la propriété position de la transformation relative à l'univers du nœud. Vous pouvez également utiliser `worldPosition` comme raccourci de `getWorldTransform().position`.

Utilisez `member(quelActeur).node(quelNœud).getWorldTransform().rotation` pour rechercher la propriété rotation de la transformation relative à l'univers du nœud.

Utilisez `member(quelActeur).node(quelNœud).getWorldTransform().scale` pour rechercher la propriété scale de la transformation relative à l'univers du nœud.

Ces propriétés peuvent être testées, mais pas définies.

Exemple

L'instruction suivante indique la transformation relative à l'univers du modèle Boîte, suivie de ses propriétés de position et de rotation.

```
put member("Univers 3D").model("Boîte").getworldTransform()
-- transform(1.000000,0.000000,0.000000,0.000000, \
  0.000000,1.000000,0.000000,0.000000, \
  0.000000,0.000000,1.000000,0.000000, -\
  94.144844,119.012825,0.000000,1.000000)
put member("Univers 3D").model("Boîte").getworldTransform().position
-- vector( -94.1448, 119.0128, 0.0000 )
put member("Univers 3D").model("Boîte").getworldTransform().rotation
-- vector( 0.0000, 0.0000, 0.0000 )
```

Voir aussi

`worldPosition`, `transform (propriété)`

go()

Utilisation

```
-- Syntaxe Lingo
_movie.go(nomOuNumDimage {, nomDanimation})

// Syntaxe JavaScript
_movie.go(nomOuNumDimage {, nomDanimation});
```

Description

Méthode d'animation ; place la tête de lecture sur l'image spécifiée dans une animation donnée.

Cette méthode peut être employée pour demander à la tête de lecture d'effectuer une boucle vers le repère précédent, offrant ainsi un moyen pratique de conserver la tête de lecture dans la même partie de l'animation pendant que le script reste actif.

Il est préférable d'utiliser des libellés de repères plutôt que des numéros d'images pour *nomOuNumDimage*, la modification d'une animation pouvant entraîner le changement des numéros d'images. L'utilisation de libellé de repères facilite également la lecture des scripts.

Lorsque vous appelez `go()` avec le paramètre *nomDanimation*, l'image 1 de l'animation est chargée. Si la méthode `go()` est appelée à partir d'un gestionnaire, le gestionnaire dans laquelle elle est placée poursuit son exécution. Pour suspendre le gestionnaire pendant la lecture de l'animation, utilisez la méthode `play()`, qui peut être suivie d'un appel de `playDone()` pour revenir en arrière.

Lorsque vous spécifiez une animation à lire, indiquez son chemin d'accès si l'animation se trouve dans un dossier différent, mais, pour éviter un éventuel échec du chargement, n'incluez pas l'extension du fichier de l'animation (.dir, .dxr ou .dcr).

Pour passer plus facilement à une animation sur une URL, utilisez la méthode `downloadNetThing()` afin de télécharger le fichier de l'animation sur un disque local, puis utilisez la méthode `go()` avec le paramètre *nomDanimation* cette animation sur le disque local.

La méthode `goLoop()` envoie la tête de lecture vers le repère précédent et offre ainsi un moyen pratique de conserver la tête de lecture dans la même partie de l'animation pendant que la syntaxe Lingo ou JavaScript reste active.

Les paramètres suivants sont réinitialisés au chargement d'une animation : propriétés `beepOn` et `constraint` ; `keyDownScript`, `mouseDownScript` et `mouseUpScript` ; propriétés d'images-objets `cursor` et `immediate` ; méthodes `cursor()` et `puppetSprite()` ; et menus personnalisés. Cependant, la propriété `timeoutScript` n'est pas réinitialisée au chargement d'une animation.

Paramètres

nomOuNumDimage Requis. Chaîne qui spécifie le repère de l'image sur laquelle la tête de lecture se place ou nombre entier qui spécifie le numéro de cette image.

nomDanimation Facultatif. Chaîne qui spécifie l'animation contenant l'image définie par *nomOuNumDimage*. Cette valeur doit spécifier un fichier d'animation ; si l'animation se trouve dans un autre dossier, *nomDanimation* doit également spécifier le chemin d'accès.

Exemple

L'instruction suivante envoie la tête de lecture au point de repère intitulé Début :

```
-- Syntaxe Lingo
_movie.go("Début")

// Syntaxe JavaScript
_movie.go("Début");
```

L'instruction suivante envoie la tête de lecture au point de repère Mémoire de l'animation intitulée Mon animation :

```
-- Syntaxe Lingo
_movie.go("Mémoire", "Mon animation")

// Syntaxe JavaScript
_movie.go("Mémoire", "Mon animation");
```

Le gestionnaire suivant indique à l'animation de faire une boucle sur l'image courante. Ce gestionnaire est pratique pour faire en sorte que l'animation s'arrête sur une image pendant la lecture tout en répondant aux événements.

```
-- Syntaxe Lingo
on exitFrame
  _movie.go(_movie.frame)
end

// Syntaxe JavaScript
function exitFrame() {
  _movie.go(_movie.frame);
}
```

Voir aussi

downloadNetThing, goLoop(), Animation

goLoop()

Utilisation

```
-- Syntaxe Lingo
_movie.goLoop()

// Syntaxe JavaScript
_movie.goLoop();
```

Description

Méthode d'animation ; envoie la tête de lecture au point de repère précédent dans l'animation, un point de repère avant l'image courante si l'image courante n'a pas de repère ou à l'image courante si l'image courante possède un repère.

Si aucun repère ne se trouve à gauche de la tête de lecture, la tête de lecture se place sur :

- Le prochain repère à droite si l'image courante ne possède pas de repère.
- L'image courante si celle-ci possède un repère.
- L'image 1 si l'animation ne contient aucun repère.

Paramètres

Aucun.

Exemple

L'instruction suivante fait faire une boucle à l'animation entre l'image courante et le repère précédent :

```
-- Syntaxe Lingo
_movie.goLoop()

// Syntaxe JavaScript
_movie.goLoop();
```

Voir aussi

go(), goNext(), goPrevious(), Animation

goNext()

Utilisation

```
-- Syntaxe Lingo
_movie.goNext()

// Syntaxe JavaScript
_movie.goNext();
```

Description

Méthode d'animation ; place la tête de lecture sur le repère suivant dans l'animation.

Si aucun repère ne se trouve à droite de la tête de lecture, celle-ci se place sur le dernier repère de l'animation ou sur l'image 1 si l'animation ne contient aucun repère.

Paramètres

Aucun.

Exemple

L'instruction suivante envoie la tête de lecture sur le repère suivant de l'animation :

```
-- Syntaxe Lingo
_movie.goNext()

// Syntaxe JavaScript
_movie.goNext();
```

Voir aussi

go(), goLoop(), goPrevious(), Animation

goPrevious()

Utilisation

```
-- Syntaxe Lingo
_movie.goPrevious()

// Syntaxe JavaScript
_movie.goPrevious();
```

Description

Méthode d'animation ; place la tête de lecture sur le point de repère précédent dans l'animation.

Ce repère est situé deux repères avant l'image courante si celle-ci ne possède pas de repère ou un repère avant l'image courante si elle en possède un.

Si aucun repère ne se trouve à gauche de la tête de lecture, la tête de lecture se place sur un des éléments suivants :

- Le prochain repère à droite si l'image courante ne possède pas de repère.
- L'image courante si celle-ci possède un repère.
- L'image 1 si l'animation ne contient aucun repère.

Paramètres

Aucun.

Exemple

L'instruction suivante envoie la tête de lecture sur le repère précédent dans l'animation :

```
-- Syntaxe Lingo
_movie.goPrevious()

// Syntaxe JavaScript
_movie.goPrevious();
```

Voir aussi

`go()`, `goLoop()`, `goNext()`, `Animation`

goToFrame()

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.goToFrame(nomOuNumDimage)

// Syntaxe JavaScript
réfObjImageObjet.goToFrame(nomOuNumDimage);
```

Description

Commande ; lit une image-objet d'animation Flash à partir de l'image identifiée par le paramètre *nomOuNumDimage*. Vous pouvez identifier l'image par un nombre entier indiquant le numéro d'image ou par une chaîne indiquant le nom du libellé. L'utilisation de la commande `goToFrame` a le même effet que la définition de la propriété `frame` d'une image-objet d'animation Flash.

Exemple

Le gestionnaire suivant passe à différents points d'une animation Flash dans la piste 5. Il accepte un paramètre qui indiquant l'image à laquelle il doit passer.

```
-- Syntaxe Lingo
on Navigate(Cible)
   sprite(5).goToFrame(Cible)
end

// Syntaxe JavaScript
function Navigate(Cible) {
   sprite(5).goToFrame(Cible);
}
```

gotoNetMovie

Utilisation

```
gotoNetMovie URL
gotoNetMovie (URL)
```

Description

Commande ; récupère et lit une nouvelle animation avec un contenu Shockwave à partir d'un serveur HTTP ou FTP. L'exécution de l'animation courante continue jusqu'à ce que la nouvelle animation soit disponible.

Seules les URL sont supportées comme paramètres valides. L'URL peut spécifier un nom de fichier ou un repère dans une animation. Les URL relatives fonctionnent si l'animation se trouve sur un serveur Internet, mais vous devez inclure l'extension avec le nom de fichier.

Lorsque vous réalisez des tests sur un disque ou réseau local, les médias doivent se trouver dans un répertoire appelé dswmedia.

Si une opération `gotoNetMovie` est en cours et que vous envoyez une seconde commande `gotoNetMovie` avant la fin de la première, la seconde commande annule la première.

Paramètres

URL Requis. Spécifie l'URL du contenu Shockwave à lire.

Exemple

Dans l'instruction suivante, l'URL indique un nom de fichier Director :

```
gotoNetMovie "http://www.votreserveur.fr/animations/animation1.dcr"
```

Dans l'instruction suivante, l'URL indique un repère dans un nom de fichier :

```
gotoNetMovie "http://www.votreserveur.fr/animations/boutons.dcr#Contenu"
```

Dans l'instruction suivante, `gotoNetMovie` est utilisée comme une fonction. Cette fonction renvoie l'identifiant réseau pour l'opération.

```
monIDréseau = gotoNetMovie ("http://www.votreServeur.fr/animations/
  boutons.dcr#Contenu")
```

gotoNetPage

Utilisation

```
gotoNetPage "URL", {"nomDeCible"}
```

Description

Commande ; ouvre une animation avec un contenu Shockwave ou un autre fichier MIME dans le navigateur.

Seules les URL sont supportées comme paramètres valides. Les URL relatives fonctionnent si l'animation se trouve sur un serveur HTTP ou FTP.

Dans l'environnement de programmation, la commande `gotoNetPage` lance le navigateur préféré s'il est activé. Dans les projections, cette commande essaie de lancer le navigateur préféré, défini dans la boîte de dialogue des préférences réseau ou la commande `browserName`. Si aucune commande n'a été utilisée pour définir le navigateur préféré, la commande `goToNetPage` essaie de trouver un navigateur web sur l'ordinateur.

Paramètres

`URL` Requis. Spécifie l'URL de l'animation avec un contenu Shockwave ou du fichier MIME à lire.

`nomDeCible` Facultatif. Paramètre HTML qui identifie le cadre ou la fenêtre dans lequel la page est chargée.

- Si *`nomDeCible`* est une fenêtre ou un cadre dans le navigateur web, `gotoNetPage` en remplace le contenu.
- Si *`nomDeCible`* n'est pas un cadre ou une fenêtre ouverts, `goToNetPage` ouvre une nouvelle fenêtre. L'utilisation de la chaîne "_new" provoque systématiquement l'ouverture d'une nouvelle fenêtre.
- Si *`nomDeCible`* est omis, `gotoNetPage` remplace la page courante, où qu'elle se trouve.

Exemple

Le script suivant charge le fichier nouvellePage.html dans le cadre ou la fenêtre intitulé(e) frwin. S'il existe une fenêtre ou un cadre intitulés frwin, cette fenêtre ou ce cadre est utilisé. Si la fenêtre frwin n'existe pas, une nouvelle fenêtre intitulée frwin est créée.

```
on keyDown
  gotoNetPage "nouvellePage.html", "frwin"
end
```

Le gestionnaire suivant ouvre une nouvelle fenêtre, quelle que soit la fenêtre ouverte par le navigateur :

```
on mouseUp
  goToNetPage "Nouvelles_du_jour.html", "_new"
end
```

Voir aussi

`browserName()`, `netDone()`

group()

Utilisation

```
member(quelActeur).group(quelGroupe)
member(quelActeur).group[index]
```

Description

Elément 3D ; nœud de l'univers 3D qui a un nom, une transformation, un parent et des enfants, mais aucune autre propriété.

Chaque acteur 3D est associé à un groupe par défaut nommé Univers et qui ne peut pas être supprimé. La hiérarchie parent de tous les modèles, lumières, caméras et groupes qui existent dans l'univers 3D se terminent dans `group("world")`.

Exemple

L'instruction suivante indique que le quatrième groupe de l'acteur nouveauMartien est le groupe Direct01.

```
put member("nouveauMartien").group[4]
-- group("Direct01")
```

Voir aussi

`newGroup`, `deleteGroup`, `child (3D)`, `parent`

halt()

Utilisation

```
-- Syntaxe Lingo
_movie.halt()

// Syntaxe JavaScript
_movie.halt();
```

Description

Méthode d'animation ; quitte le gestionnaire courant et tout autre gestionnaire qui l'a appelé et arrête l'animation pendant la programmation ou quitte la projection pendant son exécution.

Paramètres

Aucun.

Exemple

L'instruction suivante vérifie si la quantité de mémoire disponible est inférieure à 50 ko et, le cas échéant, quitte tous les gestionnaires qui l'ont appelée et arrête l'animation :

```
-- Syntaxe Lingo
if (_system.freeBytes < (50*1024)) then
  _movie.halt()
end if

// Syntaxe JavaScript
if (_system.freeBytes < (50*1024)) {
  _movie.halt();
}
```

Voir aussi

Animation

handler()

Utilisation

objetScript.handler(#*symboleDeGestionnaire*)

Description

Cette fonction renvoie TRUE si l'*objetScript* donné contient un gestionnaire spécifié et FALSE dans le cas contraire. L'objet script doit être un script parent, un objet enfant ou un comportement.

Paramètres

symboleDeGestionnaire Requis. Spécifie le nom du gestionnaire.

Exemple

Le code Lingo suivant appelle un gestionnaire sur un objet, uniquement si le gestionnaire en question existe :

```
if objetAraignée.handler(#saut) = TRUE then
  objetAraignée.saut()
end if
```

Voir aussi

handlers(), new(), rawNew(), script()

handlers()

Utilisation

objetScript.handlers()

Description

Cette fonction renvoie une liste linéaire des gestionnaires dans l'*objetScript* donné. Chaque nom de gestionnaire est présenté sous forme de symbole dans la liste. Cette fonction est pratique pour le débogage des animations.

Vous ne pouvez pas accéder directement aux gestionnaires d'un acteur script. Vous devez y accéder par l'intermédiaire de la propriété script de l'acteur.

Paramètres

Aucun.

Exemple

L'instruction suivante affiche la liste des gestionnaires de l'objet enfant voitureRouge dans la fenêtre Messages :

```
put voitureRouge.handlers()
-- [#accélérer, #tourner, #arrêter]
```

L'instruction suivante affiche la liste des gestionnaires du script parent ScriptParentDeVoiture dans la fenêtre Messages :

```
put member("ScriptParentDeVoiture").script.handlers()
-- [#accélérer, #tourner, #arrêter]
```

Voir aussi

handler(), script()

hilite (commande)

Utilisation

```
expressionSousChaîneChamp.hilite()
```

```
hilite expressionSousChaîneChamp
```

Description

Commande ; sélectionne l'expression de sous-chaîne spécifiée dans l'image-objet champ, qui peut être n'importe quelle sous-chaîne que Lingo vous permet de définir, telle qu'un caractère, un mot ou une ligne. Sur Macintosh, la couleur de sélection est définie dans le tableau de bord Couleur.

Paramètres

Aucun.

Exemple

L'instruction suivante sélectionne le troisième mot dans l'acteur champ Commentaires, qui contient la chaîne Pensée du jour :

```
member("Commentaires").word[4].hilite()
```

L'instruction suivante entraîne l'affichage du texte sélectionné dans le champ mesRecettes de l'image-objet sans surbrillance :

```
monNombreDeLignes = member("mesRecettes").line.count
member("mesRecettes").line[monNombreDeLignes + 1].hilite()
```

Voir aussi

char...of, item...of, line...of, word...of, delete(), mouseChar, mouseLine, mouseWord, field, selection() (fonction), selEnd, selStart

hitTest()

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.hitTest(point)

// Syntaxe JavaScript
réfObjImageObjet.hitTest(point);
```

Description

Fonction ; indique la partie d'une animation Flash se trouvant directement sur un emplacement spécifique de la scène Director. L'emplacement de la scène Director est exprimé en valeur de points Director : par exemple, point (100,50). La fonction `hitTest` renvoie ces valeurs :

- `#background` – L'emplacement spécifié est dans l'arrière-plan de l'image-objet animation Flash.
- `#normal` – L'emplacement spécifié est dans une image-objet remplie.
- `#button` – L'emplacement spécifié est dans la zone référencée d'un bouton.
- `#editText` – L'emplacement spécifié est dans un champ de texte modifiable Flash.

Paramètres

`point` Requis. Spécifie le point à tester.

Exemple

Le script d'image suivant vérifie si la souris se trouve au-dessus d'un bouton dans une image-objet animation Flash dans la piste 5 et, le cas échéant, le script définit un champ de texte utilisé pour afficher un message d'état :

```
-- Syntaxe Lingo
on exitFrame
  if sprite(5).hitTest(_mouse.mouseLoc) = #button then
    member("Ligne de message").text = "Cliquez ici pour lire l'animation."
    _movie.updatestage()
  else
    member("message").text = " "
  end if
  _movie.go(_movie.frame)
end

// Syntaxe JavaScript
function exitFrame() {
  var hT = sprite(5).hitTest(_mouse.mouseLoc);
  if (hT = "bouton") {
    member("Ligne de message").text = "Cliquez ici pour lire l'animation.";
    _movie.updatestage();
  } else {
    member("Ligne de message").text = "";
  }
  _movie.go(_movie.frame)
}
```

HMStoFrames()

Utilisation

`HMStoFrames(hms, cadence, compensé, fractions)`

Description

Fonction ; convertit les animations mesurées en heures, minutes et secondes au nombre équivalent d'images ou convertit un nombre d'heures, minutes et secondes en temps si vous donnez à l'argument *cadence* la valeur 1 (1 image = 1 seconde).

Paramètres

hms Requis. Expression de chaîne qui spécifie le temps sous la forme `sHH:MM:SS.FFD`, où :

s	Un caractère est utilisé si le temps est inférieur à zéro ou un espace si le temps est supérieur ou égal à zéro.
HH	Heures.
MM	Minutes.
SS	Secondes.
FF	Indique une fraction de seconde si `fractions` a la valeur `TRUE` ou des images si `fractions` a la valeur `FALSE`.
D	Un d est utilisé si *composé* a la valeur `TRUE` ou un espace si *composé* a la valeur `FALSE`.

cadence Requis. Spécifie la cadence en images par seconde.

compensé Requis. Expression logique qui détermine si l'image est compensée (`TRUE`) ou non (`FALSE`). Si la chaîne *hms* se termine par *d*, le temps est traité comme une image compensée, quelle que soit la valeur de l'argument *compensé*.

fractions Requis. Expression logique qui détermine la signification des nombres après les secondes ; il peut s'agir de fractions de secondes arrondies au centième de seconde le plus proche (`TRUE`) ou du nombre d'images résiduelles (`FALSE`).

Exemple

L'instruction suivante détermine le nombre d'images dans une animation d'une minute, 30,1 secondes lorsque la cadence est de 30 images par seconde. Les arguments *compensé* et *fractions* ne sont pas utilisés.

```
put HMStoFrames(" 00:01:30.10 ", 30, FALSE, FALSE)
-- 2710
```

L'instruction suivante convertit 600 secondes en minutes :

```
put framesToHMS(600, 1,0,0)
>> -- " 00:10:00.00 "
```

L'instruction suivante convertit une heure et demie en secondes :

```
put HMStoFrames("1:30:00", 1,0,0)
>> -- 5400
```

Voir aussi

`framesToHMS()`

hold()

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.hitTest(point)

// Syntaxe JavaScript
réfObjImageObjet.hitTest(point);
```

Description

Commande Flash ; arrête une image-objet animation Flash lue dans l'image courante, sans interrompre la lecture audio.

Paramètres

Aucun.

Exemple

Le script d'image suivant arrête la lecture des images-objets animation Flash dans les pistes 5 à 10 sans interrompre la lecture audio de ces pistes :

```
-- Syntaxe Lingo
on enterFrame
  repeat with i = 5 to 10
    sprite(i).hold()
  end repeat
end

// Syntaxe JavaScript
function enterFrame() {
  var i = 5;
  while (i < 11) {
    sprite(i).hold();
    i++;
  }
}
```

Voir aussi

playRate (QuickTime, AVI)

identity()

Utilisation

```
member(quelActeur).model(quelModèle).transform.identity()
member(quelActeur).group(quelGroupe).transform.identity()
member(quelActeur).camera(quelleCaméra).transform.identity()
sprite(quelleImageObjet).camera{(index)}.transform.identity()
member(quelActeur).light(quelleLumière).transform.identity()
référenceDeTransformation.identity()
```

Description

Commande 3D ; fait de la transformation la transformation d'identité, qui est `transform(1.0000,0.0000,0.0000,0.0000, 0.0000,1.0000,0.0000,0.0000, 0.0000,0.0000,1.0000,0.0000, 0.0000,0.0000,0.0000,1.0000)`.

La propriété `position` de la transformation d'identité est `vector(0, 0, 0)`.

La propriété `rotation` de la transformation d'identité est `vector(0, 0, 0)`.

La propriété `scale` de la transformation d'identité est `vector(1, 1, 1)`.

La transformation d'identité est relative au parent.

Paramètres

Aucun.

Exemple

L'instruction suivante fait de la transformation du modèle Boîte la transformation d'identité.

```
member("Univers 3D").model("Boîte").transform.identity()
```

Voir aussi

transform (propriété), getWorldTransform()

idleLoadDone()

Utilisation

```
-- Syntaxe Lingo
_movie.idleLoadDone(entBaliseDeChargement)

// Syntaxe JavaScript
_movie.idleLoadDone(entBaliseDeChargement);
```

Description

Méthode d'animation ; indique si tous les acteurs ayant la balise spécifiée ont été chargés (`TRUE`) ou s'ils doivent encore être chargés (`FALSE`).

Paramètres

`entBaliseDeChargement` Requis. Nombre entier qui spécifie la balise de chargement des acteurs à tester.

Exemple

L'instruction suivante vérifie si tous les acteurs dont la balise de chargement est 20 ont été chargés et, le cas échéant, lit l'animation Kiosque :

```
-- Syntaxe Lingo
if (_movie.idleLoadDone(20)) then
  _movie.play(1, "Kiosque")
end if

// Syntaxe JavaScript
if (_movie.idleLoadDone(20)) {
  _movie.play(1, "Kiosque");
}
```

Voir aussi

idleHandlerPeriod, idleLoadMode, idleLoadPeriod, idleLoadTag, idleReadChunkSize, Animation

ignoreWhiteSpace()

Utilisation

objetDanalyseXML.ignoreWhiteSpace(*trueOuFalse*)

Description

Commande XML ; spécifie si l'objet d'analyse doit ignorer ou conserver les espaces vierges dans les listes Lingo. Lorsque `ignoreWhiteSpace()` a pour valeur `TRUE` (la valeur par défaut), l'objet d'analyse ignore les espaces vierges. Lorsque cette commande a pour valeur `FALSE`, l'objet d'analyse conserve les espaces vierges et les traite comme des données réelles.

Lorsqu'un élément possède des balises initiales et finales distinctes, telles que `<exemple> </exemple>`, les caractères de l'élément sont ignorés si (et uniquement si) il n'est composé que d'espaces vierges. En cas de présence d'un espace autre qu'un espace vierge, ou si `ignoreWhiteSpace()` a pour valeur `FALSE`, un nœud CDATA contenant le même texte y compris l'espace vierge, sera créé.

Paramètres

trueOuFalse Requis. Valeur qui spécifie si l'objet d'analyse doit ignorer les espaces vierges (`TRUE`) ou non (`FALSE`).

Exemple

Les instructions Lingo suivantes laissent à `ignoreWhiteSpace()` sa valeur par défaut de `TRUE` et convertissent le document XML donné en une liste. L'élément `<exemple>` ne possède pas d'enfant dans la liste.

```
TexteXML = "<exemple> </exemple>"
ObjetDanalyse.parseString(TexteXML)
laListe = ObjetDanalyse.makelist()
put laListe
-- ["ROOT OF XML DOCUMENT": ["!ATTRIBUTES": [:], "exemple": ["!ATTRIBUTES":
  [:]]]]
```

Les instructions Lingo suivantes donnent à `ignoreWhiteSpace()` la valeur `FALSE` et convertissent le document XML donné en une liste. L'élément `<exemple>` possède maintenant un enfant contenant un espace.

```
TexteXML = "<exemple> </exemple>"
objetDanalyse.ignoreWhiteSpace(FALSE)
ObjetDanalyse.parseString(TexteXML)
laListe = ObjetDanalyse.makelist()
put laListe
-- ["ROOT OF XML DOCUMENT": ["!ATTRIBUTES": [:], "exemple": ["!ATTRIBUTES":
  [:], "!CHARDATA": " "]]]
```

Les instructions Lingo suivantes laissent à `ignoreWhiteSpace()` sa valeur par défaut de `TRUE` et analysent le document XML donné. Il n'existe qu'un seul nœud enfant de la balise `<exemple>` et qu'un seul nœud enfant de la balise `<inf>`.

```
texteXML = "<exemple> <inf> phrase 1 </inf></exemple>"
ObjetDanalyse.parseString(TexteXML)
laListe = objetDanalyse.makeList()
put laListe
-- ["ROOT OF XML DOCUMENT": ["!ATTRIBUTES": [:], "exemple": ["!ATTRIBUTES":
  [:], "inf": ["!ATTRIBUTES": [:], "!CHARDATA": " phrase 1 "]]]]
```

Les instructions Lingo suivantes donnent à `ignoreWhiteSpace()` la valeur `FALSE` et analysent le document XML donné. Il existe maintenant deux nœuds enfants de la balise `<exemple>`, le premier étant un caractère d'espace.

```
texteXML = "<exemple> <inf> phrase 1 </inf></exemple>"
gObjetDanalyse.ignoreWhiteSpace(FALSE)
gObjetDanalyse.parseString(texteXML)
laListe = gObjetDanalyse.makeList()
put laListe
-- ["ROOT OF XML DOCUMENT": ["!ATTRIBUTES": [:], "exemple": ["!ATTRIBUTES":
  [:], "!CHARDATA": " ", "inf": ["!ATTRIBUTES": [:], "!CHARDATA": " phrase 1
  "]]]]
```

ilk()

Utilisation

```
ilk(objet)
ilk(objet, type)
```

Description

Fonction ; indique le type d'un objet.

Le tableau suivant présente la valeur renvoyée pour chaque type d'objet reconnu par `ilk()` :

Type d'objet	`ilk(objet)` renvoie	`ilk(objet, type)` renvoie 1 uniquement si le `type =`	Exemple
liste linéaire	`#list`	`#list` ou `#linearlist`	`ilk ([1,2,3])`
liste de propriétés	`#proplist`	`#list` ou `#proplist`	`ilk ([#Alui: 1234, #Aelle: 7890])`
entier	`#integer`	`#integer` ou `#number`	`ilk (333)`
valeur à virgule flottante	`#float`	`#float` ou `#number`	`ilk (123.456)`
chaîne	`#string`	`#string`	`ilk ("asdf")`
rect	`#rect`	`#rect` ou `#list`	`ilk (sprite(1).rect)`
point	`#point`	`#point` ou `#list`	`ilk (sprite(1).loc)`
couleur	`#color`	`#color`	`ilk (sprite(1).color)`
date	`#date`	`#date`	`ilk (the systemdate)`
symbole	`#symbol`	`#symbol`	`ilk (#bonjour)`
void	`#void`	`#void`	`ilk (void)`
image	`#picture`	`#picture`	`ilk (member (2).picture)`
instance de script parent	`#instance`	`#object`	`ilk (new (script "blabla"))`
instance d'Xtra	`#instance`	`#object`	`ilk (new (xtra "fileio"))`

Type d'objet	`ilk(objet)` renvoie	`ilk(objet, type)` renvoie 1 uniquement si le `type =`	Exemple
acteur	`#member`	`#object` ou `#member`	`ilk (member 1)`
Xtra	`#xtra`	`#object` ou `#xtra`	`ilk (xtra "fileio")`
script	`#script`	`#object` ou `#script`	`ilk (script "blabla")`
castlib	`#castlib`	`#object` ou `#castlib`	`ilk (castlib 1)`
image-objet	`#sprite`	`#object` ou `#sprite`	`ilk (sprite 1)`
son	`#instance` ou `#sound` (lorsque l'Xtra de contrôle audio est absent)	`#instance` ou `#sound`	`ilk (sound "tralala")`
fenêtre	`#window`	`#object` ou `#window`	`ilk (the stage)`
médias	`#media`	`#object` ou `#media`	`ilk (member (2).media)`
timeout	`#timeout`	`#object` ou `#timeout`	`ilk (timeOut("compteurIntervalle"))`
image	`#image`	`#object` ou `#image`	`ilk ((the stage).image)`

Paramètres

`objet` Requis. Spécifie l'objet à tester.

`type` Facultatif. Spécifie le type par rapport auquel l'*`objet`* est comparé. Si l'objet est du type spécifié, la fonction `ilk` renvoie la valeur `TRUE`. Si l'objet n'est pas du type spécifié, la fonction `ilk` renvoie la valeur `FALSE`.

Exemple

L'instruction `ilk` suivante identifie le type de l'objet appelé Devis :

```
Devis = [:]
put ilk( Devis )
-- #proplist
```

L'instruction suivante `ilk` teste si la variable Total est une liste et affiche le résultat dans la fenêtre Messages :

```
Total = 2+2
put ilk( Total, #list )
-- 0
```

Dans ce cas, étant donné que la variable n'est pas une liste, la fenêtre Messages affiche 0, l'équivalent numérique de FALSE.

L'exemple suivant teste une variable intitulée maVariable et vérifie qu'elle correspond à un objet de date avant de l'afficher dans la fenêtre Messages :

```
maVariable = the systemDate
if ilk(maVariable, #date) then put maVariable
-- date( 1999, 2, 19 )
```

ilk (3D)

Utilisation

```
ilk(objet)
ilk(objet,type)
objet.ilk
objet.ilk(type)
```

Description

Fonction Lingo ; indique le type d'un objet.

Le tableau suivant présente la valeur renvoyée pour chaque type d'objet 3D reconnu par `ilk()`. Le dictionnaire Lingo principal contient une liste des valeurs renvoyées pour les objets autres que 3D.

Type d'objet	`ilk(objet)` renvoie	`ilk(objet, type)` uniquement si `type =`
services de rendu	`#renderer`	`#renderer`
ressource de modèle	`#modelResource`, `#plane`, `#box`, `#sphere`, `#cylinder`, `#particle`, `#mesh`	Identique à ilk(objet), à l'exception de `#modelResource` qui est équivalent aux ressources générées par un fichier *.w3d importé
modèle	`#model`	`#model`
mouvement	`#motion`	`#motion` ou `#list`
matériau	`#shader`	`#shader` ou `#list`
texture	`#texture`	`#texture` ou `#list`
groupe	`#group`	`#group`
caméra	`#camera`	`#camera`
données de collision	`#collisiondata`	`#collisiondata`
vecteurs	`#vector`	`#vector`
transformation	`#transform`	`#transform`

Paramètres

`objet` Requis. Spécifie l'objet à tester.

`type` Facultatif. Spécifie le type par rapport auquel l'*`objet`* est comparé. Si l'objet est du type spécifié, la fonction `ilk` renvoie la valeur `TRUE`. Si l'objet n'est pas du type spécifié, la fonction `ilk` renvoie la valeur `FALSE`.

Exemple

L'instruction suivante indique que monObjet est un objet de mouvement.

```
put MonObjet.ilk
-- #motion
```

L'instruction suivante vérifie si monObjet est un objet de mouvement. La valeur renvoyée, 1, indique qu'il s'agit bien d'un tel objet.

```
put MonObjet.ilk(#motion)
-- 1
```

Voir aussi

tweenMode

image()

Utilisation

```
-- Syntaxe Lingo
image(entLargeur, entHauteur, entProfondeurDeBit)

// Syntaxe JavaScript
image(entLargeur, entHauteur, entProfondeurDeBit);
```

Description

Fonction du niveau supérieur ; crée et renvoie une nouvelle image avec des dimensions spécifiées.

Si vous créez une nouvelle image en utilisant la fonction `image()` du niveau supérieur, la nouvelle image est un ensemble de données d'images autonome et elle est indépendante de toutes les autres images. Les modifications apportées aux autres images n'ont aucune incidence sur la nouvelle image.

Si vous faites référence à une image en définissant une variable égale à une image source telle qu'un acteur ou l'image de la scène, la variable contient une référence à l'image source. Une modification apportée à l'image dans l'objet source ou la variable sera répercutée dans l'autre image.

Pour éviter ce comportement et créer une copie d'une image indépendante de l'image source, utilisez la méthode `duplicate()`. La méthode `duplicate()` renvoie une copie d'une image source qui hérite de toutes les valeurs de l'image source, mais qui n'est pas liée à cette dernière. Par conséquent, une modification apportée à l'image source ou à la nouvelle copie de l'image source n'aura aucune incidence sur l'autre image.

Si un objet image est créé en référence à un acteur, l'objet contiendra une référence à l'image de l'acteur. Toutes les modifications apportées à l'image sont répercutées dans l'acteur et dans les images-objets créées à partir de cet acteur.

Lorsque vous créez un nouvel objet image, par défaut, la couleur d'arrière-plan est blanche (color(255,255,255)) et la couche alpha est complètement opaque (color(0,0,0)).

La couleur de la couche alpha pour une transparence de 100 % est le blanc (color(255,255,255)) ; pour une opacité de 100 %, la couleur de la couche alpha est le noir (color(0,0,0)).

Pour un exemple d'utilisation d'une méthode `image()` dans une animation, reportez-vous à l'animation Imaging du dossier Learning/Lingo, lui-même inclus dans le dossier de Director.

Paramètres

entLargeur Requis. Nombre entier qui spécifie la largeur de la nouvelle image.

entHauteur Requis. Nombre entier qui spécifie la hauteur de la nouvelle image.

entProfondeurDeBit Requis. Nombre entier qui spécifie la profondeur de bit de la nouvelle image. Les valeurs correctes sont 1, 2, 4, 8, 16 et 32.

Exemple

L'exemple suivant crée une image 8 bits d'une largeur de 200 pixels et d'une hauteur de 200 pixels.

```
-- Syntaxe Lingo
monImage = image(200, 200, 8)

// Syntaxe JavaScript
var objImage = image(200, 200, 8);
```

L'exemple suivant crée une image en faisant référence à l'image de la scène.

```
-- Syntaxe Lingo
objImage = _movie.stage.image

// Syntaxe JavaScript
var objImage = _movie.stage.image;
```

Voir aussi

duplicate() (image), fill(), image (image)

importFileInto()

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.importFileInto(chaîneFichierOuUrl)

// Syntaxe JavaScript
réfObjActeur.importFileInto(chaîneFichierOuUrl);
```

Description

Méthode d'acteur ; remplace le contenu d'un acteur donné par un fichier spécifié.

La méthode `importFileInto()` est pratique dans les situations suivantes.

- Lorsque vous finalisez ou développez une animation, elle permet d'intégrer des médias externes liés de façon à pouvoir les modifier pendant le projet.
- Lorsque vous créez un scénario à partir d'une syntaxe Lingo ou JavaScript pendant la création de l'animation, elle permet d'affecter un contenu aux nouveaux acteurs.
- Lors du téléchargement de fichiers à partir d'Internet, elle permet de télécharger le fichier à partir d'une URL spécifique et de définir le nom de fichier du média lié.

  Remarque : Pour importer un fichier à partir d'une URL, il est généralement plus efficace de télécharger d'abord le fichier sur un disque local avec `preloadNetThing()`, puis de l'importer depuis le disque local. La fonction `preloadNetThing()` minimise également les problèmes de téléchargement potentiels.

- Elle permet d'importer des documents RTF et HTML dans des acteurs texte en gardant le format et les liens intacts.

L'utilisation de la méthode `importFileInto()` dans les projections peut rapidement consommer la mémoire disponible et il est donc plus judicieux de réutiliser, si possible, les mêmes acteurs pour les données importées.

Dans Director et les projections, `importFileInto()` télécharge automatiquement le fichier. Dans Shockwave Player, appelez `preloadNetThing()` et attendez la fin du téléchargement avant d'utiliser `importFileInto()` avec le fichier.

Paramètres

`chaîneFichierOuUrl` Requis. Chaîne qui spécifie le fichier qui remplacera le contenu de l'acteur.

Exemple

Le gestionnaire suivant affecte une URL contenant un fichier GIF à la variable `URLtemporaire`, puis utilise la commande `importFileInto` pour importer le fichier de l'URL dans un nouvel acteur bitmap :

```
-- Syntaxe Lingo
on exitFrame
    URLtemporaire = "http://www.uneUrl.fr/couronne.gif"
  _movie.newMember(#bitmap).importFileInto(URLtemporaire)
end

// Syntaxe JavaScript
function exitFrame() {
  var URLtemporaire = "http://www.dukeOfUrl.com/crown.gif";
  _movie.newMember("bitmap").importFileInto(URLtemporaire);
}
```

L'instruction suivante remplace le contenu de l'acteur son Mémoire par le fichier audio Vent :

```
-- Syntaxe Lingo
member("Mémoire").importFileInto("Vent.wav")

// Syntaxe JavaScript
member("Mémoire").importFileInto("Vent.wav");
```

Les instructions suivantes téléchargent un fichier externe depuis une URL dans le dossier de Director et importent ce fichier dans l'acteur son Norma Desmond parle :

```
-- Syntaxe Lingo
downLoadNetThing("http://www.cbDeMille.com/CinémaParlant.AIF", \
  _player.applicationPath & "CinémaParlant.AIF")
member("Norma Desmond parle").importFileInto(_player.applicationPath &
  "CinémaParlant.AIF")

// Syntaxe JavaScript
downLoadNetThing("http://www.cbDeMille.com/CinémaParlant.AIF",
  _player.applicationPath + "CinémaParlant.AIF");
member("Norma Desmond parle").importFileInto(_player.applicationPath +
  "CinémaParlant.AIF");
```

Voir aussi

downloadNetThing, fileName (fenêtre), Acteur, preloadNetThing()

insertBackdrop

Utilisation

```
sprite(quelleImageObjet).camera{(index)}.insertBackdrop(index, \
  texture, emplacementDansLimageObjet, rotation)
member(quelActeur).camera(quelleCaméra).\
  insertBackdrop(index, texture, emplacementDansLimageObjet, rotation)
```

Description

Commande 3D de caméra ; ajoute un fond à la liste des fonds de la caméra, à une position spécifiée dans la liste.

Paramètres

index Requis. Spécifie la position d'index où le fond est ajouté dans la liste des fonds de la caméra.

texture Requis. Spécifie la texture du fond ajouté.

emplacementDansLimageObjet Requis. Emplacement 2D dont le fond est affiché dans l'image-objet 3D. Cet emplacement est mesuré à partir du coin supérieur gauche de l'image-objet.

rotation Facultatif. Spécifie la rotation texture du fond ajouté.

Exemple

La première ligne de l'exemple suivant crée une texture nommée Cèdre. La deuxième ligne insère cette texture à la première position de la liste des fonds de la caméra de l'image-objet 5. Le fond est placé au point (300, 120), mesuré à partir du coin supérieur gauche de l'image-objet. La rotation est de 45°.

```
t1 = member("Scène").texture("Cèdre")
sprite(5).camera.insertBackdrop(1, t1, point(300, 120), 45)
```

Voir aussi

removeBackdrop, bevelDepth, overlay

insertFrame()

Utilisation

```
-- Syntaxe Lingo
_movie.insertFrame()

// Syntaxe JavaScript
_movie.insertFrame();
```

Description

Méthode d'animation ; crée une copie de l'image courante et de son contenu.

L'image copiée est insérée après l'image courante et devient alors l'image courante.

Cette méthode ne peut être utilisée que pendant une session d'enregistrement du scénario et remplit la même fonction que la méthode `duplicateFrame()`.

Paramètres

Aucun.

Exemple

Le gestionnaire suivant crée une image dans laquelle l'acteur transition Brouillard est affecté à la piste des transitions, suivie d'un groupe d'images vides. L'argument `nombreDimages` définit ce nombre d'images.

```
-- Syntaxe Lingo
on animBalle(nombreDimages)
  _movie.beginRecording()
    _movie.frameTransition = member("Brouillard").number
    _movie.go(_movie.frame +1)
    repeat with i = 0 to nombreDimages
      _movie.insertFrame()
    end repeat
  _movie.endRecording()
end animBalle

// Syntaxe JavaScript
function animBalle(nombreDimages) {
  _movie.beginRecording();
    _movie.frameTransition = member("Brouillard").number;
    _movie.go(_movie.frame+1);
    for (var i = 0; i <= nombreDimages; i++) {
      _movie.insertFrame();
    }
  _movie.endRecording();
}
```

Voir aussi

`duplicateFrame()`, `Animation`

insertOverlay

Utilisation

```
sprite(quelleImageObjet).camera{(index)}.insertOverlay(index, \
  texture, emplacementDansLimageObjet, rotation)
member(quelActeur).camera(quelleCaméra).\
  insertOverlay(index, texture, \
  emplacementDansLimageObjet, rotation)
```

Description

Commande 3D de caméra ; ajoute un recouvrement à la liste des recouvrements de la caméra, à une position spécifiée dans la liste.

Paramètres

`index` Requis. Spécifie la position d'index où le recouvrement est ajouté dans la liste des recouvrements de la caméra.

`texture` Requis. Spécifie la texture du recouvrement ajouté.

`emplacementDansLimageObjet` Requis. Emplacement 2D auquel où le recouvrement est affiché dans l'image-objet 3D. Cet emplacement est mesuré à partir du coin supérieur gauche de l'image-objet.

`rotation` Facultatif. Spécifie la rotation texture du recouvrement ajouté.

Exemple

La première ligne de l'exemple suivant crée une texture nommée Cèdre. La deuxième ligne insère cette texture à la première position de la liste des recouvrements de la caméra de l'image-objet 5. Le recouvrement est placé au point (300, 120), mesuré à partir du coin supérieur gauche de l'image-objet. La rotation est de 45°.

```
t1 = member("Scène").texture("Cèdre")
sprite(5).camera.insertOverlay(1, t1, point(300, 120), 45)
```

Voir aussi

removeOverlay, overlay, bevelDepth

inside()

Utilisation

```
point.inside(rectangle)
inside(point, rectangle)
```

Description

Fonction ; indique si un point donné se trouve à l'intérieur d'un rectangle spécifié (TRUE) ou en dehors (FALSE).

Paramètres

rectangle Requis. Spécifie le rectangle qui contient le point à tester.

Exemple

L'instruction suivante indique si le point Centre se trouve à l'intérieur du rectangle Zone et affiche le résultat dans la fenêtre Messages :

```
put Centre.inside(Zone)
```

Voir aussi

map(), mouseH, mouseV, point()

installMenu

Utilisation

```
installMenu réfObjActeurChamp
```

Description

Commande ; installe le menu défini dans l'acteur champ spécifié par *réfObjActeurChamp*. Ces menus personnalisés n'apparaissent que pendant la lecture de l'animation. Pour les supprimer, utilisez la commande installMenu sans argument, ou avec 0 comme argument. Cette commande ne fonctionne pas avec les menus hiérarchiques.

Pour plus de détails sur la définition d'articles de menus dans les acteurs champ, consultez le mot-clé menu.

Il est conseillé d'éviter la modification fréquente des menus, celle-ci affectant les ressources du système.

Sous Windows, si le menu ne tient pas à l'écran, seule une partie est affichée. Sur Macintosh, vous pouvez faire défiler les menus qui ne tiennent pas à l'écran.

Remarque : Les menus ne sont pas disponibles dans Shockwave Player.

Paramètres

réfObjActeurChamp Facultatif. Spécifie l'acteur champ dans lequel un menu est installé.

Exemple

L'instruction suivante installe le menu défini dans l'acteur champ 37 :

```
installMenu 37
```

L'instruction suivante installe le menu défini dans l'acteur champ Barre de menu :

```
installMenu member "Barre de menu"
```

L'instruction suivante désactive les menus qui ont été installés avec la commande `installMenu` :

```
installMenu 0
```

Voir aussi

`menu`

integer()

Utilisation

```
(expressionNumérique).integer
integer(expressionNumérique)
```

Description

Fonction (Lingo uniquement) ; arrondit la valeur d'une expression au nombre entier le plus proche.

Vous pouvez forcer un nombre entier en une chaîne avec la fonction `string()`.

Dans la syntaxe JavaScript, utilisez la fonction `parseInt()`.

Paramètres

expressionNumérique Requis. La valeur numérique à arrondir à un nombre entier.

Exemple

L'instruction suivante arrondit le nombre 3,75 au nombre entier le plus proche :

```
put integer(3.75)
-- 4
```

L'instruction suivante arrondit la valeur entre parenthèses. Cela permet de donner une valeur utilisable à la propriété d'image-objet `locH`, qui exige un nombre entier :

```
sprite(1).locH = integer(0.333 * largeurDeScène)
```

Voir aussi

`float()`, `string()`

integerP()

Utilisation

```
expression.integerP
(expressionNumérique).integerP
integerP(expression)
```

Description

Fonction (Lingo uniquement) ; indique si une expression spécifiée est un entier (`1 ou TRUE`) ou non (`0 ou FALSE`). Le *P* de la fonction `integerP` signifie *prédicat*.

Paramètres

`expression` Requis. Expression à tester.

Exemple

L'instruction suivante vérifie si le nombre 3 est un entier et affiche 1 (`TRUE`) dans la fenêtre Messages :

```
put(3).integerP
-- 1
```

L'instruction suivante vérifie si le nombre 3 est un entier. Puisque le nombre 3 est entouré de guillemets droits, il n'est pas considéré comme un entier et 0 (`FALSE`) est affiché dans la fenêtre Messages :

```
put("3").integerP
-- 0
```

L'instruction suivante vérifie si la valeur numérique de la chaîne de l'acteur champ Saisie est un entier et affiche une alerte si ce n'est pas le cas :

```
if field("Saisie").value.integerP = FALSE then alert "Saisissez un entier."
```

Voir aussi

`floatP()`, `integer()`, `ilk()`, `objectP()`, `stringP()`, `symbolP()`

interface()

Utilisation

```
xtra("nomDeLxtra").interface()
interface(xtra "nomDeLxtra")
```

Description

Fonction ; renvoie une chaîne délimitée par des retours chariot décrivant l'Xtra et fournissant une liste de ses méthodes. Cette fonction remplace maintenant la fonction obsolète `mMessageList`.

Paramètres

Aucun.

Exemple

L'instruction suivante affiche les données produites par cette fonction utilisée avec l'Xtra QuickTime Asset dans la fenêtre Messages :

```
put Xtra("QuickTimeSupport").interface()
```

interpolate()

Utilisation

transformation1.interpolate(*transformation2*,*pourcentage*)

Description

Méthode 3D `transform` ; renvoie une copie de *transformation1* créée par l'interpolation de la position et de la rotation de *transformation1* et de la position et de la rotation de *transformation2*, en fonction du pourcentage spécifié. La *transformation1* d'origine n'est pas affectée. Pour interpoler *transformation1*, utilisez `interpolateTo()`.

Pour effectuer l'interpolation manuellement, multipliez la différence des deux nombres par le pourcentage. Par exemple, l'interpolation de 4 à 8 par 50 % donne 6.

Exemple

Dans l'exemple suivant, tBoîte est la transformation du modèle nommé Boîte et tSphère est la transformation du modèle nommé Sphère. La troisième ligne de l'exemple interpole une copie de la transformation de Boîte à mi-chemin jusqu'à la transformation de Sphère.

```
tBoîte = member("Univers 3D").model("Boîte").transform
tSphère = member("Univers 3D").model("Sphère").transform
tNouv = tBoîte.interpolate(tSphère, 50)
```

Voir aussi

interpolateTo()

interpolateTo()

Utilisation

transformation1.interpolateTo(*transformation2*, *pourcentage*)

Description

Méthode 3D `transform` ; modifie *transformation1* en interpolant la position et la rotation de *transformation1* par la position et la rotation d'une nouvelle transformation en fonction d'un pourcentage spécifié. La *transformation1* d'origine est changée. Pour interpoler une copie de *transformation1*, utilisez la fonction `interpolate()`.

Pour effectuer l'interpolation manuellement, multipliez la différence des deux nombres par le pourcentage. Par exemple, l'interpolation de 4 à 8 par 50 % donne 6.

Paramètres

transformation2 Requis. Spécifie la transformation résultant de l'interpolation d'une transformation donnée.

pourcentage Requis. Spéifie le pourcentage de rotation de *transformation2*.

Exemple

Dans l'exemple suivant, tBoîte est la transformation du modèle nommé Boîte et tSphère est la transformation du modèle nommé Sphère. La troisième ligne de l'exemple interpole la transformation de Boîte à mi-chemin jusqu'à la transformation de Sphère.

```
tBoîte = member("Univers 3D").model("Boîte").transform
tSphère = member("Univers 3D").model("Sphère").transform
tBoîte.interpolateTo(tSphère, 50)
```

Voir aussi

interpolate()

intersect()

Utilisation

```
rectangle1. intersect(rectangle2)
intersect(rectangle1, rectangle2)
```

Description

Fonction ; détermine le rectangle formé par l'intersection de deux rectangles.

Paramètres

rectangle2 Requis. Spécifie le second rectangle du test d'intersection.

Exemple

Cette instruction affecte la variable nouveauRectangle au rectangle formé par l'intersection du rectangle Outils avec le rectangle Rampe :

```
nouveauRectangle = Outils.intersect(Rampe)
```

Voir aussi

map(), rect(), union()

inverse()

Utilisation

```
member(quelActeur).model(quelModèle).transform.inverse()
member(quelActeur).group(quelGroupe).transform.inverse()
member(quelActeur).camera(quelleCaméra).transform.inverse()
sprite(quelleImageObjet).camera{(index)}.transform.inverse()
member(quelActeur).light(quelleLumière).transform.inverse()
référenceDeTransformation.inverse()
```

Description

Méthode 3D transform ; renvoie une copie de la transformation, avec ses propriétés de position et de rotation inversées.

Cette méthode ne change pas la transformation d'origine. Pour inverser la transformation d'origine, utilisez la fonction invert().

Paramètres

Aucun.

Exemple

L'instruction suivante inverse une copie de la transformation du modèle Fauteuil.

```
boxInv = member("Univers 3D").model("Fauteuil").transform.inverse()
```

Voir aussi

invert()

invert()

Utilisation

```
member(quelActeur).model(quelModèle).transform.invert()
member(quelActeur).group(quelGroupe).transform.invert()
member(quelActeur).camera(quelleCaméra).transform.invert()
sprite(quelleImageObjet).camera{(index)}.transform.invert()
member(quelActeur).light(quelleLumière).transform.invert()
référenceDeTransformation.invert()
```

Description

Méthode 3D `transform` ; inverse les propriétés de position et de rotation de la transformation.

Cette méthode change la transformation d'origine. Pour inverser une copie de la transformation d'origine, utilisez la fonction `inverse()`.

Paramètres

Aucun.

Exemple

L'instruction suivante inverse la transformation du modèle Boîte.

```
member("Univers 3D").model("Boîte").transform.invert()
```

Voir aussi

inverse()

isBusy()

Utilisation

```
-- Syntaxe Lingo
réfObjPisteAudio.isBusy()

// Syntaxe JavaScript
réfObjPisteAudio.isBusy();
```

Description

Méthode de piste audio ; détermine si un son est lu (`TRUE`) ou non (`FALSE`) sur une piste audio.

Vérifiez d'abord que la tête de lecture a bougé avant d'utiliser `isBusy()` pour vérifier la piste audio. Si cette fonction continue de renvoyer la valeur `FALSE` après la lecture présumée d'un son, ajoutez la méthode `updateStage()` pour démarrer la lecture du son avant que la tête de lecture ne se remette à bouger.

Cette méthode fonctionne avec les pistes audio occupées par des acteurs son réels. Dans la mesure où les composants audio QuickTime, Flash et Shockwave gèrent le son différemment, cette méthode ne peut pas fonctionner avec ces types de médias.

Il est conseillé d'utiliser la propriété `status` d'une piste audio, plutôt que la propriété `isBusy()`. La propriété `status` s'avère plus appropriée dans la plupart des cas.

Paramètres

Aucun.

Exemple

L'instruction suivante vérifie si un son est lu dans la piste audio 1 et, le cas échéant, effectue une lecture en boucle dans l'image. Cela permet au son d'être lu en entier avant que la tête de lecture ne passe à une autre image.

```
-- Syntaxe Lingo
if (sound(1).isBusy()) then
  _movie.go(_movie.frame)
end if

// Syntaxe JavaScript
if (sound(1).isBusy()) {
  _movie.go(_movie.frame);
}
```

Voir aussi

`status`, `Piste audio`

isInWorld()

Utilisation

```
member(quelActeur).model(quelModèle).isInWorld()
member(quelActeur).camera(quelleCaméra).isInWorld()
member(quelActeur).light(quelleLumière).isInWorld()
member(quelActeur).group(quelGroupe).isInWorld()
```

Description

Commande 3D ; renvoie la valeur `TRUE` si la hiérarchie parent du modèle, de la caméra, de la lumière ou du groupe se termine dans l'univers. Si la valeur de `isInWorld` est `TRUE`, le modèle, la caméra, la lumière, ou le groupe, fonctionne dans l'univers 3D de l'acteur.

Les modèles, caméras, lumières et groupes peuvent être enregistrés dans un acteur 3D mais ne peuvent pas être utilisés dans son univers 3D. Utilisez les commandes `addToWorld` et `removeFromWorld` pour ajouter et supprimer des modèles, des caméras, des lumières et des groupes de l'univers 3D de l'acteur.

Paramètres

Aucun.

Exemple

L'instruction suivante indique que le modèle Théière existe dans l'univers 3D de l'acteur scèneDeTable.

```
put member("ScèneDeTable").model("Théière").isInWorld()
-- 1
```

Voir aussi

addToWorld, removeFromWorld, child (3D)

isPastCuePoint()

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.isPastCuePoint(IDduRepère)

// Syntaxe JavaScript
réfObjImageObjet.isPastCuePoint(IDduRepère);
```

Description

Fonction ; détermine si une piste d'image-objet ou une piste audio est passée par un point de repère spécifié dans ses médias. Cette fonction peut être utilisée avec les fichiers audio (WAV, AIFF, SND, SWA, AU), QuickTime ou Xtra supportant les points de repère.

Remplacez *réfObjImageObjet* par une piste d'image-objet ou une piste audio. Les sons Shockwave Audio (SWA) peuvent apparaître sous forme d'images-objets dans les pistes d'images-objets, mais sont lus dans les pistes audio. Il est conseillé de faire référence aux images-objets audio SWA par le numéro de leur piste d'image-objet plutôt que par celui de leur piste audio.

Remplacez *IDduRepère* par une référence à un point de repère :

- Si *IDduRepère* est un entier, `isPastCuePoint` renvoie 1 si le point de repère a été dépassé et 0 dans le cas contraire.
- Si *IDduRepère* est un nom, `isPastCuePoint` renvoie le nombre de points de repère dépassés portant ce nom.

Si la valeur spécifiée pour *IDduRepère* n'existe pas dans l'image-objet ou le son, la fonction renvoie 0.

Le nombre renvoyé par `isPastCuePoint` est basé sur la position absolue de l'image-objet dans son média. Par exemple, lorsqu'un son dépasse le point de repère Principal, puis se met en boucle et dépasse de nouveau ce point de repère, `isPastCuePoint` renvoie 1 au lieu de 2.

Lorsque le résultat de `isPastCuePoint` est traité comme un opérateur booléen, la fonction renvoie la valeur `TRUE` si un point identifié par *IDduRepère* a été dépassé et `FALSE` si ce n'est pas le cas.

Paramètres

IDduRepère Requis. Chaîne ou nombre entier qui spécifie le nom ou le numéro du point de repère spécifié.

Exemple

L'instruction suivante lit un son jusqu'au troisième passage sur le point de repère Fin du choeur :

```
-- Syntaxe Lingo
if (sound(1).isPastCuePoint("Fin du chœur")=3) then
  sound(1).stop()
end if

// Syntaxe JavaScript
var fc = sound(1).isPastCuePoint("Fin du chœur");
if (fc == 3) {
  sound(1).stop();
}
```

L'exemple suivant permet d'afficher dans l'acteur « champ 2 » des informations concernant la lecture de la musique sur la piste audio 1. Si le point de repère « Apogée » n'a pas encore été atteint, le texte de « champ 2 » est « Début du morceau. » Sinon, le texte est « Fin du morceau. »

```
-- Syntaxe Lingo
if not sound(1).isPastCuePoint("Apogée") then
  member("champ 2").text = "Début du morceau."
else
  member("champ 2").text = "Fin du morceau."
end if

// Syntaxe JavaScript
var fc = sound(1).isPastCuePoint("Apogée");
if (cmx != 1) {
  member("champ 2").text = "Début du morceau.";
} else {
  member("champ 2").text = "Fin du morceau.";
}
```

keyPressed()

Utilisation

```
-- Syntaxe Lingo
_key.keyPressed({codeOuCaractèreDeTouche})

// Syntaxe JavaScript
_key.keyPressed({codeOuCaractèreDeTouche});
```

Description

Méthode de touche ; renvoie la chaîne de caractères affectée à la dernière touche activée ou, éventuellement, indique si une touche spécifiée a été activée.

Si le paramètre *codeOuCaractèreDeTouche* est omis, cette méthode renvoie la chaîne de caractères affectée à la dernière touche activée. Si aucune touche n'a été activée, elle renvoie une chaîne vide.

Lorsque le paramètre *codeOuCaractèreDeTouche* est utilisé pour spécifier la touche actuellement activée, cette méthode renvoie `TRUE` si l'utilisateur appuie sur cette touche spécifique ou `FALSE` dans le cas contraire.

Cette méthode est mise à jour lorsque l'utilisateur appuie sur des touches pendant une boucle `repeat` (Lingo) ou `for` (syntaxe JavaScript). Elle présente un avantage par rapport à la propriété `key`, qui n'est pas mise à jour pendant une boucle `repeat` ou `for`.

Utilisez l'animation Clavier et Lingo pour tester la correspondance entre les caractères et les touches sur différents claviers.

Paramètres

codeOuCaractèreDeTouche Facultatif. Code de touche ou chaîne de caractères ASCII à tester.

Exemple

L'instruction suivante vérifie si l'utilisateur a appuyé sur la touche Entrée sous Windows ou sur la touche Retour du Macintosh et, le cas échéant, exécute le gestionnaire `miseAjourDesDonnées` :

```
-- Syntaxe Lingo
if (_key.keyPressed(RETURN)) then
  miseAjourDesDonnées
end if

// Syntaxe JavaScript
if (_key.keyPressed(36)) {
  miseAjourDesDonnées();
}
```

L'instruction suivante utilise le code de la touche *a* pour vérifier si elle est enfoncée et affiche le résultat dans la fenêtre Messages :

```
-- Syntaxe Lingo
if (_key.keyPressed(0)) then
  put("La touche est enfoncée")
end if

// Syntaxe JavaScript
if (_key.keyPressed(0)) {
  put("La touche est enfoncée");
}
```

L'instruction suivante utilise les chaînes ASCII pour vérifier si les touches *a* et *b* sont enfoncées et affiche le résultat dans la fenêtre Messages :

```
-- Syntaxe Lingo
if (_key.keyPressed("a") and _key.keyPressed("b")) then
  put("Les deux touches sont enfoncées")
end if

// Syntaxe JavaScript
if (_key.keyPressed("a") && _key.keyPressed("b")) {
  put("Les deux touches sont enfoncées");
}
```

Voir aussi

Touche, key, keyCode

label()

Utilisation

```
-- Syntaxe Lingo
_movie.label(chaîneNomDeRepère)

// Syntaxe JavaScript
_movie.label(chaîneNomDeRepère);
```

Description

Méthode d'animation ; indique l'image associée à un libellé de repère.

Le paramètre *chaîneNomDeRepère* doit être un libellé dans l'animation courante. Si ce n'est pas le cas, cette méthode renvoie 0.

Paramètres

chaîneNomDeRepère Requis. Chaîne qui spécifie le nom du libellé de repère associé à une image.

Exemple

L'instruction suivante place la tête de lecture sur la dixième image après le repère intitulé Début :

```
-- Syntaxe Lingo
_movie.go(_movie.label("Début") + 10)

// Syntaxe JavaScript
_movie.go(_movie.label("Début") + 10);
```

Voir aussi

frameLabel, go(), labelList, Animation

last()

Utilisation

```
the last sousChaîne of ( expressionSousChaîne )
the last sousChaîne in (expressionSousChaîne)
```

Description

Fonction ; identifie la dernière sous-chaîne dans une expression de sous-chaîne.

Les expressions de sous-chaînes peuvent correspondre à tout caractère, mot, élément ou ligne extrait d'un conteneur de caractères. Les conteneurs peuvent être des acteurs champ, des variables contenant des chaînes, et les caractères, mots, lignes et plages spécifiés dans les conteneurs.

Paramètres

expressionSousChaîne Requis. Spécifie l'expression de sous-chaîne qui contient la dernière sous-chaîne.

Exemple

L'instruction suivante identifie le dernier mot de la chaîne Macromedia, la société multimédia et affiche le résultat dans la fenêtre Messages :

```
put the last word of "Macromedia, la société multimédia"
```

Le résultat est le mot *multimédia*.

L'instruction suivante identifie le dernier caractère de la chaîne Macromedia, la société multimédia et affiche le résultat dans la fenêtre Messages :

```
put last char("Macromedia, la société multimédia")
```

Le résultat est la lettre *a*.

Voir aussi

`char...of`, `word...of`

lastClick()

Utilisation

```
the lastClick
```

Description

Fonction ; renvoie le nombre de battements (1 battement = 1/60ème de seconde) écoulés depuis le dernier clic de la souris.

Cette fonction peut être testée, mais pas définie.

Paramètres

Aucun.

Exemple

L'instruction suivante vérifie si 10 secondes se sont écoulées depuis le dernier clic de la souris et, le cas échéant, fait passer la tête de lecture sur le repère Pas de clic :

```
if the lastClick > 10 * 60 then go to "Pas de clic"
```

Voir aussi

`lastEvent()`, `lastKey`, `lastRoll`, `milliseconds`

lastEvent()

Utilisation

```
the lastEvent
```

Description

Fonction ; renvoie le nombre de battements (1 battement = 1/60ème de seconde) écoulés depuis la dernière fois où l'utilisateur a appuyé sur le bouton de la souris, a survolé un élément avec la souris ou a appuyé sur une touche.

Paramètres

Aucun.

Exemple

L'instruction suivante vérifie si 10 secondes se sont écoulées depuis la dernière fois où l'utilisateur a appuyé sur le bouton de la souris, a survolé un élément avec la souris ou a appuyé sur une touche. Si c'est le cas, la tête de lecture est placée sur le repère Aide :

```
if the lastEvent > 10 * 60 then go to "Aide"
```

Voir aussi

lastClick(), lastKey, lastRoll, milliseconds

length()

Utilisation

```
chaîne.length
length(chaîne)
```

Description

Fonction ; renvoie le nombre de caractères de la chaîne spécifiée par *chaîne*, y compris les espaces et les caractères de contrôle tels que Tab et Retour.

Paramètres

Aucun.

Exemple

L'instruction suivante affiche le nombre de caractères de la chaîne « Macro »&« media » :

```
put ("Macro" & "media").length
-- 10
```

L'instruction suivante vérifie si le contenu de l'acteur champ Nom de fichier contient plus de 31 caractères et, le cas échéant, affiche un message d'alerte :

```
-- Syntaxe Lingo
if member("Nom de fichier").text.length > 31 then
  alert "Ce nom de fichier est trop long."
end if

// Syntaxe JavaScript
if (member("Nom de fichier").text.length > 31) {
  alert ("Ce nom de fichier est trop long.");
}
```

Voir aussi

chars(), offset() (fonction de chaîne)

light()

Utilisation

```
member(quelActeur).light(quelleLumière)
member(quelActeur).light[index]
member(quelActeur).light(quelleLumière).quellePropriétéDeLumière
member(quelActeur).light[index].quellePropriétéDeLumière
```

Description

Elément 3D ; objet à une position de vecteur à partir de laquelle la lumière émane.

Pour une liste complète des propriétés et des méthodes, consultez les rubriques Utilisation de Director dans le panneau d'aide de Director.

Exemple

L'exemple suivant indique les deux façons de faire référence à une lumière. La première ligne utilise une chaîne entre parenthèses et la seconde utilise un nombre entre crochets. La chaîne correspond au nom de la lumière et le nombre à la position de la lumière dans la liste des lumières de l'acteur.

```
cetteLumière = member("Univers 3D").light("spot01")
cetteLumière = member("Univers 3D").light[2]
```

Voir aussi

`newLight`, `deleteLight`

lineHeight()

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.lineHeight(numéroDeLigne)

// Syntaxe JavaScript
réfObjActeur.lineHeight(numéroDeLigne);
```

Description

Fonction ; renvoie la hauteur, en pixels, d'une ligne spécifique dans un acteur champ spécifié.

Paramètres

numéroDeLigne Requis. Nombre entier qui spécifie la ligne à mesurer.

Exemple

Cette instruction détermine la hauteur, en pixels, de la première ligne de l'acteur champ Nouvelles du jour et affecte le résultat à la variable `titre` :

```
-- Syntaxe Lingo
titre = member("Nouvelles du jour").lineHeight(1)

// Syntaxe JavaScript
var titre = member("Nouvelles du jour").lineHeight(1);
```

linePosToLocV()

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.linePosToLocV(numéroDeLigne)

// Syntaxe JavaScript
réfObjActeur.linePosToLocV(numéroDeLigne);
```

Description

Fonction ; renvoie la distance, en pixels, d'une ligne spécifique à partir du bord supérieur de l'acteur champ spécifié.

Paramètres

numéroDeLigne Requis. Nombre entier qui spécifie la ligne à mesurer.

Exemple

Cette instruction mesure la distance, en pixels, entre la deuxième ligne de l'acteur champ Nouvelles du jour et le bord supérieur de l'acteur et affecte le résultat à la variable débutDeChaîne :

```
-- Syntaxe Lingo
débutDeChaîne = member("Nouvelles du jour").linePosToLocV(2)

// Syntaxe JavaScript
var débutDeChaîne = member("Nouvelles du jour").linePosToLocV(2);
```

linkAs()

Utilisation

```
unActeur.linkAs()
```

Description

Fonction d'acteur script ; ouvre une boîte de dialogue d'enregistrement, permettant d'enregistrer le contenu du script dans un fichier externe. L'acteur script est alors lié à ce fichier.

Les scripts liés sont importés dans l'animation lorsque vous les enregistrez comme projection ou comme animation avec un contenu Shockwave. Ceci diffère des autres médias liés, qui restent externes à l'animation, à moins que vous ne les importiez explicitement.

Paramètres

Aucun.

Exemple

L'instruction suivante, saisie dans la fenêtre Messages, ouvre une boîte de dialogue d'enregistrement, permettant d'enregistrer le script Mouvement aléatoire comme fichier externe :

```
member("Mouvement aléatoire").linkAs()
importFileInto, linked
```

list()

Utilisation

```
-- Syntaxe Lingo
list()
[]
list(valeurDeChaîne1, valeurDeChaîne2, ...)
[valeurDeChaîne1, valeurDeChaîne2, ...]

// Syntaxe JavaScript
list();
list(valeurDeChaîne1, valeurDeChaîne2, ...);
```

Description

Fonction du niveau supérieur ; crée une liste linéaire.

Lorsque vous créez une liste en utilisant la syntaxe `list()`, avec ou sans paramètres, les valeurs de la liste sont indexées à partir de 1.

Lorsque vous créez une liste en utilisant la syntaxe `[]`, avec ou sans paramètres, les valeurs de la liste sont indexées à partir de 0.

La longueur maximale d'une ligne de script exécutable est de 256 caractères. Il n'est pas possible de créer des listes importantes avec `list()`. Pour créer une liste contenant une grande quantité de données, mettez les données entre crochets ([]), placez-les dans un champ, puis affectez le champ à une variable. Le contenu de cette variable deviendra une liste des données.

Paramètres

`valeurDeChaîne1, valeurDeChaîne2 ...` Facultatif. Liste de chaînes qui spécifient les valeurs initiales dans la liste.

Exemple

L'instruction suivante associe la variable Designers à une liste linéaire contenant les noms Jean, Pierre et Marc :

```
-- Syntaxe Lingo
designers = list("Jean", "Pierre", "Marc") - utilisation de list()
designers = ["Jean", "Pierre", "Marc"] - utilisation de crochets

// Syntaxe JavaScript
var designers = list("Jean", "Pierre", "Marc");
```

Voir aussi

`propList()`

listP()

Utilisation

`listP(`*`élément`*`)`

Description

Fonction ; indique si un élément spécifié est une liste, un rectangle ou un point (1 `ou TRUE`) ou non (0 `ou FALSE`).

Paramètres

`élément` Requis. Spécifie l'élément à tester.

Exemple

L'instruction suivante vérifie si la liste contenue dans la variable `designers` est une liste, un rectangle ou un point et affiche le résultat dans la fenêtre Messages :

```
put listP(designers)
```

Le résultat est 1, équivalent numérique de `TRUE`.

Voir aussi

`ilk()`, `objectP()`

loadFile()

Utilisation

```
member(quelActeur).loadFile(nomDeFichier {, écraser?, \
  générerDesNomsUniques?})
```

Description

Commande d'acteur 3D ; importe les actifs d'un fichier *.w3d dans un acteur.

La propriété `state` de l'acteur doit avoir pour valeur -1 (erreur) ou 4 (chargé) avant l'utilisation de la commande `loadFile`.

Paramètres

`nomDeFichier` Requis. Spécifie le fichier *.w3d qui contient les actifs à importer.

`écraser?` Facultatif. Indique si les actifs du fichier *.w3d remplacent ceux de l'acteur (`TRUE`) ou s'ils sont ajoutés à ceux de l'acteur (`FALSE`). La valeur par défaut de *`écraser?`* est `TRUE`.

`générerDesNomsUniques?` Facultatif. Si ce paramètre a pour valeur `TRUE`, tout élément du fichier *.w3d portant le même nom qu'un élément correspondant de l'acteur sera renommé. S'il a pour valeur `FALSE`, les éléments de l'acteur seront remplacés par les éléments correspondants ayant le même nom dans le fichier *.w3d. La valeur par défaut de *`générerDesNomsUniques?`* est `TRUE`.

Exemple

L'instruction suivante importe le contenu du fichier Camion.w3d dans l'acteur Route. Le contenu de Camion.w3d sera ajouté au contenu de Route. Si certains objets importés ont le même nom que des objets déjà présents dans Route, Director leur donnera un nouveau nom.

```
member("Route").loadFile("Camion.w3d", FALSE, TRUE)
```

L'instruction suivante importe le contenu du fichier Chevy.w3d dans l'acteur Route. Chevy.w3d se trouve dans un dossier Modèles, un niveau au-dessous de l'animation. Le contenu de Route sera remplacé par celui de Chevy.w3d. Le troisième paramètre n'a aucune importance étant donné que la valeur du second paramètre est `TRUE`.

```
member("Route").loadFile(the moviePath & "Modèles\Chevy.w3d", \
  TRUE, TRUE)
```

Voir aussi

`state (3D)`

locToCharPos()

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.locToCharPos(emplacement)

// Syntaxe JavaScript
réfObjActeur.locToCharPos(emplacement);
```

Description

Fonction ; renvoie le numéro du caractère de l'acteur champ spécifié qui est le plus proche d'un point dans le champ.

La valeur 1 correspond au premier caractère de la chaîne, la valeur 2 au second caractère de cette chaîne, et ainsi de suite.

Paramètres

emplacement Requis. Point à l'intérieur de l'acteur champ. La valeur de *emplacement* est un point calculé par rapport au coin supérieur gauche de l'acteur champ.

Exemple

L'instruction suivante détermine le caractère le plus proche du point situé à 100 pixels sur la droite et à 100 pixels en dessous du coin supérieur gauche de l'acteur champ Nouvelles du jour. Elle affecte ensuite le résultat à la variable `MiseEnPage`.

```
-- Syntaxe Lingo
miseEnPage = member("Nouvelles du jour").locToCharPos(point(100, 100))

// Syntaxe JavaScript
var miseEnPage = member("Nouvelles du jour").locToCharPos(point(100, 100));
```

locVToLinePos()

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.locVToLinePos(locV)

// Syntaxe JavaScript
réfObjActeur.locVToLinePos(locV);
```

Description

Fonction ; renvoie le numéro de la ligne de caractères apparaissant à une position verticale spécifiée.

Paramètres

emplacementV Requis. Spécifie la position verticale de la ligne de caractères. Cette valeur correspond au nombre de pixels depuis le haut de l'acteur champ, et non depuis la partie de l'acteur champ actuellement affichée sur la scène.

Exemple

L'instruction suivante détermine la ligne de caractères qui apparaît à 150 pixels du haut de l'acteur champ Nouvelles du jour et affecte le résultat à la variable `sautDePage`:

```
-- Syntaxe Lingo
sautDePage = member("Nouvelles du jour").locVToLinePos(150)

// Syntaxe JavaScript
var sautDePage = member("Nouvelles du jour").locVToLinePos(150);
```

log()

Utilisation

```
log(nombre)
```

Description

Fonction mathématique (Lingo uniquement) ; calcule le logarithme naturel d'un nombre spécifié.

Dans la syntaxe JavaScript, utilisez la fonction `log()` de l'objet Math.

Paramètres

`nombre` Requis. Valeur numérique à partir de laquelle le logarithme naturel est calculé. Il doit s'agir d'un nombre entier supérieur à 0.

Exemple

L'instruction suivante affecte le logarithme naturel de 10,5 à la variable `Réponse`.

```
Réponse = log(10.5)
```

Exemple

L'instruction suivante calcule le logarithme de la racine carrée de la valeur `Nombre` et affecte le résultat à la variable `Réponse` :

```
Réponse = log(Nombre.sqrt)
```

makeList()

Utilisation

```
-- Syntaxe Lingo
objetDanalyse.makeList()

// Syntaxe JavaScript
objetDanalyse.makeList();
```

Description

Fonction ; renvoie une liste de propriétés basée sur le document XML analysé à l'aide de `parseString()` ou `parseURL()`.

Paramètres

Aucun.

Exemple

Le gestionnaire suivant analyse un document XML et renvoie la liste résultante :

```
-- Syntaxe Lingo
on conversionEnListe chaîneXML
  objetDanalyse = new(xtra "xmlparser")
  codeErreur = objetDanalyse.parseString(chaîneXML)
  chaîneDerreur = ObjetDanalyse.getError()
  if voidP(chaîneDerreur) then
    listeAnalysée = objetDanalyse.makeList()
  else
    alert "Désolé. Une erreur est survenue" && chaîneDerreur
    exit
  end if
  return listeAnalysée
end

// Syntaxe JavaScript
function conversionEnListe(chaîneXML) {
  objetDanalyse = new Xtra("xmlparser"); // vérifier la syntaxe
  codeErreur = objetDanalyse.parseString(chaîneXML);
  chaîneDerreur = objetDanalyse.getError();
  if (voidP(chaîneDerreur)) {
    listeAnalysée = objetDanalyse.makeList();
  } else {
    alert ("Désolé. Une erreur est survenue" + chaîneDerreur);
    return FALSE;
  }
    return listeAnalysée;
}
```

Voir aussi

`makeSubList()`

makeScriptedSprite()

Utilisation

```
-- Syntaxe Lingo
réfObjPisteDimageObjet.makeScriptedSprite({réfObjActeur, emplacement})

// Syntaxe JavaScript
réfObjPisteDimageObjet.makeScriptedSprite({réfObjActeur, emplacement});
```

Description

Méthode de piste d'image-objet ; bascule le contrôle d'une piste d'image-objet du scénario vers le script et place éventuellement une image-objet d'un acteur donné à un emplacement spécifié sur la scène.

Appelez `removeScriptedSprite()` pour basculer de nouveau le contrôle de l'image-objet vers le scénario.

Paramètres

réfObjActeur Facultatif. Référence à l'acteur à partir duquel une image-objet contrôlée par script est créée. Si vous founissez seulement ce paramètre, l'image-objet est placée au centre de la scène.

emplacement Facultatif. Point qui spécifie l'emplacement de la scène où est placée l'image-objet contrôlée par script.

Exemple

L'instruction suivante crée une image-objet scriptée dans la piste des images-objets 5 à partir de l'acteur intitulé cerf-volant et la place à un point spécifique de la scène :

```
-- Syntaxe Lingo
channel(5).makeScriptedSprite(member("cerf-volant"), point(35, 70))

// Syntaxe JavaScript
channel(5).makeScriptedSprite(member("cerf-volant"), point(35, 70));
```

Voir aussi

removeScriptedSprite(), Piste d'image-objet

makeSubList()

Utilisation

nœudXML.makeSubList()

Description

Fonction ; renvoie une liste de propriétés d'un nœud enfant de la même façon que makeList() renvoie la racine d'un document XML sous la forme d'une liste.

Paramètres

Aucun.

Exemple

Avec le code XML suivant :

```
<?xml version="1.0"?>
  <e1>
    <nomDeBalise attr1="val1" attr2="val2"/>
    <e2>élém 2</e2>
    <e3>élém 3</e3>

    </e1>
```

L'instruction suivante renvoie une liste de propriétés constituée du contenu du premier enfant de la balise <e1> :

```
put gObjetDanalyse.child[ 1 ].child[ 1 ].makeSubList()
-- ["nomDeBalise": ["!ATTRIBUTES": ["attr1": "val1", "attr2": "val2"]]]
```

Voir aussi

makeList()

map()

Utilisation

```
map(rectCible, rectSource, rectDeDestination)
map(pointCible, rectSource, rectDeDestination)
```

Description

Fonction ; permet de positionner et de dimensionner un rectangle ou un point en fonction du rapport existant entre un rectangle source et un rectangle cible.

La relation de la valeur *pointCible* avec la valeur *rectSource* contrôle la relation du résultat de la fonction avec *rectDeDestination*.

Paramètres

rectCible Requis. Rectangle cible dans la relation.

pointCible Requis. Point cible dans la relation.

rectSource Requis. Rectangle source dans la relation.

rectDeDestination Requis. Rectangle de destination.

Exemple

Dans le comportement suivant, toutes les images-objets ont déjà été définies comme déplaçables. L'image-objet 2b contient un petit bitmap. L'image-objet 1s est une image-objet de forme rectangulaire, suffisamment grande pour contenir facilement l'image-objet 2b. L'image-objet 4b est une version plus grande du bitmap compris dans l'image-objet 2b. L'image-objet 3s est une version plus grande de la forme comprise dans l'image-objet 1s. Le déplacement des images-objets 2b ou 1s entraîne le déplacement de l'image-objet 4b. Lorsque vous faites glisser l'image-objet 2b, ses mouvements sont copiés par l'image-objet 4b. Lorsque vous faites glisser l'image-objet 1s, l'image-objet 4b se déplace dans la direction opposée. Le redimensionnement de l'image-objet 2b ou 1s produira des résultats intéressants.

```
on exitFrame
  sprite(4b).rect = map(sprite(2b).rect, sprite(1s).rect, sprite(3s).rect)
  go the frame
end
```

map (3D)

Utilisation

```
member(quelActeur).motion(quelMouvement).\
  map(quelAutreMouvement {, nomDeSegment})
```

Description

Commande de mouvement 3D ; mappe un mouvement spécifié sur le mouvement courant et l'applique à un segment et à tous les enfants de ce dernier. Cette commande remplace tout mouvement précédemment mappé dans le segment spécifié et ses enfants. Cette commande ne modifie pas la liste de lecture d'un modèle.

Paramètres

quelAutreMouvement Requis. Chaîne qui spécifie le nom du mouvement à mapper.

nomDeSegment Facultatif. Chaîne qui spécifie le nom du segment auquel le mouvement mappé est appliqué. Si ce paramètre est omis, le segment racine est utilisé.

Exemple

L'instruction suivante mappe le mouvement Plafond au mouvement Assis, à partir du segment Cou. Le modèle s'assiéra et regardera le plafond simultanément.

```
member("Restaurant").motion("Assis").map("Plafond", "Cou")
```

Voir aussi

motion(), duration (3D), cloneMotionFromCastmember

mapMemberToStage()

Utilisation

```
sprite(quelNuméroDimageObjet).mapMemberToStage(quelPointDeLacteur)
mapMemberToStage(sprite quelNuméroDimageObjet, quelPointDeLacteur)
```

Description

Fonction ; utilise l'image-objet et le point spécifiés pour renvoyer un point équivalent dans les dimensions de la scène. Cette propriété tient compte de la transformation actuelle de l'image-objet en utilisant `quad` ou le rectangle si celui-ci n'est pas transformé.

Elle est idéale pour déterminer si l'utilisateur a cliqué sur une zone spécifique d'un acteur, même si son image-objet sur la scène a été transformée de manière importante.

Si le point de la scène spécifié ne se trouve pas à l'intérieur de l'image-objet, la valeur `VOID` est renvoyée.

Paramètres

quelPointDeLacteur Requis. Point à partir duquel un point équivalent est renvoyé.

Voir aussi

map(), mapStageToMember()

mapStageToMember()

Utilisation

```
sprite(quelNuméroDimageObjet).mapStageToMember(quelPointDeLaScène)
mapStageToMember(sprite quelNuméroDimageObjet, quelPointDeLaScène)
```

Description

Fonction ; utilise l'image-objet et le point spécifiés pour renvoyer un point équivalent dans les dimensions de l'acteur. Cette propriété tient compte de la transformation actuelle de l'image-objet en utilisant `quad` ou le rectangle si celui-ci n'est pas transformé.

Elle est idéale pour déterminer si l'utilisateur a cliqué sur une zone spécifique d'un acteur, même si son image-objet sur la scène a été transformée de manière importante.

Si le point de la scène spécifié ne se trouve pas à l'intérieur de l'image-objet, cette fonction renvoie la valeur `VOID`.

Paramètres

`quelPointDeLaScène` Requis. Point à partir duquel un point équivalent est renvoyé.

Voir aussi

`map()`, `mapMemberToStage()`

marker()

Utilisation

```
-- Syntaxe Lingo
_movie.marker(nomOuNumDeRepère)

// Syntaxe JavaScript
_movie.marker(nomOuNumDeRepère);
```

Description

Méthode d'animation ; renvoie le numéro des repères situés avant et après l'image courante.

Cette méthode est utile pour implémenter un bouton Suivant ou Précédent ou pour définir une boucle d'animation.

Si le paramètre *`nomOuNumDeRepère`* est un nombre entier, il peut être positif, négatif ou égal à zéro. Par exemple :

- `marker(2)` – Renvoie le numéro d'image du deuxième repère après l'image courante.
- `marker(1)` – Renvoie le numéro d'image du premier repère après l'image courante.
- `marker(0)` – Renvoie le numéro de l'image courante si celle-ci contient un repère ou, si elle n'en contient pas, celui de l'image contenant le repère précédent.
- `marker(-1)` – Renvoie le numéro d'image du premier repère précédant le repère (0).
- `marker(-2)` – Renvoie le numéro d'image du second repère précédant le repère (0).

Si le paramètre *`nomOuNumDeRepère`* est une chaîne, `marker()` renvoie le numéro de la première image dont le repère correspond à la chaîne.

Paramètres

`nomOuNumDeRepère` Requis. Chaîne qui spécifie un libellé de repère ou nombre entier qui spécifie un numéro de repère.

Exemple

L'instruction suivante fait passer la tête de lecture au début de l'image courante si celle-ci contient un repère ou la fait passer au repère précédent dans le cas contraire :

```
-- Syntaxe Lingo
_movie.go(_movie.marker(0))

// Syntaxe JavaScript
_movie.go(_movie.marker(0));
```

L'instruction suivante donne à la variable `repèreSuivant` la même valeur que celle du repère suivant du scénario :

```
-- Syntaxe Lingo
repèreSuivant = _movie.marker(1)

// Syntaxe JavaScript
repèreSuivant = _movie.marker(1);
```

Voir aussi

`frame`, `frameLabel`, `go()`, `label()`, `markerList`, `Animation`

max()

Utilisation

```
liste.max()
max(liste)
max(valeur1, valeur2, valeur3, ...)
```

Description

Fonction (Lingo uniquement) ; renvoie la valeur la plus élevée dans la liste spécifiée ou dans une série de valeurs donnée.

La fonction `max` peut également être utilisée avec les caractères ASCII, de la même manière que les opérateurs < et > peuvent être utilisés avec des chaînes.

Paramètres

`valeur1, valeur2, valeur3, ...` Facultatif. Liste de valeurs dans laquelle la valeur la plus élevée est sélectionnée.

Exemple

Le gestionnaire suivant affecte à la variable `Gagnant` la valeur maximum de la liste `Devis`, qui est composée de [#Martin:600, #Dupont:750, #Lajoie:230]. Le résultat est ensuite inséré dans le contenu de l'acteur champ Félicitations.

```
-- Syntaxe Lingo
on rechercheDuGagnant Devis
  Gagnant = Devis.max()
  member("Félicitations").text = \
  "Vous avez gagné, avec un devis de " & Gagnant &"euros !"
end

// Syntaxe JavaScript
function rechercheDuGagnant(Devis) {
  Gagnant = Devis.max();
  member("Félicitations").text = "Vous avez gagné, avec un devis de " +
  Gagnant \
  + "euros !");
}
```

maximize()

Utilisation

```
-- Syntaxe Lingo
réfObjFenêtre.maximize()

// Syntaxe JavaScript
réfObjFenêtre.maximize();
```

Description

Méthode de fenêtre ; agrandit une fenêtre.

Utilisez cette méthode lorsque vous créez des barres de titre personnalisées.

Paramètres

Aucun.

Exemple

Ces instructions agrandissent la fenêtre intitulée Artistes si elle n'est pas déjà agrandie.

```
-- Syntaxe Lingo
if (window("Artistes").sizeState <> #maximized) then
  window("Artistes").maximize()
end if

// Syntaxe JavaScript
if (window("Artistes").sizeState != symbol("maximized")) {
  window("Artistes").maximize();
}
```

Voir aussi

minimize(), Fenêtre

mci

Utilisation

```
mci "chaîne"
```

Description

Commande ; exclusive à l'environnement Windows, passe les chaînes spécifiées par *chaîne* à l'interface de contrôle des médias de Windows pour le contrôle des extensions multimédia.

Remarque : Microsoft ne recommande plus l'utilisation de l'interface MCI de 16 bits. Vous devrez peut-être utiliser des Xtras de fabricants tiers pour remplacer cette fonctionnalité.

Paramètres

chaîne Requis. Chaîne qui est transmise à l'interface MCI.

Exemple

L'instruction suivante ordonne à la commande `play cdaudio from 200 to 600 track 7` de ne lire le CD audio que lorsque l'animation est lue sous Windows :

```
mci "play cdaudio from 200 to 600 track 7"
```

member()

Utilisation

```
-- Syntaxe Lingo
member(nomOuNumDacteur {, nomOuNumDeDistribution})

// Syntaxe JavaScript
member(nomOuNumDacteur {, nomOuNumDeDistribution});
```

Description

Fonction du niveau supérieur ; crée une référence à un acteur et, éventuellement, spécifie la distribution qui contient cet acteur.

Lorsqu'elle est utilisée à la fois avec le paramètre *nomOuNumDacteur* et le paramètre *nomOuNumDeDistribution*, la méthode `member()` est une référence spécifique à une distribution et à un de ses acteurs :

```
trace(sprite(1).member);
// (member 1 of castLib 1)
```

Cette méthode est différente de la propriété d'image-objet `spriteNum`, qui est toujours un nombre entier désignant une position dans une distribution, mais qui ne spécifie pas la distribution :

```
trace(sprite(2).spriteNum);
// 2
```

Le numéro d'un acteur est également une référence absolue à un acteur spécifique d'une distribution particulière :

```
trace(sprite(3).member.number)
// 3
```

Paramètres

nomOuNumDacteur Requis. Chaîne qui spécifie le nom de l'acteur à référencer ou nombre entier qui spécifie sa position d'index.

nomOuNumDeDistribution Facultatif. Chaîne qui spécifie le nom de la distribution à laquelle l'acteur appartient ou nombre entier qui spécifie sa position d'index. Si ce paramètre est omis, `member()` effectue une recherche dans toutes les distributions jusqu'à ce qu'il trouve une correspondance.

Exemple

Cette instruction affecte à la variable `ailesMem` l'acteur Avions, qui se trouve dans la distribution Transport.

```
-- Syntaxe Lingo
ailesMem = member("Avions", "Transport")

// Syntaxe JavaScript
var ailesMem = member("Avions", "Transport");
```

Voir aussi

Acteur, Image-objet, spriteNum

mergeDisplayTemplate()

Utilisation

```
-- Syntaxe Lingo
_movie.mergeDisplayTemplate(listeDesPropriétés)

// Syntaxe JavaScript
_movie.mergeDisplayTemplate(listeDesPropriétés);
```

Description

Méthode d'animation ; fusionne un nombre arbitraire de propriétés de gabarit d'affichage avec le jeu de propriétés existant en une seule opération.

Paramètres

listeDesPropriétés Requis. Liste de propriétés qui contient les propriétés de gabarit d'affichage à fusionner avec le jeu de propriétés existant. En Lingo, *listeDesPropriétés* est une liste de paires nom/valeur ou symbole/valeur séparées par des virgules. Dans la syntaxe JavaScript, *listeDesPropriétés* est une liste de paires nom/valeur séparées par des virgules.

Exemple

Cette instruction fusionne une valeur pour la propriété `title` dans `displayTemplate` :

```
-- Syntaxe Lingo
_movie.mergeDisplayTemplate(propList(#title, "Bienvenue !"))

// Syntaxe JavaScript
_movie.mergeDisplayTemplate(propList("title", "Bienvenue !"))
```

Voir aussi

appearanceOptions, displayTemplate, Animation, propList(), titlebarOptions

mergeProps()

Utilisation

```
-- Syntaxe Lingo
réfObjFenêtre.mergeProps(listeDesPropriétés)

// Syntaxe JavaScript
réfObjFenêtre.mergeProps(listeDesPropriétés);
```

Description

Méthode de fenêtre. Fusionne un nombre arbitraire de propriétés de fenêtre avec le jeu de propriétés existant en une seule opération.

Paramètres

listeDesPropriétés Requis. Jeu de propriétés de fenêtre à fusionner avec le jeu de propriétés existant. Les propriétés sont spécifiées par les propriétés `appearanceOptions` et `titlebarOptions`.

- En Lingo, *listeDesPropriétés* est une liste de paires nom/valeur ou symbole/valeur séparées par des virgules.
- Dans la syntaxe JavaScript, *listeDesPropriétés* est une liste de paires nom/valeur séparées par des virgules.

Exemple

Cette instruction définit diverses propriétés pour la fenêtre intitulée Voitures.

```
-- Syntaxe Lingo
window("Voitures").mergeProps([#title:"Photos de voitures", #resizable:FALSE,
   \
   #titlebarOptions:[#closebox:TRUE, #icon:member(2)], \
   #appearanceOptions:[#border:#line, #shadow:TRUE]])

// Syntaxe JavaScript
window("Voitures").mergeProps(propList("title","Photos de voitures",
   "resizable",false,
   "titlebarOptions",propList("closebox",true, "icon",member(2)),
   "appearanceOptions",propList("border","line", "shadow",true)));
```

Voir aussi

appearanceOptions, titlebarOptions, Fenêtre

mesh (propriété)

Utilisation

```
member(quelActeur).model(quelModèle).\
   meshdeform.mesh[index].propriétéDeMaille
```

Description

Commande 3D ; permet d'accéder aux propriétés de maille des modèles auxquels est associé le modificateur `meshDeform`. Lorsque utilisée comme `mesh.count`, cette commande renvoie le nombre total de mailles du modèle référencé.

Les propriétés de chacune des mailles auxquelles il est possible d'accéder sont les suivantes :

- `colorList` permet d'obtenir ou de définir la liste des couleurs utilisées par la maille spécifiée.
- `vertexList` permet d'obtenir ou de définir la liste des sommets utilisés par la maille spécifiée.
- `normalList` vous permet d'obtenir ou de définir la liste des vecteurs de normales utilisés par la maille spécifiée.
- `textureCoordinateList` permet d'obtenir ou de définir les coordonnées de texture utilisées par la première couche de texture de la maille spécifiée. Pour obtenir ou définir les coordonnées de texture pour toute autre couche de texture de la maille spécifiée, utilisez `meshdeform.mesh[index].texturelayer[index].textureCoordinateList`.
- `textureLayer[index]` permet d'obtenir ou de définir l'accès aux propriétés de la couche de texture spécifiée.
- `face[index]` permet d'obtenir ou de définir les sommets, les normales, les coordonnées de texture, les couleurs et les matériaux utilisés par les faces de la maille spécifiée.
- `face.count` permet d'obtenir le nombre total des faces qui se trouvent à l'intérieur de la maille spécifiée.

Remarque : Pour plus d'informations sur ces propriétés, consultez les entrées correspondantes (et qui apparaissent sous la section Voir aussi de cette entrée).

Paramètres

Aucun.

Exemple

Le code Lingo suivant ajoute le modificateur #meshDeform au modèle truc1, puis affiche la liste des sommets de la première maille du modèle truc1.

```
member("nouveauMartien").model("truc1").addModifier(#meshDeform)
put member("nouveauMartien").model("truc1").meshDeform.mesh[1].vertexList
-- [vector(239.0, -1000.5, 27.4), vector\
  (162.5, -1064.7, 29.3), vector(115.3, -1010.8, -40.6),
vector(239,0, -1000,5, 27,4), vector(115,3, -1010,8, -40,6),
vector(162.5, -1064.7, 29.3), vector(359.0, -828.5, -46.3),
vector(309.9, -914.5, -45.3)]
```

L'instruction suivante affiche le nombre de mailles du modèle Avion.

```
put member("world").model("Avion").meshDeform.mesh.count
-- 4
```

Voir aussi

meshDeform (modificateur), colorList, textureCoordinateList, textureLayer, normalList, vertexList (déformation de maille), face[ ]

meshDeform (modificateur)

Utilisation

member(*quelActeur*).model(*quelModèle*).meshDeform.*nomDePropriété*

Description

Modificateur 3D ; permet de contrôler les différents aspects de la structure de maille du modèle référencé. Lorsque vous ajoutez le modificateur #meshDeform (à l'aide de la commande addModifier) à un modèle, vous avez accès aux propriétés suivantes du modificateur #meshDeform :

Remarque : Pour plus d'informations sur ces propriétés, consultez les entrées correspondantes (et qui apparaissent sous la section Voir aussi de cette entrée).

- face.count renvoie le nombre total de faces du modèle référencé.
- mesh.count renvoie le nombre de mailles du modèle référencé.
- mesh[index] permet d'accéder aux propriétés de la maille spécifiée.

Paramètres

Aucun.

Exemple

L'instruction suivante affiche le nombre de faces du modèle gbFace.

```
put member("Univers 3D").model("gbFace").meshDeform.face.count
-- 432
```

L'instruction suivante affiche le nombre de mailles du modèle gbFace.

```
put member("Univers 3D").model("gbFace").meshDeform.mesh.count
-- 2
```

L'instruction suivante affiche le nombre de faces de la deuxième maille du modèle gbFace.

```
put member("Univers 3D").model("gbFace").meshDeform.mesh[2].face.count
-- 204
```

Voir aussi

mesh (propriété), addModifier

min

Utilisation

```
liste.min
min(liste)
min (a1, a2, a3...)
```

Description

Fonction (Lingo uniquement) ; spécifie la valeur minimale dans une liste.

Paramètres

a1, a2, a3, ... Facultatif. Liste de valeurs dans laquelle la valeur la plus faible est sélectionnée.

Exemple

Le gestionnaire suivant donne à la variable `vPlusBas` la valeur la plus faible de la liste Devis, qui est composée de [#Pierre:600, #Paul:750, #Jean:230]. Le résultat est alors inséré dans l'acteur champ Désolé :

```
on rechercheDuPlusBas Devis
  vPlusBas = Devis.min()
  member("Désolé").text = \
  "Votre offre de " & vPlusBas && " euros n'a pas été acceptée !"
end
```

Voir aussi

max()

minimize()

Utilisation

```
-- Syntaxe Lingo
réfObjFenêtre.minimize()

// Syntaxe JavaScript
réfObjFenêtre.minimize();
```

Description

Méthode de fenêtre ; réduit une fenêtre.

Utilisez cette méthode lorsque vous créez des barres de titre personnalisées.

Paramètres

Aucun.

Exemple

Ces instructions réduisent la fenêtre intitulée Artistes si elle n'est pas déjà réduite.

```
-- Syntaxe Lingo
if (window("Artistes").sizeState <> #minimized) then
  window("Artistes").minimize()
end if

// Syntaxe JavaScript
if (window("Artistes").sizeState != symbol("minimized")) {
  window("Artistes").minimized();
}
```

Voir aussi

maximize(), Fenêtre

model

Utilisation

```
member(quelActeur).model(quelModèle)
member(quelActeur).model[index]
member(quelActeur).model.count
member(quelActeur).model(quelModèle).nomDePropriété
member(quelActeur).model[index].nomDePropriété
```

Description

Commande 3D ; renvoie le modèle trouvé dans l'acteur référencé dont le nom est spécifié par *quelModèle*, ou trouvé à la position d'index spécifiée par *index*. Si aucun modèle n'existe pour le paramètre spécifié, la commande renvoie `void`. Tout comme `model.count`, la commande renvoie le nombre de modèles détectés dans l'acteur référencé. Cette commande permet également d'accéder aux propriétés du modèle spécifié.

Les comparaisons de noms de modèle ne sont pas sensibles à la hauteur de casse. La position d'index d'un modèle particulier peut changer lorsque des objets dans des positions inférieures sont supprimés.

Si aucun modèle n'utilise le nom spécifié ou n'est trouvé à la position d'index spécifiée, cette commande renvoie `void`.

Paramètres

quelModèle Facultatif. Chaîne qui spécifie le nom du modèle à renvoyer.

Exemple

L'instruction suivante enregistre une référence au modèle Lecteur dans la variable ceModèle :

```
ceModèle = member("Univers3D").model("Lecteur")
```

L'instruction suivante enregistre une référence au huitième modèle de l'acteur Univers3D dans la variable ceModèle.

```
ceModèle = member("Univers3D").model[8]
```

L'instruction suivante indique que quatre modèles se trouvent dans l'acteur de l'image-objet 1.

```
put sprite(1).member.model.count
-- 4
```

modelResource

Utilisation

```
member(quelActeur).modelResource(quelleRessDeMod)
member(quelActeur).modelResource[index]
member(quelActeur).modelResource.count
member(quelActeur).modelResource(quelleRessDeMod).\
  nomDePropriété
member(quelActeur).modelResource[index].nomDePropriété
```

Description

Commande 3D ; renvoie la ressource de modèle trouvée dans l'acteur référencé dont le nom est spécifié par *quelleRessDeMod*, ou trouvée à la position d'index spécifiée par le paramètre *index*. Si aucune ressource de modèle n'existe pour le paramètre spécifié, la commande renvoie `void`. Tout comme `modelResource.count`, la commande renvoie le nombre de ressources de modèle détectées dans l'acteur référencé. Cette commande permet également d'accéder aux propriétés du modèle spécifié.

Les comparaisons de chaînes de noms de ressources de modèle ne sont pas sensibles à la hauteur de casse. La position d'index d'une ressource de modèle particulière peut changer lorsque des objets dans des positions inférieures sont supprimés.

Paramètres

quelleRessDeMod Facultatif. Chaîne qui spécifie le nom de la ressource de modèle à renvoyer.

Exemple

L'instruction suivante enregistre une référence à la ressource de modèle MaisonA dans la variable `cetteRessourceDeModèle`.

```
cetteRessourceDeModèle = member("Univers3D").modelResource("MaisonA")
```

L'instruction suivante enregistre une référence à la quatorzième ressource de modèle de l'acteur Univers3D dans la variable `cetteRessourceDeModèle`.

```
cetteRessourceDeModèle = member("Univers3D").modelResource[14]
```

L'instruction suivante indique que dix ressources de modèle se trouvent dans l'acteur de l'image-objet 1.

```
put sprite(1).member.modelResource.count
--10
```

modelsUnderLoc

Utilisation

```
member(quelActeur).camera(quelleCaméra).modelsUnderLoc\
  (pointDansLimageObjet {, nombreMaxDeModèles, précision})
```

Description

Commande 3D ; renvoie une liste des modèles situés sous un point spécifié à l'intérieur du rectangle d'une image-objet utilisant la caméra référencée.

Dans la liste renvoyée, le premier modèle listé est celui qui est le plus proche du spectateur et le dernier modèle listé est le plus éloigné.

Une seule intersection (la plus proche) est renvoyée par modèle.

La commande renvoie une liste vide s'il n'existe aucun modèle sous le point spécifié.

Paramètres

pointDansLimageObjet Requis. Point sous lequel est située une liste de modèles. Ce point est défini par rapport au coin supérieur gauche de l'image-objet, en pixels.

nombreMaxDeModèles Facultatif. Nombre entier qui spécifie la longueur maximale de la liste renvoyée. Si ce paramètre n'est pas spécifié, la commande renvoie une liste contenant les références de tous les modèles détectés sous le point spécifié.

précision Facultatif. Symbole qui spécifie le niveau de détail des informations renvoyées. Les valeurs correctes sont :

- `#simple` renvoie une liste contenant les références des modèles situés sous le point. C'est la valeur par défaut.
- `#detailed` renvoie une liste de listes de propriétés, représentant chacune un modèle rencontré. Chaque liste de propriétés doit contenir les propriétés suivantes :
  - `#model` est une référence à l'objet de modèle rencontré.
  - `#distance` est la distance séparant la caméra du point d'intersection avec le modèle.
  - `#isectPosition` est un vecteur représentant la position du point d'intersection dans l'univers.
  - `#isectNormal` est le vecteur de la maille au point d'intersection.
  - `#meshID` est l'identifiant de la maille coupée, qui peut être utilisé comme index dans la liste des mailles du modificateur `meshDeform`.
  - `#faceID` est l'identifiant de la face coupée, qui peut être utilisé comme index dans la liste des faces du modificateur `meshDeform`.
  - `#vertices` est une liste de vecteurs contenant trois éléments qui représentent les positions dans l'univers des sommets de la face intersectée.
  - `#uvCoord` est une liste de propriétés contenant les propriétés `#u` et `#v` représentant les coordonnées barycentriques u et v de la face.

Exemple

La première ligne du gestionnaire suivant transfère l'emplacement du curseur d'un point de la scène à un point de l'image-objet 5. La deuxième ligne utilise la commande `modelsUnderLoc` pour obtenir les trois premiers modèles situés sous ce point. La troisième ligne affiche des informations détaillées sur les modèles dans la fenêtre Messages.

```
-- Syntaxe Lingo
on mouseUp
  pt = the mouseLoc - point(sprite(5).left, sprite(5).top)
  m = sprite(5).camera.modelsUnderLoc(pt, 3, #detailed)
  put m
end
```

```
// Syntaxe JavaScript
function mouseUp() {
  pt = _mouse.mouseLoc - point(sprite(5).left, sprite(5).top);
m = sprite(5).camera.modelsUnderLoc(pt, 3, #detailed)
  put(m);
}
```

Voir aussi

modelsUnderRay, modelUnderLoc

modelsUnderRay

Utilisation

```
member(quelActeur).modelsUnderRay(vecteurDemplacement, vecteurDeDirection {,
  nombreMaxDeModèles, précision})
```

Description

Commande 3D ; renvoie une liste des modèles situés sous un rayon tracé à partir d'une position spécifiée dans une direction donnée ; les deux vecteurs étant spécifiés en coordonnées relatives à l'univers.

Dans la liste renvoyée, le premier modèle listé est le plus proche de la position spécifiée par *vecteurDemplacement* et le dernier modèle listé est le plus éloigné de cette position.

Une seule intersection (la plus proche) est renvoyée par modèle.

La commande renvoie une liste vide s'il n'existe aucun modèle sous le rayon spécifié.

Paramètres

vecteurDemplacement Requis. Vecteur à partir duquel un rayon est dessiné et sous lequel se trouve une liste de modèles.

vecteurDeDirection Requis. Vecteur qui spécifie la direction dans laquelle le rayon est orienté.

nombreMaxDeModèles Facultatif. Nombre entier qui spécifie la longueur maximale de la liste renvoyée. Si ce paramètre n'est pas spécifié, la commande renvoie une liste contenant les références de tous les modèles détectés sous le rayon spécifié.

précision Facultatif. Symbole qui spécifie le niveau de détail des informations renvoyées. Les valeurs correctes sont :

- #simple renvoie une liste contenant les références des modèles situés sous le point. C'est la valeur par défaut.
- #detailed renvoie une liste de listes de propriétés, représentant chacune un modèle rencontré. Chaque liste de propriétés doit contenir les propriétés suivantes :
  - #model est une référence à l'objet de modèle rencontré.
  - #distance est la distance séparant la position d'univers spécifiée par *vecteurDemplacement* du point d'intersection avec le modèle.
  - #isectPosition est un vecteur représentant la position du point d'intersection dans l'univers.
  - #isectNormal est le vecteur de la maille au point d'intersection.
  - #meshID est l'identifiant de la maille coupée, qui peut être utilisé comme index dans la liste des mailles du modificateur meshDeform.

- `#faceID` est l'identifiant de la face coupée, qui peut être utilisé comme index dans la liste des faces du modificateur `meshDeform`.
- `#vertices` est une liste de vecteurs contenant trois éléments qui représentent les positions dans l'univers des sommets de la face intersectée.
- `#uvCoord` est une liste de propriétés contenant les propriétés `#u` et `#v` représentant les coordonnées barycentriques u et v de la face.

Exemple

L'instruction suivante affiche les informations détaillées d'un modèle intersecté par un rayon tracé à partir de la position de vector(0, 0, 300), en direction de l'axe des z négatif.

```
put member("3d").modelsUnderRay(vector(0, 0, 300), vector(0, 0, -\
  1), 3, #detailed)
-- [[#model: model("mSphere"), #distance: 275.0000, \
  #isectPosition: vector( 0.0000, 0.0000, 25.0000 ), #isectNormal: \
  vector( -0.0775, 0.0161, 0.9969 ), #meshID: 1, #faceID: 229, \
  #vertices: [vector( 0.0000, 0.0000, 25.0000 ), vector( -3.6851, \
  1.3097, 24.6922 ), vector( -3.9017, 0.2669, 24.6922 )], \
  #uvCoord: [#u: 0.0000, #v: 0.0000]]]
```

Voir aussi

`modelsUnderLoc`, `modelUnderLoc`

modelUnderLoc

Utilisation

```
member(quelActeur).camera(quelleCaméra).\
  modelUnderLoc(pointDansLimageObjet)
```

Description

Commande 3D ; renvoie une référence au premier modèle situé sous un point spécifié à l'intérieur du rectangle d'une image-objet utilisant la caméra référencée.

Cette commande renvoie `void` si aucun modèle n'est situé sous le point spécifié.

Pour obtenir une liste de tous les modèles situés sous un point spécifié et des informations détaillées les concernant, consultez `modelsUnderLoc`.

Paramètres

`pointDansLimageObjet` Requis. Point sous lequel se trouve le premier modèle. L'emplacement de *`pointDansLimageObjet`* est calculé en fonction du coin supérieur gauche de l'image-objet, en pixels.

Exemple

La première ligne du gestionnaire suivant transfère l'emplacement du curseur d'un point de la scène à un point de l'image-objet 5. La deuxième ligne détermine le premier modèle situé sous ce point. La troisième ligne affiche les résultats dans la fenêtre Messages.

```
-- Syntaxe Lingo
on mouseUp
  pt = the mouseLoc - point(sprite(5).left, sprite(5).top)
  m = sprite(5).camera.modelUnderLoc(pt)
  put m
end
```

```
// Syntaxe JavaScript
function mouseUp() {
  pt = _mouse.mouseLoc - point(sprite(5).left, sprite(5).top);
  m = sprite(5).camera.modelUnderLoc(pt);
  put(m);
}
```

Voir aussi

modelsUnderLoc, modelsUnderRay

motion()

Utilisation

```
member(quelActeur).motion(quelMouvement)
member(quelActeur).motion[index]
member(quelActeur).motion.count
```

Description

Commande 3D ; renvoie le mouvement trouvé dans l'acteur référencé dont le nom est spécifié par *quelMouvement*, ou trouvé à la position d'index spécifiée par *index*. Tout comme `motion.count`, cette propriété renvoie le nombre total de mouvements détectés dans l'acteur.

Les comparaisons de chaînes de nom d'objet ne sont pas sensibles à la hauteur de casse. La position d'index d'un mouvement particulier peut changer lorsque des objets dans des positions inférieures sont supprimés.

Si aucun mouvement n'utilise le nom spécifié ou n'est trouvé à la position d'index spécifiée, cette commande renvoie `void`.

Exemple

```
ceMouvement = member("Univers 3D").motion("Aile")
ceMouvement = member("Univers 3D").motion[7]
put member("séquence").motion.count
-- 2
```

Voir aussi

duration (3D), map (3D)

move()

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.move({entPosn, nomDeDistribution})

// Syntaxe JavaScript
réfObjActeur.move({entPosn, nomDeDistribution});
```

Description

Méthode d'acteur ; déplace un acteur spécifié vers le premier emplacement vide de la distribution dans laquelle il est inclus ou vers un emplacement spécifié dans une distribution donnée.

Pour obtenir les meilleurs résultats, utilisez cette méthode pendant la programmation, et non pendant l'exécution d'une animation, car le déplacement est en général enregistré avec le fichier. En général, l'emplacement réel de l'acteur n'affecte pas la plupart des présentations quand l'utilisateur final les lit. Pour changer le contenu de l'image-objet ou modifier son affichage pendant la lecture de l'animation, utilisez la propriété `member` de l'image-objet.

Paramètres

`entPosn` Facultatif. Nombre entier qui spécifie la position vers laquelle l'acteur est déplacé dans la distribution *`nomDeDistribution`*.

`nomDeDistribution` Facultatif. Chaîne qui spécifie le nom de la distribution vers laquelle l'acteur est déplacé.

Exemple

L'instruction suivante déplace l'acteur Temple vers le premier emplacement vide de la fenêtre Distribution :

```
-- Syntaxe Lingo
member("Temple").move()

// Syntaxe JavaScript
member("Temple").move();
```

L'instruction suivante déplace l'acteur Temple vers l'emplacement 20 de la fenêtre Distribution Bitmaps :

```
-- Syntaxe Lingo
member("Temple").move(20, "Bitmaps")

// Syntaxe JavaScript
member("Temple").move(20, "Bitmaps");
```

Voir aussi

Acteur

moveToBack()

Utilisation

```
-- Syntaxe Lingo
réfObjFenêtre.moveToBack()

// Syntaxe JavaScript
réfObjFenêtre.moveToBack();
```

Description

Méthode de fenêtre ; déplace une fenêtre derrière toutes les autres fenêtres.

Paramètres

Aucun.

Exemple

Les instructions suivantes placent la première fenêtre de la liste `windowList` derrière toutes les autres fenêtres :

```
-- Syntaxe Lingo
maFenêtre = _player.windowList[1]
maFenêtre.moveToBack()

// Syntaxe JavaScript
var maFenêtre = _player.windowList[1];
maFenêtre.moveToBack();
```

Si vous connaissez le nom de la fenêtre à déplacer, utilisez la syntaxe suivante :

```
-- Syntaxe Lingo
window("Fenêtre Démo").moveToBack()

// Syntaxe JavaScript
window("Fenêtre Démo").moveToBack();
```

Voir aussi

`moveToFront()`, `Fenêtre`

moveToFront()

Utilisation

```
-- Syntaxe Lingo
réfObjFenêtre.moveToFront()

// Syntaxe JavaScript
réfObjFenêtre.moveToFront();
```

Description

Méthode de fenêtre ; déplace une fenêtre devant toutes les autres fenêtres.

Paramètres

Aucun.

Exemple

Les instructions suivantes placent la première fenêtre de la liste `windowList` devant toutes les autres fenêtres :

```
-- Syntaxe Lingo
maFenêtre = _player.windowList[1]
maFenêtre.moveToFront()

// Syntaxe JavaScript
var maFenêtre = _player.windowList[1];
maFenêtre.moveToFront();
```

Si vous connaissez le nom de la fenêtre à déplacer, utilisez la syntaxe suivante :

```
-- Syntaxe Lingo
window("Fenêtre Démo").moveToFront()

// Syntaxe JavaScript
window("Fenêtre Démo").moveToFront();
```

Voir aussi

moveToBack(), Fenêtre

moveVertex()

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.moveVertex(indexDeSommet, changementX, changementY)

// Syntaxe JavaScript
réfObjActeur.moveVertex(indexDeSommet, changementX, changementY);
```

Description

Fonction ; place le sommet d'un acteur forme vectorielle à un autre emplacement.

Les coordonnées horizontales et verticales du déplacement sont calculées en fonction de la position courante du point du sommet. L'emplacement de celui-ci dépend de l'origine de l'acteur forme vectorielle.

La modification de l'emplacement du sommet affecte la forme de la même manière que si vous faites glisser le sommet dans l'éditeur.

Paramètres

indexDeSommet Requis. Spécifie la position d'index du sommet à déplacer.

changementX Requis. Spécifie la distance sur laquelle le sommet doit être déplacé horizontalement.

changementY Requis. Spécifie la distance sur laquelle le sommet doit être déplacé verticalement.

Exemple

L'instruction suivante déplace le premier sommet de l'acteur forme vectorielle Archie de 25 pixels vers la droite et de 10 pixels vers le bas par rapport à sa position actuelle :

```
-- Syntaxe Lingo
member("Archie").moveVertex(1, 25, 10)

// Syntaxe JavaScript
member("Archie").moveVertex(1, 25, 10);
```

Voir aussi

addVertex(), deleteVertex(), moveVertexHandle(), originMode, vertexList

moveVertexHandle()

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.moveVertexHandle(indexDeSommet, indexDePoignée, changementX,
  changementY)

// Syntaxe JavaScript
réfObjActeur.moveVertexHandle(indexDeSommet, indexDePoignée, changementX,
  changementY);
```

Description

Fonction ; déplace la poignée du sommet d'un acteur de forme vectorielle vers un autre emplacement.

Les coordonnées horizontales et verticales du déplacement sont calculées par rapport à la position courante de la poignée du sommet. L'emplacement de la poignée du sommet dépend du sommet qu'elle contrôle.

La modification de l'emplacement de la poignée de contrôle affecte la forme de la même manière que si vous faites glisser le sommet dans un éditeur.

Paramètres

indexDeSommet Requis. Spécifie la position d'index du sommet qui contient la poignée à déplacer.

indexDePoignée Requis. Spécifie la position d'index de la poignée à déplacer.

changementX Requis. Spécifie la distance sur laquelle vous voulez déplacer la poignée du sommet horizontalement.

changementY Requis. Spécifie la distance sur laquelle vous voulez déplacer la poignée du sommet verticalement.

Exemple

L'instruction suivante déplace la première poignée de contrôle du deuxième sommet de l'acteur forme vectorielle Archie de 15 pixels vers la droite et de 5 pixels vers le haut :

```
-- Syntaxe Lingo
moveVertexHandle(member("Archie"), 2, 1, 15, -5)

// Syntaxe JavaScript
moveVertexHandle(member("Archie"), 2, 1, 15, -5)
```

Voir aussi

addVertex(), deleteVertex(), originMode, vertexList

multiply()

Utilisation

```
transformation.multiply(transformation2)
```

Description

Commande 3D ; applique les effets de position, de rotation et de redimensionnement de *transformation2* après la transformation d'origine.

Paramètres

transformation2 Requis. Spécifie la transformation qui contient les effets à appliquer à une autre transformation.

Exemple

L'instruction suivante applique les effets de position, rotation et redimensionnement de la transformation du modèle Mars à la transformation du modèle Pluton. Cela a le même effet que de définir Mars comme le parent de Pluton pour une image.

```
member("Scène").model("Pluton").transform.multiply(member("Scène")\
   .model("Mars").transform)
```

neighbor

Utilisation

```
member(quelActeur).model(quelModèle).meshdeform.mesh[index].\
   face[index].neighbor[index]
```

Description

Commande 3D ; commande `meshDeform` qui renvoie une liste de listes décrivant les voisins d'une face particulière d'une maille à l'opposé du coin de face spécifié par l'index de voisinage `(1,2,3)`. Si la liste est vide, la face n'a aucun voisin dans cette direction. Si la liste contient plus d'une liste, la maille est non-manifold. La liste contient généralement une liste unique de quatre valeurs entières : `[indexDeMaille, indexDeFace, indexDeSommet, inverse?]`.

La valeur `indexDeMaille` est l'index de la maille contenant la face voisine. La valeur `indexDeFace` est l'index de la face voisine dans la maille. La valeur `indexDeSommet` est l'index des sommets non partagés de la face voisine. La valeur `inverse?` indique si l'orientation de la face est la même `(1)` ou l'inverse `(2)` de la face d'origine.

Paramètres

Aucun.

Voir aussi

`meshDeform (modificateur)`

netAbort

Utilisation

```
netAbort(URL)
netAbort(IDréseau)
```

Description

Commande ; annule une opération réseau sans attendre de résultat.

L'utilisation d'un identifiant de réseau constitue la manière la plus efficace d'annuler une opération réseau. Cet identifiant est renvoyé lorsque vous utilisez une fonction réseau telle que `getNetText()` ou `postNetText()`.

Dans certains cas, si l'identifiant de réseau n'est pas disponible, vous pouvez utiliser une URL pour annuler la transmission de données à partir de cette URL. Celle-ci doit être identique à l'URL utilisée au départ de l'opération réseau. Si le transfert des données est terminé, cette commande n'a aucun effet.

Paramètres

URL Requis. Spécifie l'URL à annuler.

IDréseau Facultatif. Spécifie l'ID de l'opération réseau à annuler.

Exemple

L'instruction suivante passe un identifiant de réseau à `netAbort` pour annuler une opération réseau particulière :

```
-- Syntaxe Lingo
on mouseUp
  netAbort(monIDréseau)
end

// Syntaxe JavaScript
function mouseUp() {
  netAbort(monIDréseau);
}
```

Voir aussi

`getNetText()`, `postNetText`

netDone()

Utilisation

```
netDone()
netDone(IDréseau)
```

Description

Fonction ; indique si une opération de chargement en tâche de fond (telle que `getNetText`, `preloadNetThing`, `gotoNetMovie`, `gotoNetPage` ou `netTextResult`) est terminée ou a été interrompue par suite d'une erreur du navigateur (`TRUE`, valeur par défaut) ou si elle est encore en cours (`FALSE`).

- Utilisez `netDone()` pour tester la dernière opération réseau.
- Utilisez `netDone(IDréseau)` pour tester l'opération réseau identifiée par `IDréseau`.

La fonction `netDone` renvoie 0 lorsqu'une opération de chargement en tâche de fond est en cours.

Paramètres

IDréseau Facultatif. Spécifie l'ID de l'opération réseau à tester.

Exemple

Le gestionnaire suivant utilise la fonction `netDone` pour vérifier si la dernière opération réseau est terminée. Si l'opération est terminée, le texte renvoyé par `netTextResult` est affiché dans l'acteur Affichage du texte.

```
-- Syntaxe Lingo
on exitFrame
  if netDone() = 1 then
    member("Affichage du texte").text = netTextResult()
  end if
end

// Syntaxe JavaScript
function exitFrame() {
  if (netDone() == 1) {
    member("Affichage du texte").text = netTextResult();
  }
}
```

Le gestionnaire suivant utilise un identifiant de réseau spécifique comme argument pour que `netDone` vérifie l'état d'une opération réseau particulière :

```
-- Syntaxe Lingo
on exitFrame
  -- rester sur cette image jusqu'à la fin de
  -- l'opération réseau
  global monIDréseau
  if netDone(monIDréseau) = FALSE then
    go to the frame
  end if
end

// Syntaxe JavaScript
function exitFrame() {
  // rester sur cette image jusqu'à la fin de l'opération réseau
  global monIDréseau;
  if (!(netDone(monIDréseau))) {
    _movie.go(_movie.frame);
  }
}
```

Voir aussi

`getNetText()`, `netTextResult()`, `gotoNetMovie`, `preloadNetThing()`

netError()

Utilisation

```
netError()
netError(IDréseau)
```

Description

Fonction ; détermine si une erreur s'est produite pendant une opération réseau, et, le cas échéant, renvoie un numéro d'erreur correspondant à un message d'erreur. Si l'opération s'est déroulée sans erreur, cette fonction renvoie un code indiquant que tout va bien. Si aucune opération de chargement en tâche de fond n'a commencé, ou si l'opération est en cours, cette fonction renvoie une chaîne vide.

- Utilisez `netError()` pour tester la dernière opération réseau.
- Utilisez `netError(`*`IDréseau`*`)` pour tester l'opération réseau spécifiée par `IDréseau`.

Plusieurs codes d'erreur peuvent être renvoyés :

0	Tout va bien.
4	Classe MOA incorrecte. Les Xtras requis, réseau ou non, ne sont pas installés correctement ou ne sont pas installés du tout.
5	Interface MOA incorrecte. Voir 4.
6	URL incorrecte ou classe MOA incorrecte. Les Xtras requis, réseau ou non, ne sont pas installés correctement ou ne sont pas installés du tout.
20	Erreur interne. Renvoyé par `netError()` dans le navigateur Netscape si celui-ci a détecté une erreur de réseau ou une erreur interne.
4146	La connexion n'a pas pu être établie avec l'hôte distant.
4149	Les données fournies par le serveur ont un format inattendu.
4150	Fermeture prématurée et inattendue de la connexion.
4154	L'opération n'a pas pu aboutir par suite du dépassement du temps imparti.
4155	Mémoire insuffisante pour terminer la transaction.
4156	La réponse du protocole à la requête indique une erreur de réponse.
4157	La transaction n'a pas pu être authentifiée.
4159	URL non valide.
4164	Impossible de créer de socket.
4165	L'objet demandé est introuvable (l'URL est peut-être incorrecte).
4166	Echec de proxy générique.
4167	Le transfert a été interrompu volontairement par le client.
4242	Téléchargement annulé par `netAbort(url)`.
4836	Téléchargement annulé ou interrompu pour une raison inconnue, probablement une erreur du réseau.

Paramètres

`IDréseau` Facultatif. Spécifie l'ID de l'opération réseau à tester.

Exemple

L'instruction suivante passe un identifiant de réseau à `netError` pour vérifier l'état d'erreur d'une opération réseau spécifique :

```
-- Syntaxe Lingo
on exitFrame
  global monIDréseau
  if netError(monIDréseau)<>"OK" then beep
end
```

```
// Syntaxe JavaScript
function exitFrame() {
  global monIDréseau;
  if (netError(monIDréseau) != "OK") {
    _sound.beep();
  }
}
```

netLastModDate()

Utilisation

`netLastModDate()`

Description

Fonction ; renvoie la date de la dernière modification de l'en-tête HTTP de l'élément spécifié. Cette chaîne utilise le format de temps universel (GMT) : *Jjj, nn Mmm aaaa hh:mm:ss* GMT (par exemple, Thu, 30 Jan 1997 12:00:00 AM GMT). Les jours et les mois peuvent être écrits en entier. Cette chaîne est toujours en anglais.

La fonction `netLastModDate` ne peut être appelée qu'une fois que `netDone` et `netError` rapportent que l'opération s'est terminée avec succès. Dès que l'opération suivante démarre, l'animation ou la projection Director efface les résultats de l'opération précédente pour économiser de la mémoire.

La chaîne de date est extraite de l'en-tête HTTP dans le format offert par le serveur. Cependant, cette chaîne n'est pas toujours fournie et, le cas échéant, `netLastModDate` renvoie `EMPTY`.

Paramètres

Aucun.

Exemple

Les instructions suivantes vérifient la date d'un fichier téléchargé depuis Internet :

```
-- Syntaxe Lingo
if netDone() then
  laDate = netLastModDate()
  if laDate.char[6..11] <> "30 Jan" then
    alert "Ce fichier est périmé."
  end if
end if

// Syntaxe JavaScript
if (netDone()) {
  laDate = netLastModDate();
  if (laDate.char[6..11] != "30 Jan") {
    alert("Ce fichier est périmé");
  }
}
```

Voir aussi

`netDone()`, `netError()`

netMIME()

Utilisation

`netMIME()`

Description

Fonction ; fournit le type MIME du fichier Internet renvoyé par la dernière opération réseau (le fichier HTTP ou FTP téléchargé en dernier).

La fonction `netMIME` ne peut être appelée qu'une fois que `netDone` et `netError` rapportent que l'opération s'est terminée avec succès. Dès que l'opération suivante démarre, l'animation ou la projection Director efface les résultats de l'opération précédente pour économiser de la mémoire.

Paramètres

Aucun.

Exemple

Le gestionnaire suivant vérifie le type MIME d'un fichier téléchargé depuis Internet et répond en conséquence :

```
-- Syntaxe Lingo
on vérifierLopérationRéseau uneURL
  if netDone (uneURL) then
    set monTypeMime = netMIME()
    case monTypeMime of
      "image/jpeg": go frame "info jpeg"
      "image/gif": go frame "info gif"
      "application/x-director": goToNetMovie uneURL
      "text/html": goToNetPage uneURL
      otherwise: alert "Veuillez choisir un autre fichier."
    end case
  else
    go the frame
  end if
end
```

```
// Syntaxe JavaScript
function vérifierLopérationRéseau(uneURL) {
  if (netDone (uneURL)) {
    monTypeMime = netMIME();
    switch (monTypeMime) {
      case "image/jpeg":
        _movie.go("Lions");
        break;
      case "image/gif":
        _movie.go("info gif");
        break;
      case "application/x-director":
        goToNetMovie(uneURL);
        break;
      case "text/html":
        goToNetPage(uneURL);
        break;
      default:
        alert("Veuillez choisir un autre fichier.");

    }
  } else {
    _movie.go(_movie.frame);
  }
}
```

Voir aussi

`netDone()`, `netError()`, `getNetText()`, `postNetText`, `preloadNetThing()`

netStatus

Utilisation

`netStatus` *`chaîneDeMessage`*

Description

Commande ; affiche la chaîne spécifiée dans la zone d'état de la fenêtre du navigateur.

La commande `netStatus` ne fonctionne pas dans les projections.

Paramètres

`chaîneDeMessage` Requis. Spécifie la chaîne à afficher.

Exemple

L'instruction suivante place la chaîne Test dans la zone d'état du navigateur web dans lequel l'animation est lue :

```
-- Syntaxe Lingo
on exitFrame
  netStatus "Test"
end

// Syntaxe JavaScript
function exitFrame() {
  _movie.netStatus("Test");
}
```

netTextResult()

Utilisation

```
netTextResult(IDréseau)
netTextResult()
```

Description

Fonction ; renvoie le texte obtenu par l'opération réseau spécifiée. Si aucun identifiant réseau n'est spécifié, `netTextResult` renvoie le résultat de la dernière opération réseau.

Si l'opération réseau spécifiée était `getNetText()`, le texte renvoyé correspond à celui du fichier sur le réseau.

Si l'opération réseau spécifiée était `postNetText`, le résultat correspond à la réponse du serveur.

Dès que l'opération suivante démarre, Director efface les résultats de l'opération précédente pour économiser de la mémoire.

Lorsqu'une animation est lue sous forme d'applet, cette fonction renvoie les résultats valides des dernières 10 demandes. Lorsqu'une animation est lue sous forme d'animation avec un contenu Shockwave, cette fonction ne renvoie que les résultats valides de l'opération `getNetText()` la plus récente.

Paramètres

IDréseau Facultatif. Spécifie l'ID de l'opération réseau qui contient le texte à renvoyer.

Exemple

Le gestionnaire suivant utilise les fonctions `netDone` et `netError` pour tester si la dernière opération réseau s'est terminée avec succès. Si l'opération est terminée, le texte renvoyé par `netTextResult` est affiché dans l'acteur Affichage du texte.

```
-- Syntaxe Lingo
global gIDréseau

on exitFrame
  if (netDone(gIDréseau) = TRUE) and (netError(gIDréseau) = "OK") then
    member("Affichage du texte").text = netTextResult()
  end if
end

// Syntaxe JavaScript
global gIDréseau;

function exitFrame() {
  if (netDone(gIDréseau) && (netError(gIDréseau) == "OK")) {
    member("Affichage du texte").text = netTextResult();
  }
}
```

Voir aussi

`netDone()`, `netError()`, `postNetText`

new()

Utilisation

```
new(type)
new(type, castLib quelleDistribution)
new(type, member quelActeur of castLib quelleDistribution)
nomDeVariable = new(scriptParent arg1, arg2, ...)
new(script nomDuScriptParent, valeur1, valeur2, ...)
timeout("nom").new(période, #gestionnaire, {, objetCible})
new(xtra "nomDeLxtra")
```

Description

Fonction ; crée un nouvel acteur, objet enfant, objet de temporisation ou instance d'Xtra, et permet d'affecter des valeurs de propriétés individuelles aux objets enfants.

Pour les acteurs, le paramètre *type* sert à définir le type de l'acteur. Les valeurs prédéfinies offertes correspondent aux différents types d'acteurs : #bitmap, #field, etc. La fonction new permet également de créer des types d'acteurs Xtra, auxquels vous pouvez attribuer n'importe quel nom.

Vous pouvez également créer un nouvel acteur curseur de couleur en utilisant l'Xtra Custom Cursor. Utilisez new(#cursor), puis définissez les propriétés de l'acteur obtenu de manière à pouvoir les utiliser.

Les paramètres facultatifs *quelActeur* et *quelleDistribution* spécifient la position de l'acteur et la fenêtre Distribution dans laquelle il est créé. Si vous ne spécifiez pas de position, l'acteur est placé dans la première position vide de cette distribution. La fonction new renvoie la position de l'acteur.

Lorsque l'argument de la fonction new est un script parent, la fonction new crée un objet enfant. Le script parent doit contenir un gestionnaire on new définissant l'état initial de l'objet enfant ou la valeur de ses propriétés ; il renvoie la référence me de l'objet enfant.

L'objet enfant possède tous les gestionnaires du script parent. Il utilise les mêmes noms de variables de propriétés que celles qui sont déclarées dans le script parent, mais peut toutefois avoir ses propres valeurs pour ces propriétés.

Puisque l'objet enfant est une valeur, il peut être affecté à des variables, placé dans des listes ou être passé comme paramètre.

Comme avec toute autre variable, vous pouvez utiliser la commande put pour afficher des informations sur un objet enfant dans la fenêtre Messages.

Lorsque new() est utilisé pour créer un objet de temporisation, la période définit le nombre de millisecondes séparant les événements de temporisation envoyés par l'objet de temporisation. La valeur *#gestionnaire* est un symbole identifiant le gestionnaire qui sera appelé lors de chaque événement de temporisation. La valeur *objetCible* identifie le nom de l'objet enfant contenant la valeur *#gestionnaire*. Si aucune valeur *objetCible* n'est définie, la valeur *#gestionnaire* est considérée comme étant dans un script d'animation.

Lorsqu'un objet de temporisation est créé, il active son *objetCible* pour la réception des événements système prepareMovie, startMovie, stopMovie, prepareFrame et exitFrame. Pour en profiter, l'*objetCible* doit contenir des gestionnaires pour ces événements. Les événements ne doivent pas obligatoirement survenir dans l'ordre pour que le reste de l'animation puisse y accéder.

Pour un exemple d'utilisation de `new()` dans une animation, reportez-vous aux animations Parent Scripts et Read and Write Text du dossier Learning/Lingo, lui-même inclus dans le dossier de Director.

Exemple

Pour créer un nouvel acteur bitmap dans le premier emplacement disponible de la distribution, utilisez la syntaxe suivante :

```
set nouvelActeur = new(#bitmap)
```

Une fois cette ligne exécutée, `nouvelActeur` contient la référence de l'acteur que vous venez de créer :

```
put nouvelActeur
-- (member 1 of castLib 1)
```

Le script `startMovie` suivant crée un nouvel acteur Flash au moyen de la commande `new`, définit la propriété `linked` de l'acteur nouvellement créé de sorte que ses éléments soient stockés dans un fichier externe, puis définit la propriété `pathName` de l'acteur comme emplacement d'une animation Flash sur le web :

```
on startMovie
  acteurFlash = new(#flash)
  member(acteurFlash).pathName = "http://www.uneURL.fr/monFlash.swf"
end
```

Lorsque l'animation démarre, le gestionnaire suivant crée un nouvel acteur curseur animé en couleur et stocke son numéro d'acteur dans la variable `curseurPersonnalisé`. Cette variable sert à définir la propriété `castMemberList` de l'acteur qui vient d'être créé et à afficher le nouveau curseur.

```
on startmovie
  curseurPersonnalisé = new(#cursor)
  member(curseurPersonnalisé).castMemberList = [member 1, member 2, member 3]
  cursor (member(curseurPersonnalisé))
end
```

Les instructions du script parent suivantes contiennent un gestionnaire `on new` servant à créer un objet enfant. Le script parent est un acteur script appelé `Oiseau` qui contient ces gestionnaires.

```
on new me, nomDeLoiseau
  return me
end

on voler me
  put "Je vole"
end
```

Dans l'exemple suivant, la première instruction crée un objet enfant à partir du script de l'exemple précédent et le place dans la variable `monOiseau`. La seconde instruction fait voler l'oiseau en appelant le gestionnaire `voler` du script parent `Oiseau` :

```
monOiseau = script("Oiseau").new()
monOiseau.voler()
```

L'instruction suivante crée un autre script parent `Oiseau` contenant la variable de propriété `vitesse` :

```
property vitesse

on new me, vitesseInitiale
  vitesse = vitesseInitiale
  return me
end
on voler me
  put "Je vole à " & vitesse & "kmh"
end
```

Les instructions suivantes créent deux objets enfants appelés `monOiseau1` et `monOiseau2`. Les vitesses suivantes leur sont attribuées : 15 et 25, respectivement. Lorsque le gestionnaire voler est appelé pour chaque objet enfant, la vitesse de ce dernier s'affiche dans la fenêtre Messages.

```
monOiseau1 = script("Oiseau").new(15)
monOiseau2 = script("Oiseau").new(25)
monOiseau1.voler()
monOiseau2.voler()
```

Le message suivant apparaît dans la fenêtre Messages :

```
-- "Je vole à 15 kmh"
-- "Je vole à 25 kmh"
```

L'instruction suivante crée un objet de temporisation nommé `compteurDintervalle` qui envoie un événement de temporisation au gestionnaire `on bipChaqueMinute` de l'objet enfant `lecteur1` toutes les 60 secondes :

```
timeout("compteurDintervalle").new(60000, #bipChaqueMinute, lecteur1)
```

Voir aussi

`on stepFrame`, `actorList`, `ancestor`, `me`, `type (acteur)`, `timeout()`

newCamera

Utilisation

`member(`*`quelActeur`*`).newCamera(`*`nomDeNouvelleCaméra`*`)`

Description

Commande 3D ; crée une caméra dans un acteur.

Paramètres

`nomDeNouvelleCaméra` Requis. Spécifie le nom de la nouvelle caméra. Le nom spécifié de la nouvelle caméra doit être unique dans l'acteur.

Exemple

L'instruction suivante crée une nouvelle caméra appelée Caméra_embarquée.

```
member("Univers 3D").newCamera("Caméra_embarquée")
```

newCurve()

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.newCurve(positionDansLaListeDesSommets)

// Syntaxe JavaScript
réfObjActeur.newCurve(positionDansLaListeDesSommets);
```

Description

Fonction ; ajoute un symbole #newCurve à la liste vertexList de *réfObjActeur*, qui ajoute une nouvelle forme à la forme vectorielle. Vous pouvez fractionner une forme existante en appelant newCurve() positionné au milieu d'une série de sommets.

Paramètres

positionDansLaListeDesSommets Requis. Indique à quelle position le symbole #newCurve est ajouté à la vertexList.

Exemple

Les instructions suivantes ajoutent une nouvelle courbe à l'acteur 2 à la troisième position de la liste des sommets de l'acteur. La seconde ligne de l'exemple remplace le contenu de la courbe 2 par le contenu de la courbe 3.

```
-- Syntaxe Lingo
member(2).newCurve(3)
member(2).curve[2]=member(2).curve[3]

// Syntaxe JavaScript
member(2).newCurve(3);
member(2).curve[2] = member(2).curve[3];
```

Voir aussi

curve, vertexList

newGroup

Utilisation

```
member(quelActeur).newGroup(nomDeNouveauGroupe)
```

Description

Commande 3D ; crée un groupe et l'ajoute à la palette des groupes.

Paramètres

nomDeNouveauGroupe Requis. Spécifie le nom du nouveau groupe. Le nom du nouveau groupe doit être unique dans la palette des groupes.

Exemple

L'instruction suivante crée le groupe gbGroupe2 dans l'acteur Scène et une référence à ce groupe dans la variable ng.

```
ng = member("Scène").newGroup("gbGroupe2")
```

newLight

Utilisation

```
member(quelActeur).newLight(nomDeNouvelleLumière, #indicateurDeType)
```

Description

Commande 3D ; crée une lumière avec un type spécifié et l'ajoute à la palette des lumières.

Paramètres

nomDeNouvelleLumière Requis. Spécifie le nom de la nouvelle lumière. Le nom de la nouvelle lumière doit être unique dans la palette des lumières.

indicateurDeType Requis. Symbole qui spécifie le type de la nouvelle lumière. Les valeurs correctes sont :

- `#ambient` est une lumière généralisée dans l'univers 3D.
- `#directional` est une lumière émise dans une direction spécifique.
- `#point` est une source lumineuse similaire à une ampoule.
- `#spot` est un effet de projecteur.

Exemple

L'instruction suivante crée une lumière dans l'acteur Univers 3D. Il s'agit d'une lumière d'ambiance appelée « lumière ambiante ».

```
member("Univers 3D").newLight("lumière ambiante", #ambient)
```

newMember()

Utilisation

```
-- Syntaxe Lingo
_movie.newMember(symbole)
_movie.newMember(chaîneTypeDacteur)

// Syntaxe JavaScript
_movie.newMember(chaîneTypeDacteur);
```

Description

Méthode d'animation ; crée un acteur et vous permet d'affecter des valeurs de propriétés individuelles aux objets enfants.

Pour les nouveaux acteurs, le paramètre *symbole* ou *chaîneTypeDacteur* définit le type de l'acteur. Les valeurs prédéfinies offertes correspondent aux différents types d'acteurs : `#bitmap`, `#field`, etc. La méthode `newMember()` crée également des types d'acteurs Xtra, auxquels vous pouvez attribuer n'importe quels noms.

Vous pouvez également créer un nouvel acteur curseur de couleur en utilisant l'Xtra Custom Cursor. Utilisez `newMember(#cursor)`, puis définissez les propriétés de l'acteur obtenu de manière à pouvoir les utiliser.

Une fois que la méthode `newMember()` a été appelée, le nouveau membre est placé dans la première miniature libre de la distribution.

Lorsque l'argument de la fonction `new()` est un script parent, la fonction `new` crée un objet enfant. Le script parent doit contenir un gestionnaire `on new` définissant l'état initial de l'objet enfant ou la valeur de ses propriétés ; il renvoie la référence `me` de l'objet enfant.

L'objet enfant possède tous les gestionnaires du script parent. Il utilise les mêmes noms de variables de propriétés que celles qui sont déclarées dans le script parent, mais peut toutefois avoir ses propres valeurs pour ces propriétés.

Puisque l'objet enfant est une valeur, il peut être affecté à des variables, placé dans des listes ou être passé comme paramètre.

Comme avec toute autre variable, vous pouvez utiliser la méthode `put()` pour afficher des informations sur un objet enfant dans la fenêtre Messages.

Lorsque `new()` est utilisé pour créer un objet de temporisation, la période définit le nombre de millisecondes séparant les événements de temporisation envoyés par l'objet de temporisation. La valeur *`#gestionnaire`* est un symbole identifiant le gestionnaire qui sera appelé lors de chaque événement de temporisation. La valeur *`objetCible`* identifie le nom de l'objet enfant contenant la valeur *`#gestionnaire`*. Si aucune valeur *`objetCible`* n'est définie, la valeur *`#gestionnaire`* est considérée comme étant dans un script d'animation.

Lorsqu'un objet de temporisation est créé, il active son *`objetCible`* pour la réception des événements système `prepareMovie`, `startMovie`, `stopMovie`, `prepareFrame` et `exitFrame`. Pour en profiter, l'*`objetCible`* doit contenir des gestionnaires pour ces événements. Les événements ne doivent pas obligatoirement survenir dans l'ordre pour que le reste de l'animation puisse y accéder.

Pour un exemple d'utilisation de `newMember()` dans une animation, reportez-vous aux animations Parent Scripts et Read and Write Text du dossier Learning/Lingo, lui-même inclus dans le dossier de Director.

Paramètres

`symbole` (Lingo uniquement) Requis. Symbole qui spécifie le type du nouvel acteur.

`chaîneTypeDacteur` Requis. Chaîne qui spécifie le type du nouvel acteur.

Exemple

Les instructions suivantes créent un nouvel acteur bitmap et l'affecte à la variable `nouveauBitmap`.

```
-- Syntaxe Lingo
nouveauBitmap = _movie.newMember(#bitmap) - avec un symbole
nouveauBitmap = _movie.newMember("bitmap") - avec une chaîne

// Syntaxe JavaScript
var nouveauBitmap = _movie.newMember("bitmap");
```

Voir aussi

`Animation`, `type (acteur)`

newMesh

Utilisation

```
member(quelActeur).newMesh(nomDeMaille, nombreDeFaces, nombreDeSommets,
nombreDeNormales,nombreDeCouleurs,nombreDeCoordonnéesDeTexture)
```

Description

Commande 3D ; crée une ressource de modèle de maille. Après avoir créé une maille, vous devez au moins définir les valeurs des propriétés `vertexList` et `face[index].vertices` de la nouvelle maille, puis appeler sa commande `build()` pour générer la géométrie.

Paramètres

`nomDeMaille` Requis. Spécifie le nom de la nouvelle ressource de modèle de maille.

`nombreDeFaces` Requis. Spécifie le nombre total de triangles devant apparaître dans la maille.

`nombreDeSommets` Requis. Spécifie le nombre total de sommets utilisés par toutes les faces (triangulaires). Un sommet peut être partagé par plus d'une face.

`nombreDeNormales` Facultatif. Spécifie le nombre total de normales. Une normale peut être partagée par plus d'une face. La normale d'un coin de triangle définit la direction extérieure au triangle, ce qui affecte l'éclairage de cet angle. Entrez 0 ou ignorez ce paramètre si vous prévoyez d'utiliser la commande `generateNormals()` de la maille pour générer les normales.

`nombreDeCouleurs` Facultatif. Spécifie le nombre total de couleurs utilisées par toutes les faces. Une couleur peut être partagée par plus d'une face. Vous pouvez spécifier une couleur pour chaque angle de chaque face. Spécifiez les couleurs pour obtenir un dégradé de couleurs progressif. Entrez 0 ou ignorez ce paramètre pour obtenir la couleur blanche par défaut pour chaque coin de face.

`nombreDeCoordonnéesDeTexture` Facultatif. Spécifie le nombre total de coordonnées de texture spécifiées par l'utilisateur utilisées par toutes les faces. Entrez 0 ou ignorez ce paramètre pour obtenir les coordonnées de texture par défaut générées par un placage planaire. Pour plus d'informations, consultez la description de `#planar` dans l'entrée `shader.textureWrapMode`. Spécifiez les coordonnées de texture lorsque vous avez besoin de contrôler précisément le placage des textures sur les faces de la maille.

Exemple

L'exemple suivant crée une ressource de modèle de type `#mesh`, en spécifie les propriétés, puis l'utilise pour créer un nouveau modèle. Le processus est décrit dans les explications accompagnant l'exemple suivant :

La **ligne 1** crée une maille contenant 6 faces, composées de 5 sommets et 3 couleurs uniques. Le nombre de normales et le nombre de coordonnées de textures ne sont pas définis. Les normales seront créées par la commande `generateNormals`.

La **ligne 2** définit les cinq sommets uniques utilisés par les faces de la maille.

La **ligne 3** définit les trois couleurs uniques utilisées par les faces de la maille.

Les **lignes 4 à 9** désignent les sommets à utiliser pour les coins de chaque face de la pyramide. Veuillez noter que l'ordre des sommets suit le sens des aiguilles d'une montre. `GenerateNormals()` est basé sur un ordre suivant le sens des aiguilles d'une montre.

Les **lignes 10 à 15** affectent des couleurs aux coins de chaque face. Les couleurs seront étalées sur les faces en dégradés.

La **ligne 16** crée les normales de Triangle en appelant la commande `generateNormals()`.

La **ligne 17** appelle la commande `build` pour construire la maille.

```
nm = member("Formes").newMesh("pyramide",6 , 5, 0, 3)
nm.vertexList = [ vector(0,0,0), vector(40,0,0), \
  vector(40,0,40), vector(0,0,40), vector(20,50,20) ]
nm.colorList = [ rgb(255,0,0), rgb(0,255,0), rgb(0,0,255) ]
nm.face[1].vertices = [ 4.10.2 ]
nm.face[2].vertices = [ 4.2,3 ]
nm.face[3].vertices = [ 5.20.1 ]
nm.face[4].vertices = [ 5.30.2 ]
nm.face[5].vertices = [ 5.4,3 ]
nm.face[6].vertices = [ 5.10.4 ]
nm.face[1].colors = [3,200.3]
nm.face[2].colors = [3.30.2]
nm.face[3].colors = [1.30.2]
nm.face[4].colors = [1.20.3]
nm.face[5].colors = [1.30.2]
nm.face[6].colors = [1.20.3]
nm.generateNormals(#flat)
nm.build()
nm = member("Formes").newModel("Pyramide1", nm)
```

Voir aussi

`newModelResource`

newModel

Utilisation

```
member( quelActeur ).newModel( nomDeNouveauModèle \
  {, quelleRessDeMod } )
```

Description

Commande 3D ; crée un modèle dans l'acteur référencé. La propriété `resource` de tous les nouveaux modèles est définie sur `VOID` par défaut.

Paramètres

`nomDeNouveauModèle` Requis. Spécifie le nom du nouveau modèle. Le nom du nouveau modèle doit être unique.

`quelleRessDeMod` Facultatif. Spécifie la ressource de modèle à utiliser pour créer le modèle.

Exemple

L'instruction suivante crée le modèle Nouvelle maison dans l'acteur Univers 3D.

```
member("Univers 3D").newModel("Nouvelle maison")
```

La ressource de modèle pour le nouveau modèle peut également être définie à l'aide du paramètre facultatif *`quelleRessDeMod`*.

```
member("Univers 3D").newModel("Nouvelle maison", member("Univers \
  3D").modelResource("grandeBoîte"))
```

newModelResource

Utilisation

```
member(quelActeur).newModelResource(nomDeNouvelleRessourceDeModèle\
  { ,#type, #faisantFaceA })
```

Description

Commande 3D ; crée une ressource de modèle, éventuellement, avec des paramètres type et faisantFaceA donnés, et l'ajoute à la palette de ressources de modèle.

Si vous ne spécifiez pas le paramètre *faisantFaceA*, mais que vous spécifiez #box, #sphere, #particle ou #cylinder comme paramètre *type*, seules les faces avant seront générées. Si vous spécifiez #plane, les faces avant et arrière sont créées. Les ressources de modèle de type #plane ont deux mailles générées (une pour chaque côté), et par conséquent, deux matériaux dans shaderList.

Une valeur de face #both crée le double de mailles et aussi le double d'entrées de matériaux dans shaderList. Il y en aura 2 pour les plans et les sphères (respectivement, pour l'intérieur et l'extérieur du modèle), 12 pour les cubes (6 à l'extérieur, 6 à l'intérieur) et 6 pour les cylindres (le sommet, la base et les contours extérieur et intérieur).

Paramètres

nomDeNouvelleRessourceDeModèle Requis. Spécifie le nom de la nouvelle ressource de modèle.

type Facultatif. Spécifie le type de primitive de la nouvelle ressource de modèle. Les valeurs correctes sont :

- #plane
- #box
- #sphere
- #cylinder
- #particle

faisantFaceA Facultatif. Spécifie la face de la nouvelle ressource de modèle. Les valeurs correctes sont :

- #front
- #back
- #both

Exemple

Le gestionnaire suivant crée une boîte. La première ligne du gestionnaire crée une ressource de modèle nommée boîte10. Elle est de type #box et est définie pour que seule sa face arrière soit présentée. Les trois lignes suivantes définissent les dimensions de boîte10 et la dernière ligne crée un nouveau modèle qui utilise boîte10 comme ressource de modèle.

```
on créationDeBoîte
  nmr = member("3D").newModelResource("boîte10", #box, #back)
  nmr.height = 50
  nmr.width = 50
  nmr.length = 50
  aa = member("3D").newModel("gb5", nmr)
end
```

L'instruction suivante crée une ressource de modèle en forme de boîte, appelée « boîteAchapeaux4 ».

```
member("Etagère").newModelResource("boîteAchapeaux4", #box)
```

Voir aussi

primitives

newMotion()

Utilisation

```
member(quelActeur).newMotion(nom)
```

Description

Commande 3D ; crée un mouvement dans un acteur référencé et renvoie une référence au nouveau mouvement. Un nouveau mouvement peut être utilisé pour combiner plusieurs mouvements existants dans la liste des mouvements de l'acteur, au moyen de la commande `map()`.

Paramètres

nom Requis. Spécifie le nom du nouveau mouvement. Le nom du nouveau mouvement doit être unique dans l'acteur référencé.

Exemple

L'instruction suivante crée un mouvement dans l'acteur 1, `courseEtOndulation`, qui est utilisé pour combiner les mouvements de course et d'ondulation provenant de la liste des mouvements de l'acteur :

```
courseEtOndulation = member(1).newMotion("courseEtOndulation")
courseEtOndulation.map("course", "bassin")
courseEtOndulation.map("ondulation", "épaule")
```

newObject()

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.newObject(typeDobjet {, arg1, arg2 ....})

// Syntaxe JavaScript
réfObjImageObjet.newObject(typeDobjet {, arg1, arg2 ....});
```

Description

Commande d'image-objet Flash ; crée un objet ActionScript du type spécifié.

La syntaxe suivante crée un objet dans une image-objet Flash :

```
réfDimageObjetFlash.newObject("typeDobjet" {, arg1, arg2 ....})
```

La syntaxe suivante crée un objet global :

```
newObject("typeDobjet" {, arg1, arg2 ....})
```

Remarque : Si vous n'avez pas importé d'acteur Flash, vous devrez ajouter manuellement l'Xtra Flash Asset à la liste des Xtras de votre animation pour permettre aux commandes Flash globales de fonctionner correctement dans Shockwave Player et dans les projections. Vous pouvez ajouter les Xtras à la liste des Xtras en choisissant Modification › Animation › Xtras. Pour plus d'informations sur la gestion des Xtras pour les animations distribuées, consultez les rubriques Utilisation de Director dans le panneau d'aide de Director.

Paramètres

typeDobjet Requis. Spécifie le type de l'objet à créer.

arg1, arg2, ... Facultatif. Spécifie les arguments d'initialisation nécessaires à l'objet. Chaque argument doit être séparé des autres par une virgule.

Exemple

Cette instruction définit la variable `tObjetDeConnexionLocale` sur une référence à un nouvel objet `LocalConnection` dans l'animation Flash, au niveau de l'image-objet 3 :

```
-- Syntaxe Lingo
tObjetDeConnexionLocale = sprite(3).newObject("LocalConnection")

// Syntaxe JavaScript
var tObjetDeConnexionLocale = sprite(3).newObject("LocalConnection");
```

L'instruction suivante définit la variable `tObjetTableau` sur une référence à un nouvel objet Tableau dans l'animation Flash, au niveau de l'image-objet 3. Le tableau contient les 3 nombres entiers : 23, 34 et 19.

```
-- Syntaxe Lingo
tObjetTableau = sprite(3).newObject("Tableau",23,34,19)

// Syntaxe JavaScript
var tObjetTableau = sprite(3).newObject("Tableau",23,34,19);
```

Voir aussi

`setCallback()`, `clearAsObjects()`

newShader

Utilisation

`member(quelActeur).newShader(nomDeNouveauMatériau, #typeDeMatériau)`

Description

Commande 3D ; crée un matériau du type spécifié dans la liste des matériaux de l'acteur référencé et renvoie une référence au nouveau matériau.

Chaque type de matériau possède un groupe de propriétés qui lui sont spécifiques, en plus des propriétés de matériau `#standard`. Toutefois, même si vous pouvez attribuer n'importe quelle propriété de matériau `#standard` à un matériau d'un autre type, il est possible que la propriété n'ait aucun effet visible. C'est ce qui se produit lorsque la propriété #standard est appliquée, car elle remplace la nature du type de matériau. Par exemple, la propriété de matériau standard `diffuseLightMap` est ignorée par les matériaux du type `#engraver`, `#newsprint` et `#painter`.

Paramètres

nomDeNouveauMatériau Requis. Spécifie le nom du nouveau matériau. Le nom du nouveau matériau doit être unique dans la liste des matériaux.

typeDeMatériau Requis. Symbole qui détermine le style dans lequel le matériau est appliqué. Les valeurs correctes sont :

- Les matériaux `#standard` sont photoréalistes et ont les propriétés suivantes : `ambient`, `blend`, `blendConstant`, `blendConstantList`, `blendFunction`, `blendFunctionList`, `blendSource`, `blendSourceList`, `diffuse`, `diffuseLightMap`, `emissive`, `flat`, `glossMap`, `ilk`, `name`, `region`, `renderStyle`, `silhouettes`, `specular`, `specularLightMap`, `texture`, `textureMode`, `textureModeList`, `textureRepeat`, `textureRepeatList`, `textureTransform`, `textureTransformList`, `transparent`, `useDiffuseWithTexture`, `wrapTransform` et `wrapTransformList`.
- Les matériaux `#painter` sont estompés, donnent l'apparence d'une peinture et ont, en plus de toutes les propriétés `#standard`, les propriétés suivantes : `colorSteps`, `hilightPercentage`, `hilightStrength`, `name`, `shadowPercentage`, `shadowStrength` et `style`.
- Les matériaux `#engraver` sont striés, donnent l'apparence d'une gravure et ont, en plus de toutes les propriétés #standard, les propriétés suivantes : `brightness`, `density`, `name` et `rotation`.
- Les matériaux `#newsprint` sont en pointillés, donnent l'apparence d'une reproduction de journal et ont, en plus de toutes les propriétés #standard, les propriétés suivantes : `brightness`, `density` et `name`.

Exemple

L'instruction suivante crée le matériau `#painter` nouveauPeintre.

```
nouveauPeintre = member("Univers 3D").newShader("nouveauPeintre",#painter)
```

Voir aussi

`shadowPercentage`

newTexture

Utilisation

```
member(quelActeur).newTexture(nomDeNouvelleTexture \
  {,#indicateurDeType, référenceDobjetSource})
```

Description

Commande 3D ; crée une texture dans la palette des textures de l'acteur référencé et renvoie une référence à la nouvelle texture. Les textures d'acteur ne fonctionnent que lorsque vous spécifiez l'acteur dans le constructeur `newTexture`.

Paramètres

nomDeNouvelleTexture Requis. Spécifie le nom de la nouvelle texture. Le nom de la nouvelle texture doit être unique dans la palette de textures de l'acteur référencé.

indicateurDeType Facultatif. Spécifie le type de la nouvelle texture. Si ce paramètre est omis, la nouvelle texture est créée sans type spécifique. Les valeurs correctes sont :

- `#fromCastMember` (un acteur)
- `#fromImageObject` (un objet image Lingo)

référenceDobjetSource Facultatif. Spécifie une référence à l'acteur ou à l'objet image Lingo source. Si ce paramètre est omis, la nouvelle texture est créée à partir d'aucune source spécifique. *référenceDobjetSource* doit faire référence à un acteur si *indicateurDeType* est défini sur `#fromCastMember` et à un objet image Lingo si *indicateurDeType* est défini sur `#fromImageObject`.

Exemple

La première ligne de cette instruction crée une texture nommée Gazon 02 à partir de l'acteur 5 de la distribution 1. La seconde ligne crée une texture vierge appelée Vierge.

```
member("Univers 3D").newTexture("Gazon \
  02",#fromCastMember,member(5,1))
member("Univers 3D").newTexture("Vierge")
```

normalize

Utilisation

```
normalize(vecteur)
vecteur.normalize()
```

Description

Commande 3D ; normalise un vecteur en divisant les composants *x*, *y* et *z* par la magnitude du vecteur. Les vecteurs normalisés ont toujours une magnitude de 1.

Paramètres

Aucun.

Exemple

L'instruction suivante indique la valeur du vecteur monVecteur avant et après la normalisation.

```
monVecteur = vector(-209.9019, 1737.5126, 0.0000)
monVecteur.normalize()
put monVecteur
-- vector( -0.1199, 0.9928, 0.0000 )
put monVecteur.magnitude
-- 1.0000
```

L'instruction suivante indique la valeur du vecteur ceVecteur avant et après la normalisation.

```
ceVecteur = vector(-50.0000, 0.0000, 0.0000)
normalize(ceVecteur)
put ceVecteur
-- vector( 0.0000, -1.0000, 0.0000 )
```

Voir aussi

`getNormalized`, `randomVector()`, `magnitude`

nothing

Utilisation

`nothing`

Description

Commande ; n'a aucun effet. Cette commande est pratique pour clarifier une instruction `if...then`. Une instruction imbriquée `if...then...else` ne contenant aucune commande explicite après la clause `else` peut nécessiter l'utilisation de `else nothing`, afin d'empêcher Lingo d'interpréter la clause `else` comme faisant partie de la clause `if` qui la précède.

Paramètres

Aucun.

Exemple

L'instruction imbriquée `if...then...else` du gestionnaire suivant utilise la commande `nothing` à la suite de la clause `else` de l'instruction :

```
-- Syntaxe Lingo
on mouseDown
  if the clickOn = 1 then
    if sprite(1).moveableSprite = TRUE then
  member("Notice").text = "Faites glisser la balle"
    else nothing
  else member("Notice").text = "Cliquez à nouveau"
  end if
end

// Syntaxe JavaScript
function mouseDown() {
  if (_mouse.clickOn == 1) {
    if (sprite(1).moveableSprite) {
      member("Notice").text = "Faites glisser la balle";
    } else {
      // do nothing
    }
  } else {
    member("Notice").text = "Cliquez à nouveau"
  }
}
```

Avec le gestionnaire suivant, l'animation n'évolue pas tant que l'utilisateur appuie sur le bouton de la souris :

```
-- Syntaxe Lingo
on mouseDown
  repeat while the stillDown
    nothing
  end repeat
end mouseDown
```

```
// Syntaxe JavaScript
function mouseDown() {
  do {
    // do nothing
  } while _mouse.stillDown;
}
```

Voir aussi

if

nudge()

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.nudge(#direction)

// Syntaxe JavaScript
réfObjImageObjet.nudge(#direction);
```

Description

Commande QuickTime VR ; déplace la perspective de l'image-objet QuickTime VR spécifiée dans la direction spécifiée.

Une poussée vers la droite entraîne un déplacement de l'image de l'image-objet vers la gauche. La commande `nudge` ne renvoie pas de valeur.

Paramètres

direction Requis. Spécifie la direction du déplacement de la perspective. Les valeurs correctes sont :

- `#down`
- `#downLeft`
- `#downRight`
- `#left`
- `#right`
- `#up`
- `#upLeft`
- `#upRight`

Exemple

Le gestionnaire suivant entraîne le déplacement vers la gauche de la perspective de l'image-objet QuickTime VR pendant que le pointeur de la souris est positionné sur l'image-objet.

```
-- Syntaxe Lingo
on mouseDown me
  repeat while the stillDown
    sprite(1).nudge(#left)
  end repeat
end
```

```
// Syntaxe JavaScript
function mouseDown() {
  do {
    sprite(1).nudge(#left);
  } while _mouse.stillDown;
}
```

numToChar()

Utilisation

`numToChar(`*`expressionEntière`*`)`

Description

Fonction ; affiche une chaîne contenant le caractère dont le code ASCII est la valeur d'une expression spécifiée. Elle est utile pour interpréter les données de sources externes qui sont présentées sous forme de nombres plutôt que sous forme de caractères.

Les valeurs ASCII allant jusqu'à 127 sont standard sur tous les ordinateurs. Les valeurs supérieures ou égales à 128 font référence à des caractères différents sur différents ordinateurs.

Paramètres

`expressionEntière` Requis. Spécifie le nombre ASCII dont le caractère correspondant est renvoyé.

Exemple

L'instruction suivante affiche dans la fenêtre Messages le caractère dont le code ASCII est 65 :

```
put numToChar(65)
```

Le résultat est la lettre *A*.

Le gestionnaire suivant supprime tous les caractères non alphabétiques d'une chaîne quelconque et ne renvoie que des majuscules :

```
-- Syntaxe Lingo
on ForcerDesMajuscules saisie
  sortie = EMPTY
  num = length(saisie)
  repeat with i = 1 to num
    leCodeASCII = charToNum(saisie.char[i])
    if leCodeASCII = min(max(96, leCodeASCII), 123) then
      leCodeASCII = leCodeASCII - 32
      if leCodeASCII = min(max(63, leCodeASCII), 91) then
        put numToChar(leCodeASCII) after sortie
      end if
    end if
  end repeat
  return sortie
end
```

```
// Syntaxe JavaScript
function ForcerDesMajuscules(saisie) {
  sortie = "";
  num = saisie.length;
  for (i=1;i<=num;i++) {
    leCodeASCII = saisie.char[i].charToNum();
    if (leCodeASCII == min(max(96, leCodeASCII), 123) {
      leCodeASCII = leCodeASCII - 32;
      if (leCodeASCII == min(max(63, leCodeASCII), 91) {
        sortie = sortie + leCodeASCII.numToChar();
      }
    }
  }
  return sortie;
}
```

Voir aussi

charToNum()

objectP()

Utilisation

objectP(*expression*)

Description

Fonction ; indique si une expression spécifiée est un objet créé par un script parent, un Xtra ou une fenêtre (TRUE) ou non (FALSE).

Le *P* dans objectP signifie *prédicat*.

Il est judicieux d'utiliser objectP pour déterminer les éléments en cours d'utilisation lors de la création d'objets par des scripts parents ou des instances d'Xtra.

Pour un exemple d'utilisation de objectP() dans une animation, reportez-vous à l'animation Read and Write Text du dossier Learning/Lingo, lui-même inclus dans le dossier de Director.

Paramètres

expression Requis. Spécifie l'expression à tester.

Exemple

L'instruction suivante vérifie si un objet est affecté à la variable globale gBaseDeDonnées et, dans la négative, en affecte un. Cette vérification s'utilise généralement lorsque vous effectuez des initialisations au début d'une animation ou d'une section qui ne doit pas être répétée.

```
-- Syntaxe Lingo
if objectP(gBaseDeDonnées) then
  nothing
else
  gBaseDeDonnées = script("Contrôleur de base de données").new()
end if
```

```
// Syntaxe JavaScript
if (objectP(BaseDeDonnées)) {
  // do nothing
} else {
  gBaseDeDonnées = script("Contrôleur de base de données").new();
}
```

Voir aussi

floatP(), ilk(), integerP(), stringP(), symbolP()

offset() (fonction de chaîne)

Utilisation

offset(*expressionChaîne1*, *expressionChaîne2*)

Description

Fonction ; renvoie un nombre entier indiquant la position du premier caractère d'une chaîne dans une autre chaîne. Cette fonction renvoie 0 si la première chaîne est introuvable dans la seconde chaîne. Lingo considère les espaces comme des caractères dans les deux chaînes.

Sur Macintosh, la comparaison des chaînes ne tient pas compte de la casse. Ainsi, Lingo considère *a* et *Å* comme identiques sur Macintosh.

Paramètres

expressionChaîne1 Requis. Spécifie la sous-chaîne à rechercher dans *expressionChaîne2*.

expressionChaîne2 Requis. Spécifie la chaîne qui contient la sous-chaîne *expressionChaîne1*.

Exemple

L'instruction suivante affiche dans la fenêtre Messages la position du début de la chaîne media dans la chaîne Macromedia :

```
put offset("media","Macromedia")
```

Le résultat est 6.

L'instruction suivante affiche dans la fenêtre Messages la position du début de la chaîne Micro dans la chaîne Macromedia :

```
put offset("Micro", "Macromedia")
```

Le résultat est 0 car « Macromedia » ne contient pas la chaîne « Micro ».

Le gestionnaire suivant recherche toutes les instances de la chaîne représentée par `chaîneAtrouver` dans la chaîne représentée par `saisie`, puis les remplace par la chaîne représentée par `chaîneAinsérer`.

```
-- Syntaxe Lingo
on ChercherEtRemplacer saisie, chaîneAtrouver, chaîneAinsérer
  sortie = ""
  Longueur = chaîneAtrouver.length - 1
  repeat saisie input contains chaîneAtrouver
    currOffset = offset(chaîneAtrouver, saisie)
    sortie = sortie & saisie.char [1..currOffset]
    delete the last char of sortie
    sortie = sortie & chaîneAinsérer
    delete input.char [1.. (currOffset + Longueur)]
  end repeat
  set sortie = sortie & saisie
  return sortie
end

// Syntaxe JavaScript
function ChercherEtRemplacer (saisie, chaîneAtrouver, chaîneAinsérer) {
  sortie = "";
  Longueur = chaîneAtrouver.length - 1;
  do {
    currOffset = offset(chaîneAtrouver, saisie);
    sortie = sortie + saisie.char [0..currOffset];
    sortie = sortie.substr(0,sortie.length-2);
    sortie = sortie + chaîneAinsérer;
    saisie = saisie.substr(currOffset+Longueur,saisie.length);
  } while (saisie.indexOf(chaîneAtrouver) >= 0);
  sortie = sortie + saisie;
  return sortie;
}
```

Voir aussi

`chars()`, `length()`, `contains`, `starts`

offset() (fonction de rectangle)

Utilisation

```
rectangle.offset(changementHorizontal, changementVertical)
offset (rectangle, changementHorizontal, changementVertical)
```

Description

Fonction ; produit un rectangle décalé par rapport au rectangle spécifié par *`rectangle`*.

Paramètres

`changementHorizontal` Requis. Spécifie le décalage horizontal, en pixels. Lorsque *`changementHorizontal`* est supérieur à 0, le décalage se produit vers la droite de la scène ; lorsque *`changementVertical`* est inférieur à 0, le décalage se produit vers la gauche de la scène.

`changementVertical` Requis. Spécifie le décalage vertical, en pixels. Lorsque *`changementVertical`* est supérieur à 0, le décalage se produit vers le bas de la scène ; lorsque *`changementHorizontal`* est inférieur à 0, le décalage se produit vers le haut de la scène.

Exemple

Le gestionnaire suivant déplace l'image-objet 1 de cinq pixels vers la droite et de cinq pixels vers le bas.

```
-- Syntaxe Lingo
on mouvementDiagonal
  nouveauRect=sprite(1).rect.offset(5, 5)
  sprite(1).rect=nouveauRect
end

// Syntaxe JavaScript
function diagonalMove() {
  nouveauRect = sprite(1).rect.offset(5,5);
  sprite(1).rect = nouveauRect;
}
```

open() (lecteur)

Utilisation

```
-- Syntaxe Lingo
_player.open({chaîneCheminDeDoc,} chaîneCheminDapplication)

// Syntaxe JavaScript
_player.open({chaîneCheminDeDoc,} chaîneCheminDapplication);
```

Description

Méthode de lecteur ; ouvre une application spécifiée et, éventuellement, un fichier spécifié en même temps que l'application.

Si *chaîneCheminDeDoc* ou *chaîneCheminDapplication* se trouvent dans un autre dossier que l'animation courante, vous devez spécifier le chemin d'accès complet du ou des fichiers.

L'ordinateur doit avoir assez de mémoire pour exécuter simultanément Director et d'autres applications.

Il s'agit d'une méthode très simple pour ouvrir une application ou un document au sein d'une application. Pour d'autres contrôles, consultez les options disponibles dans les Xtras fournis par d'autres développeurs.

Paramètres

chaîneCheminDeDoc Facultatif. Chaîne qui spécifie le document à ouvrir en même temps que l'application spécifiée par *chaîneCheminDapplication*.

chaîneCheminDapplication Requis. Chaîne qui spécifie le chemin de l'application à ouvrir.

Exemple

L'instruction suivante ouvre l'application TextEdit, qui se trouve dans le dossier Applications sur le disque dur (Macintosh), ainsi que le document Synopsis :

```
-- Syntaxe Lingo
_player.open("Synopsis", "HD:Applications:TextEdit")

// Syntaxe JavaScript
_player.open("Synopsis", "HD:Applications:TextEdit");
```

Voir aussi

Lecteur

open() (fenêtre)

Utilisation

```
-- Syntaxe Lingo
réfObjFenêtre.open()

// Syntaxe JavaScript
réfObjFenêtre.open();
```

Description

Méthode de fenêtre ; ouvre une fenêtre et la place devant toutes les autres fenêtres.

Si aucune animation n'est affectée à la fenêtre sur laquelle la méthode `open()` est appelée, la boîte de dialogue d'ouverture de fichier s'affiche.

Si la référence à l'objet fenêtre *`réfObjFenêtre`* est remplacée par le nom de fichier d'une animation, la fenêtre utilise le nom de fichier comme nom de fenêtre. Mais vous devez alors affecter une animation à la fenêtre avec la propriété `fileName` de la fenêtre.

Si la référence à l'objet fenêtre *`réfObjFenêtre`* est remplacée par un nom de fenêtre, la fenêtre prend ce nom. Mais vous devez alors affecter une animation à la fenêtre avec la propriété `fileName` de la fenêtre.

Pour ouvrir une fenêtre qui utilise une animation provenant d'une adresse URL, téléchargez le fichier sur votre disque local à l'aide de la commande `downloadNetThing`, puis utilisez-le depuis votre disque local. Cette procédure minimise les problèmes d'attente pendant le téléchargement.

Lorsque vous utilisez une animation locale, chargez au moins la première image de l'animation avec `preloadMovie()` avant d'appeler `open()`. Cette procédure réduit les risques de retard liés au chargement des animations.

L'ouverture d'une animation dans une fenêtre n'est pas encore supportée pour la lecture dans un navigateur web.

Paramètres

Aucun.

Exemple

L'instruction suivante ouvre la fenêtre Tableau de commande et la place au premier plan :

```
-- Syntaxe Lingo
window("Tableau de commande").open()

// Syntaxe JavaScript
window("Tableau de commande").open();
```

Voir aussi

close(), downloadNetThing, fileName (fenêtre), preLoadMovie(), Fenêtre

openFile()

Utilisation

```
-- Syntaxe Lingo
réfObjFileio.openFile(chaîneNomDeFichier, entMode)

// Syntaxe JavaScript
réfObjFileio.openFile(chaîneNomDeFichier, entMode)
```

Description

Méthode de Fileio ; ouvre un fichier spécifié avec un mode spécifié.

Paramètres

chaîneNomDeFichier Requis. Chaîne qui spécifie le chemin complet et le nom du fichier à ouvrir.

entMode Requis. Nombre entier qui spécifie le mode du fichier. Les valeurs correctes sont :

- 0—Lecture/écriture
- 1—Lecture seule
- 2—Ecriture

Voir aussi

Fileio

openXlib

Utilisation

```
openXlib quelFichier
```

Description

Commande ; ouvre un fichier Xlibrary spécifié.

Il est recommandé de fermer tout fichier ouvert dès que vous n'en avez plus besoin. La commande `openXlib` n'a aucun effet sur un fichier ouvert.

La commande `openXlib` ne supporte pas les URL comme références de fichiers.

Les fichiers Xlibrary contiennent des Xtras. A la différence de `openResFile`, `openXlib` met ces Xtras à la disposition de Director.

Lorsque vous ouvrez un Xtra de programmation avec `openXlib`, utilisez `closeXlib` pour le fermer lorsque Director a fini de l'utiliser.

Sous Windows, l'extension .dll est facultative.

Remarque : Cette commande n'est pas prise en charge dans Shockwave Player.

Paramètres

quelFichier Requis. Spécifie le fichier Xlibrary à ouvrir. Si le fichier n'est pas dans le dossier de l'animation courante, *quelFichier* doit inclure le chemin.

Exemple

L'instruction suivante ouvre le fichier Xlibrary Vidéodisque :

```
openXlib "Xlibrary Vidéodisque"
```

L'instruction suivante ouvre le fichier Xlibrary Xtras, situé dans un dossier différent de celui de l'animation courante :

```
openXlib "Mon disque:Nouveautés:Transporter Xtras"
```

Voir aussi

closeXlib, interface()

param()

Utilisation

param(*positionDeParamètre*)

Description

Fonction ; renvoie la valeur d'un paramètre transmis à un gestionnaire.

Cette fonction peut s'utiliser pour déterminer le type d'un paramètre particulier afin d'éviter les erreurs dans un gestionnaire.

Paramètres

positionDeParamètre Requis. Spécifie la position du paramètre dans les arguments transmis au gestionnaire.

Exemple

Le gestionnaire suivant accepte un nombre quelconque d'arguments, fait le total du nombre transmis en paramètres, puis renvoie la somme :

```
-- Syntaxe Lingo
on ajouterLesNombres
  somme = 0
  repeat with numDeParamCourant = 1 to the paramCount
    somme = somme + param(numDeParamCourant)
  end repeat
  return somme
end
```

```
// Syntaxe JavaScript
function ajouterLesNombres() {
  somme = 0
  for (numDeParamCourant=1;numDeParamCourant<=paramCount;numDeParamCourant++)
  {
    somme = somme + param(numDeParamCourant);
  }
  return somme;
}
```

Vous l'utilisez en transmettant les valeurs à ajouter :

```
put ajouterLesNombres(3, 4, 5, 6)
-- 18
put ajouterLesNombres(5, 5)
-- 10
```

Voir aussi

`getAt`, `param()`, `paramCount()`, `return (mot-clé)`

paramCount()

Utilisation

```
the paramCount
```

Description

Fonction ; indique le nombre de paramètres transmis au gestionnaire courant.

Paramètres

Aucun.

Exemple

L'instruction suivante définit la variable `compteur` en fonction du nombre de paramètres transmis au gestionnaire courant :

```
set compteur = the paramCount
```

parseString()

Utilisation

`objetDanalyse``.parseString(`*`chaîneAanalyser`*`)`

Description

Fonction ; utilisée pour analyser un document XML déjà disponible pour une animation Director. Le premier paramètre est la variable contenant l'objet d'anamyse. La valeur renvoyée est `<VOID>` en cas de succès de l'opération ou un code d'erreur en cas d'échec. L'échec est généralement dû à un problème au niveau de la syntaxe ou de la structure XML. Une fois l'opération terminée, l'objet d'analyse contient les données XML analysées.

L'analyse du code XML d'une URL requiert l'utilisation de la commande `parseURL()`.

Paramètres

`chaîneAanalyser` Requis. Spécifie la chaîne de données XML à analyser.

Exemple

L'instruction suivante analyse les données XML de l'acteur texte texteXML. Une fois l'opération terminée, la variable `gObjetDanalyse` contient les données XML analysées.

```
codeErreur = gObjetDanalyse.parseString(member("texteXML"))
```

Voir aussi

`getError() (XML)`, `parseURL()`

parseURL()

Utilisation

```
objetDanalyse.parseURL(chaîneURL {,#gestionnaireAappelerAlaFin} {,
  objetContenantLeGestionnaire})
```

Description

Fonction ; analyse un document XML résidant à une adresse Internet externe. Le premier paramètre est l'objet d'analyse contenant une instance de l'Xtra XMLParser.

Cette fonction renvoie un résultat immédiat et l'adresse URL complète peut donc ne pas être encore analysée. Il est important d'utiliser la fonction `doneParsing()` avec `parseURL()` afin d'être averti(e) de la fin de l'opération d'analyse.

Cette opération étant asynchrone (et pouvant prendre un certain temps), vous pouvez utiliser des paramètres facultatifs permettant d'appeler un gestionnaire spécifique à la fin de l'opération.

La valeur renvoyée est `VOID` en cas de succès de l'opération ou un code d'erreur en cas d'échec.

L'analyse du code XML local requiert l'utilisation de la commande `parseString()`.

Paramètres

`chaîneURL` Requis. Spécifie l'URL réelle où résident les données XML.

`gestionnaireAappelerAlaFin` Facultatif. Spécifie le nom du gestionnaire à exécuter une fois l'analyse de l'URL terminée.

`objetContenantLeGestionnaire` Facultatif. Spécifie le nom de l'objet script contenant le gestionnaire *`gestionnaireAappelerAlaFin`*. Si ce paramètre est omis, le gestionnaire est supposé être un gestionnaire d'animation.

Exemple

L'instruction suivante analyse le fichier sample.xml de monEntreprise.fr. Utilisez la fonction `doneParsing()` pour déterminer à quel moment l'opération d'analyse est terminée.

```
codeErreur = gObjetDanalyse.parseURL("http://www.monEntreprise.fr/
  exemple.xml")
```

L'instruction Lingo suivante analyse le fichier exemple.xml et appelle le gestionnaire `on analyseTerminée`. Aucun objet script n'étant donné avec la fonction `doneParsing()`, le gestionnaire `on analyseTerminée` est considéré comme étant dans un script de l'animation.

```
codeErreur = gObjetDanalyse.parseURL("http://www.monEntreprise.fr/
  exemple.xml", #analyseTerminée)
```

Le script de l'animation contient le gestionnaire `on analyseTerminée` :

```
on analyseTerminée
  global gObjetDanalyse
  if voidP(gObjetDanalyse.getError()) then
    put "Analyse réussie"
  else
    put "Erreur d'analyse :"
    put "   " & gObjetDanalyse.getError()
  end if
end
```

L'instruction Lingo suivante analyse le document exemple.xml qui se trouve sur www.monEntreprise.fr et appelle le gestionnaire `on analyseTerminée` de l'objet script `objetTest`, qui est un enfant du script parent scriptTest :

```
objetDanalyse = new(xtra "XMLParser")
objetTest = new(script "scriptTest", objetDanalyse)
codeErreur = gObjetDanalyse.parseURL("http://www.monEntreprise.fr/
  exemple.xml", #analyseTerminée, objetTest)
```

Le script parent scriptTest est le suivant :

```
property monObjetDanalyse

on new me, objetDanalyse
  monObjetDanalyse = objetDanalyse
end

on analyseTerminée me
  if voidP(monObjetDanalyse.getError()) then
    put "Analyse réussie"
  else
    put "Erreur d'analyse :"
    put "   " & monObjetDanalyse.getError()
  end if
end
```

Voir aussi

`getError() (XML)`, `parseString()`

pass

Utilisation

```
pass
```

Description

Commande ; transmet un message d'événement à la position suivante dans la hiérarchie des messages et déclenche l'exécution de plusieurs gestionnaires pour un événement donné.

La commande `pass` passe à la position suivante dès son exécution. Aucun élément suivant la commande `pass` dans le gestionnaire n'est exécuté.

Par défaut, un message d'événement s'arrête à la première position contenant un gestionnaire pour l'événement, généralement au niveau de l'image-objet.

L'inclusion de la commande `pass` dans un gestionnaire entraîne la transmission de l'événement à d'autres objets dans la hiérarchie, même si le gestionnaire pourrait autrement intercepter l'événement.

Paramètres

Aucun.

Exemple

Le gestionnaire suivant détermine les touches enfoncées et autorise leur transmission à l'image-objet texte modifiable s'il s'agit de caractères valides :

```
-- Syntaxe Lingo
on keyDown me
  caractèresAdmis = "1234567890"
  if caractèresAdmis contains the key then
    pass
  else
    beep
  end if
end

// Syntaxe JavaScript
function keyDown() {
  caractèresAdmis = "1234567890";
  if (caractèresAdmis.indexOf(_key.key) >= 0) {
    pass();
  } else {
    _sound.beep();
  }
}
```

Voir aussi

`stopEvent()`

pasteClipBoardInto()

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.pasteClipBoardInto()

// Syntaxe JavaScript
réfObjActeur.pasteClipBoardInto();
```

Description

Méthode d'acteur ; colle le contenu du Presse-papiers dans un acteur spécifié et efface l'acteur existant.

Vous pouvez coller tout élément dont le format peut être utilisé par Director pour un acteur.

Lorsque vous copiez une chaîne à partir d'une autre application, son format n'est pas conservé.

Cette méthode est un moyen pratique de copier dans la fenêtre Distribution des objets provenant d'autres animations et d'autres applications. Les acteurs copiés devant être conservés en RAM, évitez d'utiliser cette commande pendant la lecture dans les situations où la mémoire arrive à épuisement.

Lorsque vous utilisez cette méthode dans Shockwave Player ou dans l'environnement auteur et avec des projections dont la propriété `safePlayer` a la valeur `TRUE`, une boîte de dialogue d'avertissement s'affiche pour permettre à l'utilisateur d'annuler l'opération de collage.

Paramètres

Aucun.

Exemple

L'instruction suivante colle le contenu du Presse-papiers dans l'acteur bitmap Temple :

```
-- Syntaxe Lingo
member("Temple").pasteClipBoardInto()

// Syntaxe JavaScript
member("Temple").pasteClipBoardInto();
```

Voir aussi

Acteur, safePlayer

pause() (DVD)

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.pause()

// Syntaxe JavaScript
réfObjDvd.pause();
```

Description

Méthode de DVD ; met en pause la lecture.

Paramètres

Aucun.

Voir aussi

DVD

pause() (piste audio)

Utilisation

```
-- Syntaxe Lingo
réfObjPisteAudio.pause()

// Syntaxe JavaScript
réfObjPisteAudio.pause();
```

Description

Méthode de piste audio ; interrompt la lecture du son courant dans une piste audio.

Une méthode `play()` subséquente entraînera la reprise de la lecture.

Paramètres

Aucun.

Exemple

L'instruction suivante met en pause la lecture de l'acteur son lu dans la piste audio 1 :

```
-- Syntaxe Lingo
sound(1).pause()

// Syntaxe JavaScript
sound(1).pause();
```

Voir aussi

breakLoop(), play() (piste audio), playNext() (piste audio), queue(), rewind() (piste audio), Piste audio, stop() (piste audio)

pause() (3D)

Utilisation

```
member(quelActeur).model(quelModèle).bonesPlayer.pause()
member(quelActeur).model(quelModèle).keyframePlayer.pause()
```

Description

Commande de modificateur 3D #keyframePlayer et #bonesPlayer ; stoppe le mouvement du modèle en cours d'exécution. Utilisez la commande play() pour que le mouvement reprenne son cours.

Lorsque le mouvement d'un modèle est arrêté à l'aide de cette commande, la propriété bonesPlayer.playing du modèle prend la valeur FALSE.

Paramètres

Aucun.

Exemple

L'instruction suivante met l'animation courante du modèle Fourmi3 en pause.

```
member("ScèneDePiqueNique").model("Fourmi3").bonesplayer.pause()
```

Voir aussi

play() (3D), playing (3D), playlist

pause() (RealMedia, SWA, Windows Media)

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.pause()

// Syntaxe JavaScript
réfObjActeurOuImageObjet.pause();
```

Description

Méthode d'image-objet ou d'acteur RealMedia et Windows Media ; interrompt la lecture du flux multimédia.

La valeur de mediaStatus devient #paused.

L'appel de cette méthode, lors de la lecture du flux RealMedia ou Windows Media, ne modifie pas la propriété `currentTime` et n'efface pas le contenu de la mémoire tampon. En revanche, elle permet aux commandes `play` suivantes de reprendre la lecture sans devoir remettre le flux RealMedia en tampon.

Paramètres

Aucun.

Exemple

Les exemples suivants arrêtent la lecture de l'image-objet 2 ou de l'acteur Real.

```
-- Syntaxe Lingo
sprite(2).pause()
member("Real").pause()

// Syntaxe JavaScript
sprite(2).pause();
member("Real").pause();
```

Voir aussi

mediaStatus (RealMedia, Windows Media), play() (RealMedia, SWA, Windows Media), seek(), stop() (RealMedia, SWA, Windows Media)

perpendicularTo

Utilisation

vecteur1.perpendicularTo(*vecteur2*)

Description

Commande 3D de vecteur ; renvoie un vecteur perpendiculaire au vecteur d'origine et à un second vecteur. Cette commande équivaut à la commande de vecteur `crossProduct`.

Paramètres

vecteur2 Requis. Spécifie le second vecteur.

Exemple

Dans l'exemple suivant, pos1 est un vecteur sur l'axe des x et pos2 est un vecteur sur l'axe des y. La valeur renvoyée par `pos1.perpendicularTo(pos2)` est `vector( 0.0000, 0.0000, 1.00000e4 )`. Les deux dernières lignes de l'exemple indiquent le vecteur perpendiculaire à pos1 et pos2.

```
pos1 = vector(100, 0, 0)
pos2 = vector(0, 100, 0)
put pos1.perpendicularTo(pos2)
-- vector( 0.0000, 0.0000, 1.00000e4 )
```

Voir aussi

crossProduct(), cross

pictureP()

Utilisation

```
-- Syntaxe Lingo
pictureP(valeurDimage)

// Syntaxe JavaScript
pictureP(valeurDimage);
```

Description

Fonction ; indique si l'état de la propriété `picture` de l'acteur spécifié est `TRUE` ou `FALSE`.

Etant donné que `pictureP` ne vérifie pas directement si une image est associée à un acteur, vous devez le faire en vérifiant la propriété `picture` de l'acteur.

Paramètres

`valeurDimage` Requis. Spécifie une référence à l'image d'un acteur.

Exemple

La première instruction affecte la valeur de la propriété `picture` de l'acteur Temple, qui est un bitmap, à la variable `valeurDimage`. La seconde vérifie si Temple est une image en vérifiant la valeur affectée à `valeurDimage`.

```
-- Syntaxe Lingo
valeurDimage = member("Temple").picture
put pictureP(valeurDimage)

// Syntaxe JavaScript
var valeurDimage = member("Temple").picture;
put(pictureP(valeurDimage));
```

Le résultat est 1, équivalent numérique de `TRUE`.

play() (3D)

Utilisation

```
member(quelActeur).model(quelModèle).bonesPlayer.play()
member(quelActeur).model(quelModèle).keyframePlayer.play()
member(quelActeur).model(quelModèle).bonesPlayer.\
  play(nomDeMouvement {, enBoucle, positionInitiale, positionFinale, échelle,
  décalage})
member(quelActeur).model(quelModèle).keyframePlayer.\
  play(nomDeMouvement {, enBoucle, positionInitiale, positionFinale, échelle,
  décalage})
```

Description

Commande 3D `#keyframePlayer` et `#bonesPlayer` ; entraîne ou reprend l'exécution d'un mouvement.

Lorsque le mouvement d'un modèle est démarré ou redémarré à l'aide de cette commande, la propriété `bonesPlayer.playing` du modèle prend la valeur `TRUE`.

Utilisez `play()` sans paramètre pour reprendre l'exécution d'un mouvement qui a été arrêté à l'aide de la commande `pause()`.

L'utilisation de la commande `play()` pour démarrer un mouvement insère le mouvement au début de la liste de lecture du modificateur. Si cela interrompt la lecture d'un autre mouvement, le mouvement interrompu reste dans la liste de lecture et est positionné après le mouvement qui vient d'être démarré. Lorsque le mouvement qui vient d'être démarré se termine (s'il n'est pas en boucle) ou que la commande `playNext()` est émise, la lecture du mouvement interrompu recommence là où elle s'était arrêtée.

Paramètres

`nomDeMouvement` Requis. Spécifie le nom du mouvement à exécuter. Lorsque *`nomDeMouvement`* est le seul paramètre transmis à `play()`, le mouvement est exécuté une fois par le modèle du début à la fin, à la cadence définie par la propriété `playRate` du modificateur.

`enBoucle` Facultatif. Spécifie si le mouvement est lu une seule fois (`FALSE`) ou continuellement (`TRUE`).

`positionInitiale` Facultatif. Ce paramètre est mesuré en millisecondes, à partir du début du mouvement. Lorsque *`enBoucle`* a pour valeur `TRUE`, la première itération de la boucle commence à *`décalage`* et se termine à *`positionFinale`*, avec toutes les répétitions suivantes du mouvement démarrant à *`positionInitiale`* et se terminant à *`positionFinale`*.

`positionFinale` Facultatif. Ce paramètre est mesuré en millisecondes, à partir du début du mouvement. Lorsque *`enBoucle`* a pour valeur `FALSE`, le mouvement démarre à la position *`décalage`* et se termine à la *`positionFinale`*. Lorsque *`enBoucle`* a pour valeur `TRUE`, la première itération de la boucle commence à *`décalage`* et se termine à *`positionFinale`*, avec toutes les répétitions suivantes démarrant à *`positionInitiale`* et se terminant à *`positionFinale`*. Donnez à *`positionFinale`* la valeur -1 si le mouvement doit être lu jusqu'à la fin.

`cadenceDeLecture` Facultatif. Spécifie la cadence réelle de la lecture du mouvement. La valeur du paramètre *`cadenceDeLecture`* est multipliée par la propriété `playRate` du modificateur `#keyframePlayer` ou `#bonesPlayer` du modèle pour déterminer la cadence réelle de la lecture du mouvement.

`décalage` Facultatif. Ce paramètre est mesuré en millisecondes, à partir du début du mouvement. Lorsque *`enBoucle`* a pour valeur `FALSE`, le mouvement démarre à la position *`décalage`* et se termine à la *`positionFinale`*. Lorsque *`enBoucle`* a pour valeur `TRUE`, la première itération de la boucle commence à *`décalage`* et se termine à *`positionFinale`*, avec toutes les répétitions suivantes démarrant à *`positionInitiale`* et se terminant à *`positionFinale`*. Vous pouvez également donner au paramètre *`décalage`* la valeur `#synchronized` pour démarrer le mouvement à la même position, par rapport à la durée, que l'animation courante.

Exemple

La commande suivante entraîne le modèle Marcheur à lire le mouvement Chute. Après la lecture de ce mouvement, le modèle reprendra la lecture de tout autre mouvement précédemment interrompu.

```
sprite(1).member.model("Marcheur").bonesPlayer.play("Chute", 0, \
  0, -1, 1, 0)
```

La commande suivante entraîne le modèle Marcheur à lire le mouvement coupDenvoi. Si Marcheur est en train d'exécuter un mouvement, il sera interrompu par le mouvement coupDenvoi dont une section sera jouée en boucle. La première itération de la boucle démarrera 2000 millisecondes à compter du début du mouvement. Toutes les itérations suivantes de la boucle démarreront à 1000 millisecondes du début de coupDenvoi et se termineront à 5000 millisecondes du début de coupDenvoi. La cadence de lecture sera égale à trois fois la propriété `playRate` du modificateur `bonesPlayer` du modèle.

```
sprite(1).member.model("Marcheur").bonesPlayer.play("coupDenvoi", 1, \
  1000, 5000, 3, 2000)
```

Voir aussi

`queue() (3D)`, `playNext() (3D)`, `playRate (3D)`, `playlist`, `pause() (3D)`, `removeLast()`, `playing (3D)`

play() (DVD)

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.play()
réfObjDvd.play(titreDébut, chapitreDébut, titreFin, chapitreFin)
réfObjDvd.play(listePositionsDeDébut, listePositionsDeFin)

// Syntaxe JavaScript
réfObjDvd.play();
réfObjDvd.play(titreDébut, chapitreDébut, titreFin, chapitreFin);
réfObjDvd.play(listePositionsDeDébut, listePositionsDeFin);
```

Description

Méthode de DVD ; commence ou reprend la lecture.

En l'absence de paramètres, cette méthode reprend la lecture au début d'un disque ou, si la lecture a été arrêtée, à la valeur spécifiée par la propriété . La lecture se poursuit jusqu'à la valeur spécifiée par la propriété `stopTimeList`, si elle a été définie.

Avec les paramètres *`titreDébut`*, *`chapitreDébut`*, *`titreFin`* et *`chapitreFin`*, cette méthode commence la lecture à un titre et un chapitre donnés. La lecture continue jusqu'aux paramètres *`titreFin`* et *`chapitreFin`*, s'ils sont définis.

Avec les paramètres *`listePositionsDeDébut`* et *`listePositionsDeFin`*, cette méthode lit la valeur spécifiée par le paramètre `listePositionsDeDébut` jusqu'à la valeur spécifiée par le paramètre `listePositionsDeFin`.

Les formats de liste suivants sont utilisés pour `listePositionsDeDébut` et `listePositionsDeFin` :

```
[#title:1, #chapter:1, #hours:0, #minutes:1, #seconds:1]
```

or

```
[#title:1, #hours:0, #minutes:1, #seconds:1]
```

Cette méthode renvoie la valeur 0 en cas de réussite.

Paramètres

titreDébut Requis si la lecture commence au niveau d'un titre et d'un chapitre donnés. Nombre entier qui spécifie le titre contenant le chapitre à lire. Ce paramètre remplacera la propriété de l'acteur.

chapitreDébut Requis si la lecture commence au niveau d'un titre et d'un chapitre donnés. Nombre entier qui spécifie le chapitre à lire. Ce paramètre remplacera la propriété de l'acteur.

titreFin Requis si la lecture doit s'arrêter au niveau d'un titre et d'un chapitre donnés. Nombre entier qui spécifie le titre auquel la lecture s'arrêtera. Ce paramètre remplacera la propriété de l'acteur.

chapitreFin Requis si la lecture doit s'arrêter au niveau d'un titre et d'un chapitre donnés. Nombre entier qui spécifie le chapitre à lire. Ce paramètre remplacera la propriété de l'acteur.

listePositionsDeDébut Requis si la lecture commence à une position donnée. Liste de propriétés qui spécifie la position à laquelle la lecture commence. Ce paramètre remplacera la propriété de l'acteur.

listePositionsDeFin Requis si la lecture commence à une position donnée. Liste de propriétés qui spécifie la position à laquelle la lecture s'arrête. Ce paramètre remplacera la propriété de l'acteur.

Exemple

Cette instruction reprend la lecture d'une image-objet en pause :

```
-- Syntaxe Lingo
member(12).play()

// Syntaxe JavaScript
member(12).play();
```

Ces instructions commencent la lecture au chapitre 2 du titre 1 et l'arrêtent au chapitre 4 :

```
member(15).play([#title:1, #chapter:2], [#title:1, #chapter:4])
```

or

```
member(15).play(1,2,1,4)
```

Cette instruction commence la lecture au niveau de la 10ème seconde du chapitre 2 et l'arrête à la 17ème seconde :

```
member(15).play([#title:2, #seconds:10], [#title:2, #seconds:17])
```

Voir aussi

DVD, startTimeList, stopTimeList

play() (piste audio)

Utilisation

```
-- Syntaxe Lingo
réfObjPisteAudio.play()
réfObjPisteAudio.play(réfObjActeur)
réfObjPisteAudio.play(listeDesPropriétés)

// Syntaxe JavaScript
réfObjPisteAudio.play();
réfObjPisteAudio.play(réfObjActeur);
réfObjPisteAudio.play(listeDesPropriétés);
```

Description

Méthode de piste audio ; commence la lecture des sons mis en file d'attente sur une piste audio ou met en file d'attente et commence de lire un acteur donné.

Les acteurs son mettent un certain temps à se charger en mémoire RAM avant que la lecture ne puisse commencer. Il est conseillé de placer les sons en file d'attente avec `queue()` avant d'entamer leur lecture et d'utiliser ensuite la première forme de cette méthode. La seconde forme ne tire pas parti du préchargement accompli à l'aide de la commande `queue()`.

L'utilisation d'une liste de propriétés facultative permet de définir les paramètres de lecture exacts d'un son.

Pour un exemple d'utilisation de `queue()` dans une animation, reportez-vous à l'animation Sound Control du dossier Learning/Lingo, lui-même inclus dans le dossier de Director.

Paramètres

`réfObjActeur` Requis si vous lisez un acteur son spécifique. Référence à l'objet acteur à mettre en file d'attente et à lire.

`listeDesPropriétés` Requis si vous spécifiez des paramètres de lecture pour un son. Liste de propriétés qui spécifie les paramètres de lecture exacts pour le son. La définition de ces propriétés est facultative :

Propriété	Description
`#member`	L'acteur à placer en file d'attente. Cette propriété doit être spécifiée. Toutes les autres sont facultatives.
`#startTime`	Position temporelle de départ de la lecture du son, en millisecondes. La valeur par défaut est le début du son. Pour plus d'informations, consultez `startTime`.
`#endTime`	Position temporelle de fin de la lecture du son, en millisecondes. La valeur par défaut est la fin du son. Pour plus d'informations, consultez `endTime`.
`#loopCount`	Nombre de répétitions d'une boucle défini avec `#loopStartTime` et `#loopEndTime`. La valeur par défaut est 1. Voir `loopCount`.
`#loopStartTime`	Position temporelle de départ de la boucle, en millisecondes. Pour plus d'informations, consultez `loopStartTime`.

Propriété	Description
#loopEndTime	Position temporelle de fin de la boucle, en millisecondes. Pour plus d'informations, consultez loopEndTime.
#preloadTime	Quantité de son à placer en mémoire tampon avant la lecture, en millisecondes. Pour plus d'informations, consultez preloadTime.

Exemple

L'instruction suivante lit l'acteur son Intro sur la piste audio 1 :

```
-- Syntaxe Lingo
sound(1).play(member("Intro"))

// Syntaxe JavaScript
sound(1).play(member("Intro"));
```

L'instruction suivante exécute la lecture de l'acteur Crédits sur la piste audio 2. La lecture commence à la quatrième seconde du son et se termine à la quinzième seconde. La section comprise entre 10,5 et 14 secondes exécute une lecture en boucle à 6 reprises.

```
-- Syntaxe Lingo
sound(2).play([#member:member("Crédits"), #startTime:4000, \
  #endTime:15000, #loopCount:6, #loopStartTime:10500, #loopEndTime:14000])

// Syntaxe JavaScript
sound(2).play(propList("member",member("Crédits"), "startTime",4000,
  "endTime",15000, "loopCount",6, "loopStartTime",10500,
  "loopEndTime",14000));
```

Voir aussi

endTime, loopCount, loopEndTime, loopStartTime, pause() (piste audio), preLoadTime, queue(), Piste audio, startTime, stop() (piste audio)

play() (RealMedia, SWA, Windows Media)

Utilisation

```
-- Syntaxe Lingo
réfObjWindowsMedia.play()
réfObjRealMedia.play()

// Syntaxe JavaScript
réfObjWindowsMedia.play();
réfObjRealMedia.play();
```

Description

Méthode d'acteur ou d'image-objet Windows Media ou RealMedia ; lit l'acteur Windows Media ou RealMedia ou lit l'image-objet sur la scène.

Pour les acteurs, seul le son est rendu s'il est présent dans l'animation. Si l'acteur est déjà en cours de lecture, cette méthode n'a aucun effet.

Paramètres

Aucun.

Exemple

Les exemples suivants démarrent le processus de lecture en flux continu de l'image-objet 2 et de l'acteur Real.

```
-- Syntaxe Lingo
sprite(2).play()
member("Real").play()

// Syntaxe JavaScript
sprite(2).play();
member("Real").play();
```

Voir aussi

RealMedia, Windows Media

playFile()

Utilisation

```
-- Syntaxe Lingo
réfObjPisteAudio.playFile(chaîneCheminDeFichier)

// Syntaxe JavaScript
réfObjPisteAudio.playFile(chaîneCheminDeFichier);
```

Description

Méthode de piste audio ; lit le son AIFF, SWA, AU ou WAV dans une piste audio.

Pour que le son soit lu correctement, l'Xtra MIX qui convient doit être accessible à l'animation (il est généralement placé dans le dossier des Xtras de l'application).

Lorsque le fichier audio ne se trouve pas dans le même dossier que l'animation, *chaîneCheminDeFichier* doit indiquer son chemin d'accès complet.

Pour lire des sons obtenus à partir d'une adresse URL, il est judicieux d'utiliser la commande `downloadNetThing()` ou `preloadNetThing()` pour télécharger d'abord le fichier sur le disque local. Vous éviterez ainsi les problèmes pouvant se produire en attente de téléchargement.

La méthode `playFile()` lit les fichiers en flux continu à partir du disque dur au lieu de le faire depuis la mémoire vive. Par conséquent, l'utilisation de la mathode `playFile()` pendant la lecture d'une vidéo numérique ou le chargement d'acteurs en mémoire peut occasionner des conflits, l'ordinateur tentant de lire simultanément deux emplacements du disque.

Paramètres

chaîneCheminDeFichier Requis. Chaîne qui spécifie le nom du fichier à lire. Lorsque le fichier audio ne se trouve pas dans le même dossier que l'animation en cours de lecture, *chaîneCheminDeFichier* doit indiquer son chemin d'accès complet.

Exemple

L'instruction suivante lit le fichier Tonnerre dans la piste 1 :

```
-- Syntaxe Lingo
sound(1).playFile("Tonnerre.wav")

// Syntaxe JavaScript
sound(1).playFile("Tonnerre.wav");
```

L'instruction suivante lit le fichier Tonnerre dans la piste 3 :

```
-- Syntaxe Lingo
sound(3).playFile(_movie.path & "Tonnerre.wav")

// Syntaxe JavaScript
sound(3).playFile(_movie.path + "Tonnerre.wav");
```

Voir aussi

play() (piste audio), Piste audio, stop() (piste audio)

playFromToTime()

Utilisation

```
-- Syntaxe Lingo
réfObjWindowsMedia.playFromToTime(entPositionInitiale, entPositionFinale)

// Syntaxe JavaScript
réfObjWindowsMedia.playFromToTime(entPositionInitiale, entPositionFinale);
```

Description

Méthode d'image-objet Windows Media. Démarre la lecture à une position de départ spécifiée et l'interrompt à une position de fin spécifiée.

Paramètres

entPositionInitiale Requis. Nombre entier qui spécifie la position, en millisecondes, à partir de laquelle la lecture commence.

entPositionFinale Requis. Nombre entier qui spécifie la position, en millisecondes, à laquelle la lecture s'arrête.

Exemple

Cette instruction spécifie que l'image-objet intitulée Vidéo doit être lue du repère des 30 secondes au repère des 40 secondes.

```
-- Syntaxe Lingo
sprite("Vidéo").playFromToTime(30000, 40000)

// Syntaxe JavaScript
sprite("Vidéo").playFromToTime(30000, 40000);
```

Voir aussi

Windows Media

playNext() (piste audio)

Utilisation

```
-- Syntaxe Lingo
réfObjPisteAudio.playNext()

// Syntaxe JavaScript
réfObjPisteAudio.playNext();
```

Description

Méthode de piste audio ; entraîne l'interruption immédiate de la lecture du son sur la piste audio concernée et entame la lecture du son suivant placé en file d'attente.

Si la file d'attente ne contient pas d'autres sons, la lecture du son est simplement stoppée.

Paramètres

Aucun.

Exemple

L'instruction suivante lit le son suivant placé en file d'attente dans la piste audio 2.

```
-- Syntaxe Lingo
sound(2).playNext()

// Syntaxe JavaScript
sound(2).playNext();
```

Voir aussi

pause() (piste audio), play() (piste audio), Piste audio,stop() (piste audio)

playNext() (3D)

Utilisation

```
member(quelActeur).model(quelModèle).bonesPlayer.playNext()
member(quelActeur).model(quelModèle).keyframePlayer.playNext()
```

Description

Commande 3D de modificateur #keyframePlayer et #bonesPlayer ; démarre la lecture du mouvement suivant dans la liste de lecture du modificateur #keyframePlayer ou #bonesPlayer du modèle. Le mouvement en cours de lecture, qui est la première entrée de la liste de lecture, est interrompu et retiré de la liste.

Si la fusion des mouvements est activée et qu'au moins deux mouvements sont présents dans la liste de lecture, la fusion du mouvement courant avec le suivant dans la liste de lecture commencera avec l'appel de playNext().

Exemple

L'instruction suivante interrompt le mouvement actuellement exécuté par le modèle 1 et démarre la lecture du mouvement suivant dans la liste de lecture :

```
member("scène").model[1].bonesPlayer.playnext()
```

Voir aussi

blend (3D), playlist

playerParentalLevel()

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.playerParentalLevel()

// Syntaxe JavaScript
réfObjDvd.playerParentalLevel();
```

Description

Méthode de DVD ; renvoie le niveau parental du lecteur.

Les niveaux parentaux possibles sont compris entre 1 et 8.

Paramètres

Aucun.

Voir aussi

DVD

point()

Utilisation

```
-- Syntaxe Lingo
point(entH, entV)

// Syntaxe JavaScript
point(entH, entV);
```

Description

Fonction du niveau supérieur et type de données. Renvoie un point avec des coordonnées horizontale et verticale spécifiées.

Un point a une propriété `locH` et une propriété `locV`.

Les coordonnées d'un point peuvent être modifiées par des opérations arithmétiques avec Lingo uniquement. Par exemple, les deux points suivants peuvent être ajoutés ensemble avec Lingo, mais `NaN` est renvoyé avec la syntaxe JavaScript :

```
-- Lingo
pointA = point(10,10)
pointB = point(5,5)
put(pointA + pointB)
-- point(15,15)

// Syntaxe JavaScript
var pointA = point(10,10);
var pointB = point(5,5);
trace(pointA + pointB);
// NaN
```

Pour un exemple d'utilisation de `point()` dans une animation, reportez-vous aux animations Imaging et Vector Shapes du dossier Learning/Lingo, lui-même inclus dans le dossier de Director.

Paramètres

entH Requis. Entier qui spécifie la coordonnée horizontale du point.

entV Requis. Entier qui spécifie la coordonnée verticale du point.

Exemple

L'instruction suivante définit la variable `dernièrePosition` au point (250, 400) :

```
-- Syntaxe Lingo
dernièrePosition = point(250, 400)

// Syntaxe JavaScript
var dernièrePosition = point(250, 400);
```

L'instruction suivante ajoute 5 pixels à la coordonnée horizontale du point affecté à la variable `monPoint` :

```
-- Syntaxe Lingo
monPoint.locH = monPoint.locH + 5

// Syntaxe JavaScript
monPoint.locH = monPoint.locH + 5;
```

En Lingo uniquement, les instructions suivantes définissent les coordonnées sur la scène d'une image-objet comme `mouseH` et `mouseV` plus 10 pixels. Ces deux instructions sont équivalentes.

```
-- Syntaxe Lingo
sprite(_mouse.clickOn).loc = point(_mouse.mouseH, _mouse.mouseV) \
  + point(10, 10)
sprite(_mouse.clickOn).loc = _mouse.mouseLoc + 10
```

Voir aussi

`locH`, `locV`

pointAt

Utilisation

```
member(quelActeur).model(quelModèle).pointAt\
  (positionDuVecteur{, vecteurVertical})
member(quelActeur).camera(quelleCaméra).pointAt\
  (positionDuVecteur{, vecteurVertical})
member(quelActeur).light(quelleLumière).pointAt\
  (positionDuVecteur{, vecteurVertical})
member(quelActeur).group(quelGroupe).pointAt\
  (positionDuVecteur{, vecteurVertical})
```

Description

Commande 3D ; fait pivoter l'objet référencé pour que son vecteur horizontal pointe vers une position relative à l'univers spécifiée, puis fait pivoter l'objet référencé pour que son vecteur vertical pointe dans la direction indiquée par un vecteur relatif spécifié.

Le vecteur vertical et le vecteur horizontal de l'objet sont définis par la propriété `pointAtOrientation` de l'objet.

Paramètres

positionDuVecteur Requis. Spécifie la position relative à l'univers. Cette valeur peut également être une référence de nœud.

vecteurVertical Facultatif. Spécifie un vecteur relatif à l'univers qui indique comment le vecteur vertical de l'objet devrait être orienté. Si ce paramètre n'est pas spécifié, `pointAt` utilise par défaut l'axe des y de l'univers comme vecteur vertical recommandé. Si vous essayez de diriger l'objet pour que son vecteur horizontal soit parallèle à l'axe des y de l'univers, l'axe des x de l'univers sera utilisé comme vecteur vertical recommandé. La direction horizontale de l'objet et la direction spécifiée par *vecteurVertical* n'ont pas besoin d'être perpendiculaires, car cette commande n'utilise que le paramètre *vecteurVertical* comme vecteur de recommandation.

Exemple

L'exemple suivant dirige trois objets vers le modèle Mars : la caméra camMars, la lumière Spot et le modèle Pistolet.

```
posDansCetUnivers = member("Scène").model("Mars").worldPosition
member("Scène").camera("camMars").pointAt(posDansCetUnivers)
member("Scène").light("Spot").pointAt(posDansCetUnivers)
member("Scène").model("Pistolet").pointAt(posDansCetUnivers, \
  vector(0,0,45))
```

Si vous utilisez un redimensionnement non uniforme et un `pointAtOrientation` personnalisé sur le même nœud (tel qu'un modèle), l'appel de `pointAt` entraînera probablement un redimensionnement non uniforme inattendu. Cela s'explique par l'ordre dans lequel le redimensionnement non uniforme et la rotation utilisés pour orienter correctement le nœud sont appliqués. Pour remédier à ce problème, effectuez l'une des opérations suivantes :

- Evitez d'utiliser un redimensionnement non uniforme et une valeur `pointAtOrientation` qui n'est pas celle définie par défaut sur le même nœud.
- Supprimez votre propriété d'échelle avant d'utiliser `pointAt`, puis appliquez-la ensuite à nouveau.

  Par exemple :

  ```
  scale = nœud.transform.scale
  nœud.scale = vector( 1, 1, 1 )
  nœud.pointAt(vector(0, 0, 0)) -- pointAtOrientation autre que défaut
  nœud.transform.scale = scale
  ```

Voir aussi

`pointAtOrientation`

pointInHyperlink()

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.pointInHyperlink(point)

// Syntaxe JavaScript
réfObjImageObjet.pointInHyperlink(point);
```

Description

Fonction d'image-objet texte ; renvoie TRUE ou FALSE selon que le point spécifié est ou non dans un hyperlien de l'image-objet texte. En règle générale, le point est la position du curseur. Cela est utile pour définir des curseurs personnalisés.

Paramètres

point Requis. Spécifie le point à tester.

Voir aussi

cursor(), mouseLoc

pointToChar()

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.pointToChar(pointAConvertir)

// Syntaxe JavaScript
réfObjImageObjet.pointToChar(pointAConvertir);
```

Description

Fonction ; renvoie un nombre entier représentant la position du caractère situé dans l'image-objet texte ou champ à une coordonnée d'écran spécifiée ou renvoie -1 si le point n'est pas dans le texte.

Cette fonction permet de déterminer le caractère sous le curseur.

Paramètres

pointAconvertir Requis. Spécifie la coordonnée d'écran à tester.

Exemple

Les instructions suivantes affichent le numéro du caractère cliqué, ainsi que la lettre, dans la fenêtre Messages :

```
-- Syntaxe Lingo
property spriteNum

on mouseDown me
  pointCliqué = _mouse.mouseLoc
  acteurCourant = sprite(spriteNum).member
  numDuCaractère = sprite(spriteNum).pointToChar(pointCliqué)
  caractère = acteurCourant.char[numDuCaractère]
  put("Caractère sélectionné" && numDuCaractère & ", lettre" && caractère)
end
```

```
// Syntaxe JavaScript
function mouseDown() {
  var pointCliqué = _mouse.mouseLoc;
  var acteurCourant = sprite(this.spriteNum).member;
  var numDuCaractère = sprite(this.spriteNum).pointToChar(pointCliqué);
  var caractère = acteurCourant.getProp("char", numDuCaractère);
  put ("Caractère sélectionné " + numDuCaractère + ", lettre " + caractère);
}
```

Voir aussi

mouseLoc, pointToWord(), pointToItem(), pointToLine(), pointToParagraph()

pointToItem()

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.pointToItem(pointAConvertir)

// Syntaxe JavaScript
réfObjImageObjet.pointToItem(pointAConvertir);
```

Description

Fonction ; renvoie un nombre entier représentant la position de l'élément situé dans l'image-objet texte ou champ à une coordonnée d'écran spécifiée ou renvoie -1 si le point n'est pas dans le texte. Les éléments sont séparés par la propriété `itemDelimiter`, qui est par défaut une virgule.

Cette fonction permet de déterminer l'élément sous le curseur.

Paramètres

pointAconvertir Requis. Spécifie la coordonnée d'écran à tester.

Exemple

Les instructions suivantes affichent le numéro de l'élément cliqué, ainsi que son texte, dans la fenêtre Messages :

```
-- Syntaxe Lingo
property spriteNum

on mouseDown me
  pointCliqué = _mouse.mouseLoc
  acteurCourant = sprite(spriteNum).member
  numéroDélément = sprite(spriteNum).pointToItem(pointCliqué)
  texteDélément = acteurCourant.item[numéroDélément]
  put("Elément cliqué" && numéroDélément & ", le texte" && texteDélément)
end
```

```
// Syntaxe JavaScript
function mouseDown() {
  var pointCliqué = _mouse.mouseLoc;
  var acteurCourant = sprite(this.spriteNum).member;
  var numéroDélément = sprite(this.spriteNum).pointToItem(pointCliqué);
  var texteDélément = acteurCourant.getProp("item" ,numéroDélément);
  trace("Elément cliqué " + numéroDélément + ", le texte" + texteDélément);
}
```

Voir aussi

itemDelimiter, mouseLoc, pointToChar(), pointToWord(), pointToItem(), pointToLine(), pointToParagraph()

pointToLine()

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.pointToLine(pointAConvertir)

// Syntaxe JavaScript
réfObjImageObjet.pointToLine(pointAConvertir);
```

Description

Fonction ; renvoie un nombre entier représentant la position de la ligne dans l'image-objet texte ou champ à une coordonnée d'écran spécifiée ou renvoie -1 si le point n'est pas dans le texte. Les lignes sont séparées par des retours chariot dans l'acteur texte ou champ.

Cette fonction permet de déterminer la ligne sous le curseur.

Paramètres

pointAconvertir Requis. Spécifie la coordonnée d'écran à tester.

Exemple

Les instructions suivantes affichent le numéro de la ligne cliquée, ainsi que son texte, dans la fenêtre Messages :

```
-- Syntaxe Lingo
property spriteNum

on mouseDown me
  pointCliqué = _mouse.mouseLoc
  acteurCourant = sprite(spriteNum).member
  numDeLigne = sprite(spriteNum).pointToLine(pointCliqué)
  texteDeLigne = acteurCourant.line[numDeLigne]
  put("Ligne cliquée" && numDeLigne & ", le texte" && texteDeLigne)
end
```

```
// Syntaxe JavaScript
function mouseDown() {
  var pointCliqué = _mouse.mouseLoc;
  var acteurCourant = sprite(this.spriteNum).member;
  var numDeLigne = sprite(this.spriteNum).pointToLine(pointCliqué);
  var texteDeLigne = acteurCourant.getProp("line", numDeLigne);
  put("Ligne cliquée" + numDeLigne + ", le texte" + texteDeLigne);
}
```

Voir aussi

itemDelimiter, mouseLoc, pointToChar(), pointToWord(), pointToItem(), pointToLine(), pointToParagraph()

pointToParagraph()

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.pointToParagraph(PointAConvertir)

// Syntaxe JavaScript
réfObjImageObjet.pointToParagraph(PointAConvertir);
```

Description

Fonction ; renvoie un nombre entier représentant le numéro du paragraphe situé dans l'image-objet texte ou champ à une coordonnée d'écran spécifiée ou renvoie -1 si le point n'est pas dans le texte. Les paragraphes sont séparés par des retours chariot dans un bloc de texte.

Cette fonction permet de déterminer le paragraphe sous le curseur.

Paramètres

pointAconvertir Requis. Spécifie la coordonnée d'écran à tester.

Exemple

Les instructions suivantes affichent le numéro du paragraphe cliqué, ainsi que son texte, dans la fenêtre Messages :

```
-- Syntaxe Lingo
property spriteNum

on mouseDown me
  pointCliqué = _mouse.mouseLoc
  acteurCourant = sprite(spriteNum).member
  numDeParagraphe = sprite(spriteNum).pointToParagraph(pointCliqué)
  texteDuParagraphe = acteurCourant.paragraph[numDeParagraphe]
  put("Paragraphe cliqué" && numDeParagraphe & ", le texte" &&
  texteDuParagraphe)
end
```

```
// Syntaxe JavaScript
function mouseDown() {
  var pointCliqué = _mouse.mouseLoc;
  var acteurCourant = sprite(this.spriteNum).member;
  var numDeParagraphe = sprite(this.spriteNum).pointToParagraph(pointCliqué);
  var texteDuParagraphe= acteurCourant.getProp("paragraph", numDeParagraphe);
  trace("Paragraphe cliqué " + numDeParagraphe + ", le texte" +
  texteDuParagraphe);
}
```

Voir aussi

itemDelimiter, mouseLoc, pointToChar(), pointToWord(), pointToItem(), pointToLine()

pointToWord()

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.pointToWord(pointAConvertir)

// Syntaxe JavaScript
réfObjImageObjet.pointToWord(pointAConvertir);
```

Description

Fonction ; renvoie un nombre entier représentant le numéro du mot situé dans l'image-objet texte ou champ à une coordonnée d'écran spécifiée ou renvoie -1 si le point n'est pas dans le texte. Les mots sont séparés par des espaces dans un bloc de texte.

Cette fonction permet de déterminer le mot sous le curseur.

Paramètres

pointAconvertir Requis. Spécifie la coordonnée d'écran à tester.

Exemple

Les instructions suivantes affichent le numéro du mot cliqué, ainsi que son texte, dans la fenêtre Messages :

```
-- Syntaxe Lingo
property spriteNum

on mouseDown me
  pointCliqué = _mouse.mouseLoc
  acteurCourant = sprite(spriteNum).member
  numDuMot = sprite(spriteNum).pointToWord(pointCliqué)
  texteDuMot = acteurCourant.word[numDuMot]
  put("Mot cliqué" && numDuMot & ", le texte" && texteDuMot)
end
```

```
// Syntaxe JavaScript
function mouseDown(me) {
  var pointCliqué = _mouse.mouseLoc;
  var acteurCourant = sprite(this.spriteNum).member;
  var numDuMot = sprite(this.spriteNum).pointToWord(pointCliqué);
  var texteDuMot = acteurCourant.getProp("word", numDuMot);
  trace("Mot cliqué" + numDuMot + ", le texte" + texteDuMot);
}
```

Voir aussi

itemDelimiter, mouseLoc, pointToChar(), pointToItem(), pointToLine(), pointToParagraph()

postNetText

Utilisation

```
postNetText(url, listeDePropriétés {,chaîneOSduServeur}
  {,chaîneJeuCarServeur})
postNetText(url, textePosté {,chaîneOSduServeur} {,chaîneJeuCarServeur})
```

Description

Commande ; envoie une requête POST à une URL (une URL HTTP) avec des données spécifiées.

Cette commande est similaire à getNetText(). Comme pour getNetText(), la réponse du serveur est renvoyée via netTextResult(*IDRéseau*) une fois que netDone(*IDRéseau*) reçoit la valeur 1 et si netError(*IDRéseau*) a la valeur 0 ou OK.

Les arguments facultatifs peuvent être omis quelle que soit leur position.

Cette commande possède en outre un avantage supplémentaire par rapport à getNetText() : en effet, une requête postNetText() peut être arbitrairement longue, alors que la longueur de la requête getNetText() est limitée à celle d'une adresse URL (1 Ko ou 4 Ko, selon le navigateur web).

Remarque : Si vous utilisez postNetText pour afficher des données dans un domaine différent de celui dans lequel l'animation est lue, une alerte de sécurité est déclenchée lors de la lecture dans Shockwave Player.

Pour un exemple d'utilisation de postNetText dans une animation, reportez-vous à l'animation Forms and Post du dossier Learning/Lingo, lui-même inclus dans le dossier de Director.

Paramètres

url Requis. Spécifie l'URL à laquelle la requête POST doit être envoyée.

listeDePropriétés ou *TextePosté* Requis. Spécifie les données à envoyer avec la demande. Lorsqu'une liste de propriétés est utilisée à la place d'une chaîne, les informations sont envoyées avec METHOD=POST, de la même manière qu'un navigateur web affiche un formulaire HTML. Cette procédure facilite la construction et l'affichage des données d'un formulaire dans un titre Director. Les noms des propriétés correspondent aux noms des champs du formulaire HTML et leurs valeurs à celles des champs.

La liste de propriétés peut utiliser des chaînes ou des symboles comme noms de propriétés. Si un symbole est utilisé, il est automatiquement converti en chaîne sans le signe # du début. De même, les valeurs numériques sont converties en chaînes si elles sont utilisées comme valeur d'une propriété.

Remarque : Si la forme secondaire est utilisée (une chaîne remplace la liste de propriétés), la chaîne *textePosté* est envoyée au serveur sous forme de requête HTTP POST en utilisant le type mime text/plain. Bien que pratique dans certaines applications, cette méthode n'est pas compatible avec l'affichage de formulaires HTML.

chaîneOSduServeur Facultatif. La valeur par défaut est `UNIX`, mais ce paramètre peut prendre la valeur `Windows` ou `Mac` et convertit les retours chariot de l'argument *listeDePropriétés* dans le format utilisé par le serveur afin d'éviter toute confusion. Pour la plupart des applications, ce paramètre n'est pas nécessaire, les ruptures de ligne n'étant généralement pas utilisées dans les réponses de formulaires.

chaîneJeuCarServeur Facultatif. Ce paramère ne s'applique que si l'utilisateur travaille sur un système Shift-JIS (japonais). Ses valeurs possibles sont `JIS`, `EUC`, `ASCII` et `AUTO`. Les données envoyées sont converties de Shift-JIS dans le jeu de caractères désigné. Les données renvoyées sont traitées de la même façon qu'avec `getNetText()` (converties du jeu de caractères nommé à Shift-JIS). Si `AUTO` est utilisé, les données affichées dans le jeu de caractères local ne sont pas converties ; les résultats renvoyés par le serveur sont convertis comme pour `getNetText()`. `"ASCII"` est la valeur par défaut si *chaîneJeuCarServeur* est omis. `ASCII` n'offre aucune conversion pour l'envoi ou les résultats.

Exemple

L'instruction suivante omet le paramètre *chaîneJeuCarServeur* :

```
IDréseau = postNetText("www.dom.fr\baseDeDonnées.cgi", "Jean Martin", "Win")
```

L'exemple suivant génère un formulaire à partir des champs de saisie de l'utilisateur pour son prénom et son nom, ainsi qu'un score. Notez que *chaîneOSduServeur* et *chaîneJeuCarServeur* ont été omis :

```
nom = member("Nom").text
prénom = member("Prénom").text
score = member("Score").text
listeInfos = ["Prénom":prénom, "Nom":nom, "Score":score]
IDréseau = postNetText("www.mondomaine.fr\baseUtilisateurs.cgi", listeInfos)
```

Voir aussi

`getNetText()`, `netTextResult()`, `netDone()`, `netError()`

power()

Utilisation

`power(`*base*`, `*exposant*`)`

Description

Fonction mathématique ; calcule la valeur d'un nombre déterminé qui est élevé à une puissance donnée.

Paramètres

base Requis. Spécifie le numéro de base.

exposant Requis. Spécifie la valeur de l'exposant.

Exemple

L'instruction suivante donne à la variable `vRésultat` la valeur du cube de 4 :

```
set vRésultat = power(4,3)
```

preLoad() (acteur)

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.preLoad({réfObjDernierActeur})

// Syntaxe JavaScript
réfObjActeur.preLoad({réfObjDernierActeur});
```

Description

Méthode d'acteur ; précharge un acteur ou une plage d'acteurs en mémoire et arrête le préchargement lorsque la mémoire est pleine ou lorsque tous les acteurs spécifiés ont été préchargés.

En l'absence du paramètre *réfObjDernierActeur*, `preLoad()` précharge tous les acteurs utilisés depuis l'image courante jusqu'à la dernière image d'une animation.

Paramètres

réfObjDernierActeur Facultatif. Référence au dernier acteur d'une plage d'acteurs qui est chargé en mémoire. Le premier acteur de la plage est spécifié par *réfObjActeur*.

Exemple

L'instruction suivante indique dans la fenêtre Messages si l'animation QuickTime Chaise pivotante peut être préchargée en mémoire :

```
-- Syntaxe Lingo
put(member("Chaise pivotante").preload())

// Syntaxe JavaScript
put(member("Chaise pivotante").preload());
```

Le gestionnaire `startMovie` suivant définit un acteur animation Flash pour une lecture en flux continu puis définit sa propriété `bufferSize` :

```
-- Syntaxe Lingo
on startMovie
  member("Démo Flash").preload = FALSE
  member("Démo Flash").bufferSize = 65536
end

// Syntaxe JavaScript
function startMovie() {
  member("Démo Flash").preload = false;
  member("Démo Flash").bufferSize = 65536;
}
```

Voir aussi

Acteur

preLoad() (animation)

Utilisation

```
-- Syntaxe Lingo
_movie.preLoad({nomOuNumDimage})
_movie.preLoad(nomOuNumDimageInitiale, nomOuNumDimageFinale)

// Syntaxe JavaScript
_movie.preLoad({nomOuNumDimage});
_movie.preLoad(nomOuNumDimageInitiale, nomOuNumDimageFinale);
```

Description

Méthode d'animation ; précharge en mémoire des acteurs de l'image ou de la plage d'images spécifiée et s'arrête lorsque la mémoire disponible est saturée ou que tous les acteurs spécifiés ont été préchargés, comme suit :

- En l'absence d'arguments, cette méthode précharge tous les acteurs utilisés depuis l'image courante jusqu'à la dernière image d'une animation.
- Si un seul argument est fourni (*nomOuNumDimage*), la méthode précharge tous les acteurs utilisés dans la plage d'images depuis l'image courante jusqu'à l'image *nomOuNumDimage*, comme spécifié par le numéro ou le nom de l'image.
- Si deux arguments sont fournis (*nomOuNumDimageInitiale* et *nomOuNumDimageFinale*) la méthode précharge tous les acteurs utilisés dans la plage d'images depuis l'image *nomOuNumDimageInitiale* jusqu'à l'image *nomOuNumDimageFinale*, comme spécifié par le numéro ou le nom de l'image.

La méthode `preLoad()` renvoie aussi le numéro de la dernière image qu'il a été possible de charger. Pour obtenir cette valeur, utilisez la fonction `result()`.

Paramètres

nomOuNumDimage Facultatif. Chaîne qui spécifie l'image à précharger ou nombre entier qui spécifie son numéro.

nomOuNumDimageInitiale Requis si vous préchargez une plage d'images. Chaîne qui spécifie le nom du libellé de la première image dans la plage d'images à précharger ou nombre entier qui spécifie le numéro de cette image.

nomOuNumDimageFinale Requis si vous préchargez une plage d'images. Chaîne qui spécifie le nom du libellé de la dernière image dans la plage d'images à précharger ou nombre entier qui spécifie le numéro de cette image.

Exemple

L'instruction suivante précharge les acteurs utilisés depuis l'image courante jusqu'à l'image contenant le repère suivant :

```
-- Syntaxe Lingo
_movie.preLoad(_movie.marker(1))

// Syntaxe JavaScript
_movie.preLoad(_movie.marker(1));
```

L'instruction suivante précharge les acteurs utilisés de l'image 10 à l'image 50 :

```
-- Syntaxe Lingo
_movie.preLoad(10, 50)

// Syntaxe JavaScript
_movie.preLoad(10, 50);
```

Voir aussi

Animation, result

preLoadBuffer()

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.preLoadBuffer()

// Syntaxe JavaScript
réfObjActeur.preLoadBuffer();
```

Description

Commande ; précharge une partie d'un fichier Shockwave Audio (SWA) spécifié en mémoire. La quantité préchargée est déterminée par la propriété `preLoadTime`. Cette commande ne fonctionne que si l'acteur SWA est arrêté.

Une fois la commande `preLoadBuffer` réussie, la propriété d'acteur `state` est égale à 2.

La plupart des propriétés des acteurs SWA ne peuvent être testées qu'après la réussite complète de la commande `preLoadBuffer`. Ces propriétés sont : `cuePointNames`, `cuePointTimes`, `currentTime`, `duration`, `percentPlayed`, `percentStreamed`, `bitRate`, `sampleRate` et `numChannels`.

Paramètres

Aucun.

Exemple

L'instruction suivante charge l'acteur Jacques Brel en mémoire :

```
-- Syntaxe Lingo
member("Jacques Brel").preLoadBuffer()

// Syntaxe JavaScript
member("Jacques Brel").preLoadBuffer();
```

Voir aussi

preLoadTime

preLoadMember()

Utilisation

```
-- Syntaxe Lingo
_movie.preLoadMember({réfObjActeur})
_movie.preLoadMember(nomOuNumDacteurInitial, nomOuNumDacteurFinal)

// Syntaxe JavaScript
_movie.preLoadMember({réfObjActeur});
_movie.preLoadMember(nomOuNumDacteurInitial, nomOuNumDacteurFinal);
```

Description

Méthode d'animation ; précharge les acteurs et arrête le préchargement lorsque la mémoire est saturée ou que tous les acteurs spécifiés ont été préchargés.

Cette méthode renvoie le numéro du dernier acteur qu'il a été possible de charger. Pour obtenir cette valeur, utilisez la fonction `result()`.

En l'absence d'arguments, `preLoadMember()` précharge tous les acteurs de l'animation.

Si l'argument *`réfObjActeur`* est fourni, `preLoadMember()` ne précharge que cet acteur. Si *`réfObjActeur`* est un nombre entier, il n'est fait référence qu'à la première distribution. Si *`réfObjActeur`* est une chaîne, le premier acteur dont le nom correspond à la chaîne est utilisé.

Si les deux arguments (*`nomOuNumDacteurInitial`* et *`nomOuNumDacteurFinal`*) sont fournis, `preLoadMember()` précharge tous les acteurs dans la plage spécifiée par les noms ou numéros des acteurs.

Paramètres

`réfObjActeur` Facultatif. Référence à l'acteur à précharger.

`nomOuNumDacteurInitial` Requis si vous préchargez une plage d'acteurs. Chaîne ou nombre entier qui spécifie le premier acteur de la plage d'acteurs à précharger.

`nomOuNumDacteurFinal` Requis si vous préchargez une plage d'acteurs. Chaîne ou nombre entier qui spécifie le dernier acteur de la plage d'acteurs à précharger.

Voir aussi

`Animation`, `preLoad() (acteur)`, `result`

preLoadMovie()

Utilisation

```
-- Syntaxe Lingo
_movie.preLoadMovie(chaîneNomDanimation)

// Syntaxe JavaScript
_movie.preLoadMovie(chaîneNomDanimation);
```

Description

Méthode d'animation ; précharge les données et les acteurs associés à la première image de l'animation spécifiée. Le préchargement d'une animation permet de la faire démarrer plus rapidement avec les méthodes `go()` ou `play()`.

Si vous devez précharger des acteurs depuis une adresse URL, utilisez `preloadNetThing()` pour charger les acteurs directement dans la mémoire cache ou `downloadNetThing()` pour charger une animation sur un disque local, à partir duquel vous pouvez alors la charger en mémoire, de façon à réduire le temps de téléchargement.

Paramètres

`chaîneNomDanimation` Requis. Chaîne qui spécifie le nom de l'animation à précharger.

Exemple

L'instruction suivante précharge l'animation Introduction, qui est située dans le même dossier que l'animation courante :

```
-- Syntaxe Lingo
_movie.preLoadMovie("Introduction")

// Syntaxe JavaScript
_movie.preLoadMovie("Introduction");
```

Voir aussi

`downloadNetThing`, `go()`, `Animation`, `preloadNetThing()`

preloadNetThing()

Utilisation

`preloadNetThing (`*`url`*`)`

Description

Fonction ; précharge un fichier depuis Internet dans la mémoire cache locale afin de le rendre utilisable par la suite sans perte de temps inhérente au téléchargement. La valeur renvoyée est un identifiant réseau à utiliser pour suivre le déroulement de l'opération.

La fonction `preloadNetThing()` télécharge le fichier pendant la lecture de l'animation courante. Utilisez `netDone()` pour vérifier si le téléchargement est terminé.

Un élément téléchargé peut être immédiatement affiché, car il est lu à partir de la mémoire cache locale et non à partir du réseau.

Bien que de nombreuses opérations réseau puissent être actives au même moment, l'exécution de plus de quatre opérations simultanées réduit considérablement les performances.

La taille de la mémoire cache et l'option de vérification des documents des préférences d'un navigateur web n'affectent pas le comportement de la commande `preloadNetThing`.

La fonction `preloadNetThing()` n'analyse pas les liens d'un fichier Director. En conséquence, même si un fichier Director est lié aux fichiers de distribution et aux fichiers graphiques, `preloadNetThing()` ne télécharge que le fichier Director. Vous devez toujours précharger séparément les autres objets liés.

Paramètres

`url` Requis. Spécifie le nom d'un fichier Internet valide tel qu'une animation Director, un graphique ou l'adresse d'un serveur FTP.

Exemple

L'instruction suivante utilise `preloadNetThing()` et renvoie l'ID réseau pour l'opération :

```
set monIDréseau = preloadNetThing("http://www.votreServeur.fr/pageDeMenu/
  monAnimation.dir")
```

Une fois le téléchargement terminé, vous pouvez naviguer jusqu'à l'animation avec la même adresse URL. L'animation sera lue depuis la mémoire cache, et non depuis l'adresse URL, puisqu'elle est chargée.

Voir aussi

`netDone()`

preMultiply

Utilisation

`transformation1``.preMultiply(`*`transformation2`*`)`

Description

Commande 3D de transformation ; modifie une transformation en appliquant les effets de positionnement, de rotation et de redimensionnement d'une autre transformation.

Si *`transformation2`* décrit une rotation de 90° autour de l'axe des x et que *`transformation1`* décrit une translation de 100 unités sur l'axe des y, `transformation1.multiply(`*`transformation2`*`)` modifie cette transformation pour lui faire décrire une translation suivie d'une rotation. L'instruction `transformation1.preMultiply(`*`transformation2`*`)` modifie cette transformation pour lui faire décrire une rotation suivie d'une translation. L'effet obtenu est l'inversement de l'ordre des opérations.

Paramètres

`transformation2` Requis. Spécifie la transformation à partir de laquelle les effets sont pré-appliqués à une autre transformation.

Exemple

L'instruction suivante exécute un calcul qui applique la transformation du modèle Mars à la transformation du modèle Pluton :

```
member("scène").model("Pluton").transform.preMultiply\
  (member("scène").model("Mars").transform)
```

preRotate

Utilisation

```
référenceDeTransformation.preRotate( angleX, angleY, angleZ )
référenceDeTransformation.preRotate(vecteur)
référenceDeTransformation.preRotate( vecteurDePosition, vecteurDeDirection, \
  angle )
member( quelActeur ).nœud.transform.preRotate( angleX, \
  angleY, angleZ )
member( quelActeur ).nœud.transform.preRotate( vecteur )
member( quelActeur ).nœud.transform.preRotate\
  ( vecteurDePosition, vecteurDeDirection, angle )
```

Description

Commande 3D de transformation ; applique une rotation avant les décalages de position, rotation et d'échelle de la transformation. La rotation peut être spécifiée sous la forme d'un ensemble de trois angles, chacun desquels spécifiant l'angle de rotation autour des trois axes correspondants. Ces angles peuvent être spécifiés de façon explicite sous la forme `angleX`, `angleY` et `angleZ`, ou au moyen d'un vecteur, dans lequel les composants x, y et z correspondent, respectivement, à la rotation autour de l'axe des *x*, des *y* et des *z*.

La rotation peut également être spécifiée comme une rotation autour d'un axe arbitraire. Cet axe est défini dans l'espace par *vecteurDePosition* et *vecteurDeDirection*. Le degré de rotation autour de cet axe est spécifié par *angle*.

nœud peut être une référence à un modèle, un groupe, une lumière ou une caméra.

Paramètres

angleX Requis si vous appliquez une rotation avec les axes des *x*, des *y* et des *z*. Spécifie l'angle de rotation autour de l'axe des *x*.

angleY Requis si vous appliquez une rotation avec les axes des *x*, des *y* et des *z*. Spécifie l'angle de rotation autour de l'axe des *y*.

angleZ Requis si vous appliquez une rotation avec les axes des *x*, des *y* et des *z*. Spécifie l'angle de rotation autour de l'axe des *z*.

vecteur Requis si vous appliquez une rotation avec un vecteur. Spécifie le vecteur dont les angles sont utilisés dans la rotation.

vecteurDePosition Requis si vous appliquez une rotation autour d'un axe arbitraire. Spécifie le décalage de la position.

vecteurDeDirection Requis si vous appliquez une rotation autour d'un axe arbitraire. Spécifie le décalage de la direction.

angle Requis si vous appliquez une rotation autour d'un axe arbitraire. Spécifie le degré de rotation autour d'un axe arbitraire.

Exemple

L'instruction suivante effectue une rotation de 20° sur chaque axe. Dans la mesure où la propriété `transform` du modèle correspond à ses décalages de position, rotation et redimensionnement par rapport à son parent et que `preRotate` applique la modification d'orientation avant tout effet existant de la propriété `transform` de ce modèle, ce dernier subira une rotation sur place plutôt qu'autour de son parent.

```
member("Scène").model("bip01").transform.preRotate(20, 20, 20)
```

L'instruction précédente équivaut à la suivante :

```
member("Scène").model("bip01").rotate(20,20,20).
```

`preRotate()` n'est généralement utile qu'avec les variables de transformation. Cette ligne fera orbiter la caméra autour du point (100, 0, 0) dans l'espace, de 180° autour de l'axe des y.

```
t = transform()
t.position = member("scène").camera[1].transform.position
t.preRotate(vector(100, 0, 0), vector(0, 1, 0), 180)
member("scène").camera[1].transform = t
```

Voir aussi

`rotate`

preScale()

Utilisation

```
référenceDeTransformation.preScale(échelleX , échelleY , échelleZ)
référenceDeTransformation.preScale(vecteur)
member( quelActeur ).nœud.transform.preScale( échelleX, \
  échelleY, échelleZ )
member( quelActeur ).nœud.transform.preScale( vecteur )
```

Description

Commande 3D de transformation ; applique une échelle avant d'appliquer les effets de position, de rotation et de redimensionnement existants de la transformation en question.

nœud peut être une référence à un modèle, un groupe, une lumière ou une caméra.

Paramètres

échelleX Requis si vous appliquez une échelle avec les axes des *x*, des *y* et des *z*. Spécifie l'échelle sur l'axe des *x*.

échelleY Requis si vous appliquez une échelle avec les axes des *x*, des *y* et des *z*. Spécifie l'échelle sur l'axe des *y*.

échelleZ Requis si vous appliquez une échelle avec les axes des *x*, des *y* et des *z*. Spécifie l'échelle sur l'axe des *z*.

vecteur Requis si vous appliquez une échelle avec un vecteur. Spécifie le vecteur qui contient l'échelle à appliquer.

Exemple

La **ligne 1** du code Lingo suivant crée un double de la transformation de Lune1. N'oubliez pas qu'on accède à la propriété de transformation d'un modèle par référence.

La **ligne 2** applique une échelle à cette transformation avant tout effet existant de position ou de rotation. Supposez que la transformation représente le décalage de position et l'orbite de rotation de Lune1 par rapport à sa planète parent. Supposez également que le parent de Lune2 est le même que celui de Lune1. Si nous utilisions ici `scale()` au lieu de `preScale()`, Lune2 serait placé deux fois plus loin et tournerait autour de la planète deux fois plus que Lune1. Cela s'explique par le fait que le redimensionnement serait appliqué aux décalages de position et de rotation existants de la transformation. L'utilisation de `preScale()` appliquera la modification de taille sans affecter les décalages de position et de rotation existants.

La **ligne 3** applique une rotation supplémentaire de 180° autour de l'axe des *x* de la planète. Cela placera Lune2 du côté opposé à l'orbite de Lune1. Veuillez noter qu'avec `preRotate()`, Lune2 serait resté à la même place que Lune1 et aurait été pivoté de 180° autour de son propre axe des *x*.

La **ligne 4** affecte cette nouvelle transformation à Lune2.

```
t = member("scène").model("Lune1").transform.duplicate()
t.preScale(2,2,2)
t.rotate(180,0,0)
member("scène").model("Lune2").transform = t
```

preTranslate()

Utilisation

```
référenceDeTransformation.preTranslate( incrémentX, incrémentY, \
  incrémentZ )
référenceDeTransformation.preTranslate(vecteur)
member( quelActeur ).nœud.transform.preTranslate\
  (incrémentX, incrémentY, incrémentZ)
member( quelActeur ).nœud.transform.preTranslate( vecteur )
```

Description

Commande 3D de transformation ; applique une translation avant les décalages de position, rotation et d'échelle de la transformation. La translation peut être spécifiée sous la forme d'un jeu de trois incréments le long des trois axes correspondants. Ces incréments peuvent être spécifiés explicitement sous la forme *incrémentX*, *incrémentY* et *incrémentZ*, ou par un vecteur, dont le composant x correspond à la translation sur l'axe des x, y sur l'axe des y et z sur l'axe des y.

A la suite d'une série de transformations, réalisées dans l'ordre suivant, l'origine locale du modèle se trouvera à (0, 0, -100), si le parent du modèle est l'univers :

```
model.transform.identity()
model.transform.rotate(0, 90, 0)
model.transform.preTranslate(100, 0, 0)
```

Si `translate()` avait été utilisée à la place de `preTranslate()`, l'origine locale du modèle aurait été (100, 0, 0) et le modèle aurait été pivoté de 90° autour de son propre axe des y. L'instruction `model.transform.pretranslate(x, y, z)` est équivalente à `model.translate(x, y, z)`. La propriété `preTranslate()` n'est généralement utile qu'avec les variables de transformation plutôt qu'avec les références `model.transform`.

Paramètres

incrémentX Requis si vous appliquez une transition avec les axes des *x*, des *y* et des *z*. Spécifie la translation autour de l'axe des *x*.

incrémentY Requis si vous appliquez une transition avec les axes des *x*, des *y* et des *z*. Spécifie la translation autour de l'axe des *y*.

incrémentZ Requis si vous appliquez une transition avec les axes des *x*, des *y* et des *z*. Spécifie la translation autour de l'axe des *z*.

vecteur Requis si vous appliquez une translation avec un vecteur. Spécifie le vecteur à utiliser dans la translation.

Exemple

```
t = transform()
t.transform.identity()
t.transform.rotate(0, 90, 0)
t.transform.preTranslate(100, 0, 0)
gbModèle = member("Scène").model("Mars")
gbModel.transform = t
put gbModèle.transform.position
-- vector(0.0000, 1.0000, 0.0000)
```

print()

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.print({nomDeCible, #printingBounds})

// Syntaxe JavaScript
réfObjImageObjet.print({nomDeCible, #printingBounds});
```

Description

Commande ; appelle la commande ActionScript `print` correspondante, qui a fait son apparition avec Flash 5. Toutes les images de l'animation Flash identifiées par `#p` sont imprimées. Si aucune image individuelle n'a été libellée, toute l'animation est imprimée.

L'impression d'animations Flash étant une opération relativement complexe, il est recommandé de lire la section consacrée à l'impression dans la documentation de Flash avant d'utiliser cette fonction d'image-objet.

Paramètres

nomDeCible Facultatif. Spécifie le nom de l'animation ou du clip d'animation cible à imprimer. Si ce paramètre est omis (si la cible a la valeur 0), l'animation Flash principale est imprimée.

printingBounds Facultatif. Spécifie les options d'impression des limites. Si ce paramètre est omis, les limites de la cible sont utilisées. S'il est spécifié, *printingBounds* doit avoir l'une des valeurs suivantes :

- `#bframe`. Avec #bframe, les limites d'impression de chaque image sont modifiées en fonction de l'image à imprimer.
- `#bmax`. Avec #bmax, les limites d'impression forment un cadre virtuel de dimensions suffisantes pour contenir toutes les images à imprimer.

printAsBitmap()

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.printAsBitmap({nomDeCible, #printingBounds})

// Syntaxe JavaScript
réfObjImageObjet.printAsBitmap({nomDeCible, #printingBounds});
```

Description

Commande d'image-objet Flash ; fonctionne comme la commande `print`, mais uniquement avec des images-objets Flash. Cependant, la commande `printAsBitmap` peut être utilisée pour imprimer des objets contenant des informations alpha.

printFrom()

Utilisation

```
-- Syntaxe Lingo
_movie.printFrom(nomOuNumDimageInitiale {, nomOuNumDimageFinale, redux})

// Syntaxe JavaScript
_movie.printFrom(nomOuNumDimageInitiale {, nomOuNumDimageFinale, redux});
```

Description

Méthode d'animation ; imprime tout le contenu de chaque image de la scène, que l'image soit sélectionnée ou non, à partir de l'image spécifiée par *nomOuNumDimageInitiale*. Vous pouvez, si vous le voulez, indiquer *nomOuNumDimageFinale* et la valeur de réduction (*redux*) : 100 %, 50 % ou 25 %.

L'image en cours d'impression n'a pas besoin d'être affichée. Cette commande imprime toujours en orientation portrait (verticale) et à une résolution de 72 points par pouce, en transformant en bitmaps tout le contenu de l'écran (ce qui peut parfois réduire la qualité du texte) ; de plus, elle ignore les paramètres de format d'impression. Pour augmenter la souplesse de l'impression depuis Director, consultez l'Xtra PrintOMatic Lite, qui se trouve sur le disque d'installation.

Paramètres

nomOuNumDimageInitiale Requis. Chaîne ou numéro qui spécifie le nom ou le numéro de la première image à imprimer.

nomOuNumDimageFinale Facultatif. Chaîne ou numéro qui spécifie le nom ou le numéro de la dernière image à imprimer.

redux Facultatif. Nombre entier qui spécifie la valeur de la réduction. Les valeurs correctes sont 100, 50 et 25.

Exemple

L'instruction suivante imprime le contenu de la scène à l'image 1 :

```
-- Syntaxe Lingo
_movie.printFrom(1)

// Syntaxe JavaScript
_movie.printFrom(1);
```

L'instruction suivante imprime le contenu de la scène dans chaque image de l'image 10 à l'image 25. La valeur de réduction est de 50 %.

```
-- Syntaxe Lingo
_movie.printFrom(10, 25, 50)

// Syntaxe JavaScript
_movie.printFrom(10, 25, 50);
```

Voir aussi

Animation

propList()

Utilisation

```
-- Syntaxe Lingo
propList()
[:]
propList(chaîne1, valeur1, chaîne2, valeur2, ...)
propList(#symbole1, valeur1, #symbole2, valeur2, ...)
[#symbole1:valeur1, #symbole2:valeur2, ...]

// Syntaxe JavaScript
propList();
propList(chaîne1, valeur1, chaîne2, valeur2, ...);
```

Description

Fonction du niveau supérieur ; crée une liste de propriétés dont chaque élément est constitué d'une paire nom/valeur.

Lorsque vous créez une liste en utilisant la syntaxe `propList()` ou `[:]` (Lingo uniquement), avec ou sans paramètres, les valeurs de la liste sont indexées à partir de 1.

La longueur maximale d'une ligne de script exécutable est de 256 caractères. Il n'est pas possible de créer des listes de propriétés importantes avec `propList()`. Pour créer une liste de propriétés contenant une grande quantité de données, mettez les données entre crochets (`[]`), placez-les dans un champ, puis affectez le champ à une variable. Le contenu de cette variable deviendra une liste des données.

Paramètres

chaîne1, *chaîne2*, . . . Facultatif. Chaînes qui spécifient les parties nom des éléments inclus dans la liste.

valeur1, *valeur2*, . . . Facultatif. Valeurs qui spécifient les parties valeur des éléments inclus dans la liste.

#symbole1, *#symbole2*, . . . (Lingo uniquement) Facultatif. Symboles qui représentent les parties nom des éléments inclus dans la liste.

Exemple

Cette instruction crée une liste de propriétés avec diverses propriétés et valeurs, puis affiche les diverses valeurs de propriétés dans la fenêtre Messages :

```
-- Syntaxe Lingo
-- avec propList()
colorList = propList(#top,"rouge", #sides,"bleu", #bottom,"vert")
-- avec des crochets
colorList = [#top:"rouge", #sides:"bleu", #bottom:"vert"]
put(colorList.top) -- "rouge"
put(colorList.sides) -- "bleu"
put(colorList.bottom) -- "vert"

// Syntaxe JavaScript
var colorList = propList("top","rouge", "sides","bleu", "bottom","vert");
put(colorList.top) // rouge
put(colorList.sides) // bleu
put(colorList.bottom) // vert
```

Voir aussi

`list()`

proxyServer

Utilisation

```
proxyServer typeDeServeur, "adresseIP", numDePort
proxyServer()
```

Description

Commande ; définit les valeurs d'un serveur proxy FTP ou http.

En l'absence de paramètres, `proxyServer()` renvoie les réglages d'un serveur proxy FTP ou HTTP.

Paramètres

`typeDeServeur` Facultatif. Symbole qui spécifie le type du serveur proxy. La valeur peut être `#ftp` ou `#http`.

`adresseIP` Facultatif. Chaîne qui spécifie l'adresse IP.

`numDePort` Facultatif. Nombre entier qui spécifie le numéro du port.

Exemple

L'instruction suivante établit un serveur proxy HTTP à l'adresse IP 197.65.208.157 avec le port 5 :

```
proxyServer #http,"197.65.208.157",5
```

L'instruction suivante renvoie le numéro de port d'un serveur proxy HTTP :

```
put proxyServer(#http,#port)
```

Si aucun type de serveur n'est spécifié, la fonction renvoie 1.

L'instruction suivante renvoie la chaîne de l'adresse IP d'un serveur proxy HTTP :

```
put proxyServer(#http)
```

L'instruction suivante désactive un serveur proxy FTP :

```
proxyServer #ftp,#stop
```

ptToHotSpotID()

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.ptToHotSpotID(point)

// Syntaxe JavaScript
réfObjImageObjet.ptToHotSpotID(point);
```

Description

Fonction QuickTime VR ; renvoie l'identifiant de la zone référencée (si elle existe) présent au point spécifié. S'il n'existe pas de zone référencée, la fonction renvoie 0.

Paramètres

point Requis. Spécifie le point à tester.

puppetPalette()

Utilisation

```
-- Syntaxe Lingo
_movie.puppetPalette(palette {, vitesse} {, images})

// Syntaxe JavaScript
_movie.puppetPalette(palette {, vitesse} {, images});
```

Description

Méthode d'animation ; asservit la piste des palettes et permet au script d'avoir priorité sur les paramètres de la piste des palettes du scénario, ainsi que d'affecter des palettes à l'animation.

La méthode `puppetPalette()` définit comme palette courante l'acteur palette spécifié par *palette*. Si *palette* est une chaîne, celle-ci spécifie le nom d'acteur de la palette. Si *palette* est un nombre entier, celui-ci spécifie le numéro d'acteur de la palette.

Pour optimiser les résultats, utilisez la méthode `puppetPalette()` avant de passer à l'image sur laquelle l'effet se produira, afin que Director puisse établir une correspondance avec la palette voulue avant de dessiner l'image suivante.

Vous pouvez faire apparaître progressivement la palette en remplaçant *vitesse* par un nombre entier compris entre 1 (la plus lente) et 60 (la plus rapide). Vous pouvez aussi faire apparaître progressivement la palette sur plusieurs images en remplaçant *images* par un nombre entier correspondant au nombre d'images.

Une palette asservie reste active jusqu'à ce que vous la désactiviez avec la syntaxe `_movie.puppetPalette(0)`. Aucune autre modification de palette ne sera apportée dans le scénario lorsque la palette asservie est active.

Remarque : Le navigateur web contrôle la palette pour toute la page web. Ainsi, Shockwave Player utilise toujours la palette du navigateur.

Paramètres

palette Requis. Chaîne ou numéro qui spécifie le nom ou le numéro de la nouvelle palette.

vitesse Facultatif. Nombre entier qui spécifie la vitesse d'un fondu. Les valeurs correctes sont comprises entre 1 et 60.

images Facultatif. Nombre entier qui spécifie le nombre d'images correspondant au déroulement du fondu.

Exemple

L'instruction suivante fait d'Arc-en-ciel la palette de l'animation :

```
-- Syntaxe Lingo
_movie.puppetPalette("Arc-en-ciel")

// Syntaxe JavaScript
_movie.puppetPalette("Arc-en-ciel");
```

L'instruction suivante fait de Niveaux de gris la palette de l'animation. La transition à la palette Niveaux de gris se fait pendant un laps de temps de 15, et sur 20 images.

```
-- Syntaxe Lingo
_movie.puppetPalette("Arc-en-ciel", 15, 20)

// Syntaxe JavaScript
_movie.puppetPalette("Arc-en-ciel", 15, 20);
```

Voir aussi

Animation

puppetSprite()

Utilisation

```
-- Syntaxe Lingo
_movie.puppetSprite(entNumDimageObjet, bool)

// Syntaxe JavaScript
_movie.puppetSprite(entNumDimageObjet, bool);
```

Description

Méthode d'animation ; détermine si une piste d'image-objet est un esclave contrôlé par script (TRUE) ou si, n'étant pas un esclave, elle est sous le contrôle du scénario (FALSE).

Tant que la tête de lecture reste dans la même image-objet, le fait de désactiver l'asservissement de la piste d'image-objet au moyen de la syntaxe puppetSprite(*entNumDimageObjet*, FALSE) rend à l'image-objet les propriétés définies dans le scénario.

Les propriétés initiales de la piste d'image-objet sont celles de la piste au moment de l'exécution de la méthode puppetSprite(). Vous pouvez utiliser des scripts pour modifier les propriétés d'image-objet comme suit :

- Si une piste d'image-objet est asservie, toute modification apportée par un script aux propriétés d'image-objet de la piste reste en vigueur une fois la tête de lecture sortie de l'image-objet.
- Si une piste d'image-objet n'est pas asservie, toute modification apportée par un script à l'image-objet ne dure que jusqu'à la fin de l'image-objet courante.

La piste doit contenir une image-objet lorsque vous utilisez la méthode `puppetSprite()`.

Le fait d'asservir la piste d'image-objet vous permet de contrôler par des scripts de nombreuses propriétés d'image-objet, telles que `member`, `locH` et `width`, une fois la tête de lecture sortie de l'image-objet.

Utilisez la syntaxe `puppetSprite(entNumDimageObjet, FALSE)` pour rendre le contrôle au scénario lorsque vous avez fini de contrôler une piste d'image-objet avec des scripts et pour éviter les résultats imprévisibles qui peuvent se produire lorsque la tête de lecture se trouve dans des images non destinées à être asservies.

Remarque : La version 6 de Director a introduit l'asservissement automatique qui, dans la plupart des cas, évite d'avoir à asservir explicitement une image-objet. Le contrôle explicite est toujours utile pour conserver le contrôle complet du contenu d'une piste, même après la fin de la lecture d'une plage d'images-objets.

Paramètres

entNumDimageObjet Requis. Nombre entier qui spécifie la piste d'image-objet à tester.

bool Requis. Valeur booléenne qui spécifie si une piste d'image-objet contrôlée par des scripts (`TRUE`) ou par le scénario (`FALSE`).

Exemple

L'instruction suivante asservit l'image-objet de la piste 15 :

```
-- Syntaxe Lingo
_movie.puppetSprite(15, TRUE)

// Syntaxe JavaScript
_movie.puppetSprite(15, true);
```

L'instruction suivante libère de son état d'asservissement l'image-objet de la piste numéro i + 1 :

```
-- Syntaxe Lingo
_movie.puppetSprite(i + 1, FALSE)

// Syntaxe JavaScript
_movie.puppetSprite(i + 1, false);
```

Voir aussi

`makeScriptedSprite()`, `Animation`, `Piste d'image-objet`

puppetTempo()

Utilisation

```
-- Syntaxe Lingo
_movie.puppetTempo(entCadence)

// Syntaxe JavaScript
_movie.puppetTempo(entCadence);
```

Description

Méthode d'animation ; asservit la piste des cadences et règle la cadence sur le nombre d'images spécifié.

Lorsque la piste des cadences est asservie, les scripts peuvent être prioritaires sur le paramètre de cadence du scénario et modifier la cadence de l'animation.

Il n'est pas nécessaire de désactiver la cadence esclave pour que les modifications ultérieurement apportées à la cadence dans le scénario prennent effet.

Remarque : Bien que la limite théorique des cadences d'image soit de 30 000 ips (images par seconde) avec la méthode `puppetTempo()`, une telle cadence ne serait possible qu'avec très peu d'animation et une machine très puissante.

Paramètres

`entCadence` Requis. Nombre entier qui spécifie la cadence.

Exemple

L'instruction suivante fixe la cadence de l'animation à 30 images par seconde :

```
-- Syntaxe Lingo
_movie.puppetTempo(30)

// Syntaxe JavaScript
_movie.puppetTempo(30);
```

L'instruction suivante augmente l'ancienne cadence de l'animation de 10 images par seconde :

```
-- Syntaxe Lingo
_movie.puppetTempo(ancienneCadence + 10)

// Syntaxe JavaScript
_movie.puppetTempo(ancienneCadence + 10);
```

Voir aussi

Animation

puppetTransition()

Utilisation

```
-- Syntaxe Lingo
_movie.puppetTransition(réfObjActeur)
_movie.puppetTransition(ent {, durée} {, taille} {, zone})

// Syntaxe JavaScript
_movie.puppetTransition(réfObjActeur);
_movie.puppetTransition(ent {, durée} {, taille} {, zone});
```

Description

Méthode d'animation ; exécute la transition spécifiée entre l'image courante et la suivante.

Pour utiliser un acteur Xtra de transition, utilisez la syntaxe `puppetTransition(réfObjActeur)`.

Pour utiliser une transition Director intégrée, remplacez *ent* par l'une des valeurs du tableau suivant. Remplacez *durée* par le nombre de quarts de secondes utilisés pour effectuer la transition. La valeur minimum est 0 ; la valeur maximum est 120 (30 secondes). Remplacez *taille* par le nombre de pixels dans chaque bloc de la transition. La valeur minimum est 1 ; la valeur maximum est 128. Plus les blocs sont petits, plus la transition se fait en douceur, mais plus elle est lente.

Code	Transition	Code	Transition
01	Balayage vers la droite	27	Rangées aléatoires
02	Balayage vers la gauche	28	Colonnes aléatoires
03	Balayage vers le bas	29	Recouvrir vers le bas
04	Balayage vers le haut	30	Recouvrir vers le bas à gauche
05	Centre vers les bords, horizontal	31	Recouvrir vers le bas à droite
06	Bords vers centre, horizontal	32	Recouvrir vers la gauche
07	Centre vers les bords, vertical	33	Recouvrir vers la droite
08	Bords vers centre, vertical	34	Recouvrir vers le haut
09	Centre vers les bords, carré	35	Recouvrir vers le haut à gauche
10	Bords vers centre, carré	36	Recouvrir vers le haut à droite
11	Pousser vers la gauche	37	Stores vénitiens
12	Pousser vers la droite	38	Damier
13	Pousser vers le bas	39	Bandes vers la gauche en bas
14	Pousser vers le haut	40	Bandes vers la droite en bas
15	Révéler vers le haut	41	Bandes vers le bas à gauche
16	Révéler vers le haut à droite	42	Bandes vers le haut à gauche
17	Révéler vers la droite	43	Bandes vers le bas à droite
18	Révéler vers le bas à droite	44	Bandes vers le haut à droite
19	Révéler vers le bas	45	Bandes vers la gauche sur le dessus
20	Révéler vers le bas à gauche	46	Bandes vers la droite sur le dessus
21	Révéler vers la gauche	47	Zoom ouvert
22	Révéler vers le haut à gauche	48	Zoom fermé
23	Fondu, pixels rapides*	49	Stores verticaux
24	Fondu, rectangles carrés	50	Fondu, bits rapides*

Code	Transition	Code	Transition
25	Fondu, carrés d'encadrement	51	Fondu, pixels*
26	Fondu, motifs	52	Fondu, bits*

Les transitions identifiées par un astérisque (*) ne fonctionnent pas sur les moniteurs 32 bits.

Il n'y a pas de lien direct entre une faible valeur de durée et une transition rapide. La vitesse réelle de la transition dépend de la relation entre *taille* et *durée*. Par exemple, si *taille* a une valeur de 1 pixel, la transition prend plus de temps, quelle que soit la valeur de durée, car cette opération représente un travail intense de l'ordinateur. Pour accélérer les transitions, augmentez la taille des blocs au lieu de réduire la durée.

Remplacez *zone* par une valeur qui détermine si la transition se produit uniquement dans la zone modifiée (TRUE) ou sur toute la scène (FALSE, par défaut). La variable *zone* est une zone dans laquelle les images-objets ont changé.

Paramètres

réfObjActeur Requis si vous utilisez un acteur Xtra de transition. Référence à l'acteur Xtra à utiliser en tant que transition.

ent Requis si vous utilisez une transition intégrée à Director. Nombre entier qui spécifie le numéro de la transition à utiliser.

durée Facultatif. Nombre entier qui spécifie le nombre de quarts de secondes utilisés pour effectuer la transition. Les valeurs correctes sont comprises entre 0 et 120.

taille Facultatif. Nombre entier qui spécifie le nombre de pixels dans chaque bloc de la transition. Les valeurs correctes sont comprises entre 1 et 128.

zone Facultatif. Valeur booléenne qui détermine si la transition se produit uniquement dans la zone modifiée (TRUE) ou sur toute la scène (FALSE).

Exemple

L'instruction suivante effectue une transition de type balayage vers la droite. Comme aucune valeur n'est spécifiée pour *zone*, la transition se produit sur toute la scène, qui correspond au choix par défaut.

```
-- Syntaxe Lingo
_movie.puppetTransition(1)

// Syntaxe JavaScript
_movie.puppetTransition(1);
```

L'instruction suivante effectue une transition sur toute la scène, de type balayage vers la gauche, d'une durée d'une seconde, la taille de blocs étant de 20 :

```
-- Syntaxe Lingo
_movie.puppetTransition(2, 4, 20, FALSE)

// Syntaxe JavaScript
_movie.puppetTransition(2, 4, 20, false);
```

Voir aussi

Animation

put()

Utilisation

```
-- Syntaxe Lingo
put(valeur)

// Syntaxe JavaScript
put(valeur);
```

Description

Fonction du niveau supérieur ; évalue une expression et affiche le résultat dans la fenêtre Messages.

La fonctionnalité de cette méthode est identique à celle de la méthode `trace()` du niveau supérieur, disponible à la fois pour Lingo et la syntaxe JavaScript.

Cette méthode peut servir d'outil de débogage pour suivre la valeur des variables au cours de la lecture d'une animation.

Paramètres

`valeur` Requis. Expression à évaluer.

Exemple

L'instruction suivante affiche l'heure dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(_system.time())

// Syntaxe JavaScript
put(_system.time());
```

L'instruction suivante affiche la valeur affectée à la variable `Devis` dans la fenêtre Messages :

```
-- Syntaxe Lingo
bid = "Dubois"
put(bid) -- "Dubois"

// Syntaxe JavaScript
var bid = "Dubois";
put(bid); // Dubois
```

Voir aussi

`trace()`

qtRegisterAccessKey()

Utilisation

```
-- Syntaaxe Lingo
qtRegisterAccessKey(chaîneDeCatégorie, chaîneClé)

// Syntaxe JavaScript
qtRegisterAccessKey(chaîneDeCatégorie, chaîneClé);
```

Description

Commande ; permet l'enregistrement d'une clé pour un média QuickTime chiffré.

La clé agit au niveau applicatif et non au niveau système. Une fois que l'application annule l'enregistrement de la clé ou se ferme, le média n'est plus accessible.

Remarque : Pour raisons de sécurité, il est impossible d'afficher la liste de toutes les clés enregistrées.

Voir aussi

`qtUnRegisterAccessKey()`

qtUnRegisterAccessKey()

Utilisation

```
-- Syntaxe Lingo
qtUnRegisterAccessKey(chaîneDeCatégorie, chaîneClé)

// Syntaxe JavaScript
qtUnRegisterAccessKey(chaîneDeCatégorie, chaîneClé);
```

Description

Commande ; permet d'annuler l'enregistrement d'une clé pour un média QuickTime chiffré.

La clé agit au niveau applicatif et non au niveau système. Une fois l'enregistrement de la clé annulé par l'application, seules les animations chiffrées avec cette clé peuvent toujours être lues. Les autres médias ne sont plus accessibles.

Voir aussi

`qtRegisterAccessKey()`

queue()

Utilisation

```
-- Syntaxe Lingo
réfObjPisteAudio.queue(réfObjActeur)
réfObjPisteAudio.queue(listeDesPropriétés)

// Syntaxe JavaScript
réfObjPisteAudio.queue(réfObjActeur);
réfObjPisteAudio.queue(listeDesPropriétés);
```

Description

Méthode de piste audio ; ajoute un acteur son à la file d'attente d'une piste audio.

Dès qu'un son est placé en file d'attente, il peut être lu immédiatement à l'aide de la méthode `play()`. Ceci s'explique par le fait que Director précharge une certaine partie de chaque son placé en file d'attente, pour éviter les délais d'attente entre la méthode `play()` et le début de la lecture. Cette proportion du son préchargée s'élève par défaut à 1 500 millisecondes. Ce paramètre peut être modifié en passant une liste des propriétés contenant un ou plusieurs paramètres, par l'intermédiaire de la méthode `queue()`. Ces paramètres peuvent également être passés avec la méthode `setPlayList()`.

Pour un exemple d'utilisation de `queue()` dans une animation, reportez-vous à l'animation Sound Control du dossier Learning/Lingo, lui-même inclus dans le dossier de Director.

Paramètres

`réfObjActeur` Requis si vous spécifiez un acteur son. Référence à l'acteur son à placer en file d'attente.

listeDesPropriétés Requis si vous passez une liste de propriétés comme paramètres. Liste de propriétés qui s'applique à l'acteur son à placer en file d'attente. Ces propriétés sont :

Propriété	Description
#member	L'acteur à placer en file d'attente. Cette propriété doit être spécifiée. Toutes les autres sont facultatives.
#startTime	Position temporelle de départ de la lecture du son, en millisecondes. La valeur par défaut est le début du son. Pour plus d'informations, consultez `startTime`.
#endTime	Position temporelle de fin de la lecture du son, en millisecondes. La valeur par défaut est la fin du son. Pour plus d'informations, consultez `endTime`.
#loopCount	Nombre de répétitions d'une boucle défini avec `#loopStartTime` et `#loopEndTime`. La valeur par défaut est 1. Voir `loopCount`.
#loopStartTime	Position temporelle de départ de la boucle, en millisecondes. Pour plus d'informations, consultez `loopStartTime`.
#loopEndTime	Position temporelle de fin de la boucle, en millisecondes. Pour plus d'informations, consultez `loopEndTime`.
#preloadTime	Quantité de son à placer en mémoire tampon avant la lecture, en millisecondes. Pour plus d'informations, consultez `preloadTime`.

Exemple

Le gestionnaire suivant place en file d'attente et lit deux sons. Le premier son, l'acteur Carillons, est lu dans son intégralité. Le second son, l'acteur Intro, est lu à partir du point 3 secondes, exécute cinq lectures en boucle successives, du point 8 secondes au point 8,9 secondes, et s'arrête au point 10 secondes.

```
-- Syntaxe Lingo
on playMusic
  sound(2).queue(member("Carillon"))
  sound(2).queue([#member:member("intro"), #startTime:3000, \
  #endTime:10000, #loopCount:5, #loopStartTime:8000, #loopEndTime:8900])
  sound(2).play()
end playMusic

// Syntaxe JavaScript
function playMusic() {
  sound(2).queue(member("Carillon"))
  sound(2).queue(propList("member",member("intro"), "startTime",3000,
  "endTime",10000, "loopCount",5, "loopStartTime",8000, "loopEndTime",8900));
  sound(2).play();
}
```

Voir aussi

endTime, loopCount, loopEndTime, loopStartTime, pause() (piste audio), play() (piste audio), preLoadTime, setPlayList(), Piste audio, startTime, stop() (piste audio)

queue() (3D)

Utilisation

```
member(quelActeur).model(quelModèle).bonesPlayer.queue\
  (nomDeMouvement {, enBoucle, positionInitiale, positionFinale, échelle,
  décalage})
member(quelActeur).model(quelModèle).keyframePlayer.\
  queue(nomDeMouvement {, enBoucle, positionInitiale, positionFinale, échelle,
  décalage})
```

Description

Commande 3D de modificateur `keyframePlayer` et `bonesPlayer` ; ajoute le mouvement spécifié à la fin de la propriété `playList` du modificateur. Le mouvement est exécuté par le modèle lorsque tous les mouvements qui le précèdent dans la liste de lecture ont été lus.

Paramètres

`nomDeMouvement` Requis. Spécifie le nom du mouvement à ajouter.

`enBoucle` Facultatif. Spécifie si le mouvement est lu une seule fois (`FALSE`) ou continuellement (`TRUE`).

`positionInitiale` Facultatif. Ce paramètre est mesuré en millisecondes, à partir du début du mouvement. Lorsque *`enBoucle`* a pour valeur `FALSE`, le mouvement démarre à la position *`décalage`* et se termine à la *`positionFinale`*. Lorsque *`enBoucle`* a pour valeur `TRUE`, la première itération de la boucle démarre à *`décalage`* et se termine à *`positionFinale`*. Toutes les répétitions suivantes démarrent à *`positionInitiale`* et se terminent à *`positionFinale`*.

`positionFinale` Facultatif. Ce paramètre est mesuré en millisecondes, à partir du début du mouvement. Lorsque *`enBoucle`* a pour valeur `FALSE`, le mouvement démarre à la position *`décalage`* et se termine à la *`positionFinale`*. Lorsque *`enBoucle`* a pour valeur `TRUE`, la première itération de la boucle démarre à *`décalage`* et se termine à *`positionFinale`*. Toutes les répétitions suivantes démarrent à *`positionInitiale`* et se terminent à *`positionFinale`*. Donnez à *`positionFinale`* la valeur -1 si le mouvement doit être lu jusqu'à la fin.

`échelle` Facultatif. Spécifie la cadence réelle de la lecture du mouvement. La valeur du paramètre *`échelle`* est multipliée par la propriété `playRate` du modificateur `#keyframePlayer` ou `#bonesPlayer` du modèle pour déterminer la cadence réelle de la lecture du mouvement.

`décalage` Facultatif. Ce paramètre est mesuré en millisecondes, à partir du début du mouvement. Lorsque *`enBoucle`* a pour valeur `FALSE`, le mouvement démarre à la position *`décalage`* et se termine à la *`positionFinale`*. Lorsque *`enBoucle`* a pour valeur `TRUE`, la première itération de la boucle démarre à *`décalage`* et se termine à *`positionFinale`*. Toutes les répétitions suivantes démarrent à *`positionInitiale`* et se terminent à *`positionFinale`*.

Exemple

La commande suivante ajoute le mouvement Chute à la fin de la liste de lecture `bonesPlayer` du modèle Marcheur. Lorsque tous les mouvements précédant Chute dans la liste de lecture ont été exécutés, Chute est lu une fois du début à la fin.

```
sprite(1).member.model("Marcheur").bonesPlayer.queue\
  ("Chute", 0, 0, -1, 1, 0)
```

La commande suivante ajoute le mouvement coupDenvoi à la fin de la liste de lecture `bonesPlayer` du modèle Marcheur. Lorsque tous les mouvements précédant coupDenvoi dans la liste de lecture ont été exécutés, une section de coupDenvoi est lue en boucle. La première itération de la boucle démarrera 2000 millisecondes à compter du début du mouvement. Toutes les itérations suivantes de la boucle démarreront à 1000 millisecondes du début de coupDenvoi et se termineront à 5000 millisecondes du début de coupDenvoi. La cadence de lecture sera égale à trois fois la propriété `playRate` du modificateur `bonesPlayer` du modèle.

```
sprite(1).member.model("Marcheur").bonesPlayer.queue("coupDenvoi", 1, \
  1000, 5000, 3, 2000)
```

Voir aussi

play() (3D), playNext() (3D), playRate (3D)

quickTimeVersion()

Utilisation

```
-- Syntaxe Lingo
quickTimeVersion()

// Syntaxe JavaScript
quickTimeVersion();
```

Description

Fonction ; renvoie une valeur à virgule flottante identifiant la version de QuickTime installée et remplace la fonction `QuickTimePresent` précédente.

Si plusieurs versions de QuickTime 3.0 (ou plus récent) sont installées sous Windows, `quickTimeVersion()` renvoie le numéro de version le plus récent. Si une version antérieure à QuickTime 3.0 est installée, `quickTimeVersion()` renvoie le numéro de version 2.1.2, quelle que soit la version installée.

Paramètres

Aucun.

Exemple

L'instruction suivante utilise `quickTimeVersion()` pour afficher la version de QuickTime courante dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(quickTimeVersion())

// Syntaxe JavaScript
put(quickTimeVersion());
```

quit()

Utilisation

```
-- Syntaxe Lingo
_player.quit()

// Syntaxe JavaScript
_player.quit();
```

Description

Méthode de lecteur ; permet de quitter Director ou une projection et de passer au bureau de Windows ou au Finder du Macintosh.

Paramètres

Aucun.

Exemple

L'instruction suivante ordonne à l'ordinateur de quitter Director ou la projection et de passer au bureau de Windows ou au Finder du Macintosh lorsque l'utilisateur appuie sur Ctrl+Q (Windows) ou sur Cmd+Q (Macintosh) :

```
-- Syntaxe Lingo
if (_key.key = "q" and _key.commandDown) then
  _player.quit()
end if

// Syntaxe JavaScript
if (_key.key == "q" && _key.commandDown) {
  _player.quit();
}
```

Voir aussi

Lecteur

ramNeeded()

Utilisation

```
-- Syntaxe Lingo
_movie.ramNeeded(entImageInitiale, entImageFinale)

// Syntaxe JavaScript
_movie.ramNeeded(entImageInitiale, entImageFinale);
```

Description

Méthode d'animation ; détermine la mémoire requise, en octets, pour afficher une plage d'images. Par exemple, vous pouvez tester la taille d'images contenant des illustrations en mode 32 bits : si la valeur de `ramNeeded()` est supérieure à celle de `freeBytes()`, utilisez des images contenant des illustrations en mode 8 bits et divisez par 1 024 pour convertir les octets en kilo-octets.

Paramètres

entImageInitiale Requis. Nombre entier qui spécifie le numéro de la première image de la plage.

entImageFinale Requis. Nombre entier qui spécifie le numéro de la dernière image de la plage.

Exemple

L'instruction suivante affecte à la variable `tailleDimage` le nombre d'octets requis pour afficher les images 100 à 125 de l'animation :

```
-- Syntaxe Lingo
tailleDimage = _movie.ramNeeded(100, 125)

// Syntaxe JavaScript
var tailleDimage = _movie.ramNeeded(100, 125);
```

L'instruction suivante détermine si la mémoire requise pour afficher les images 100 à 125 excède la mémoire disponible et, le cas échéant, passe à la section utilisant des acteurs qui possèdent un nombre de couleurs inférieur :

```
-- Syntaxe Lingo
if (_movie.ramNeeded(100, 125) > _system.freeBytes) then
  _movie.go("8 bits")
end if

// Syntaxe JavaScript
if (_movie.ramNeeded(100, 125) > _system.freeBytes) {
  _movie.go("8 bits");
}
```

Voir aussi

freeBytes(), Animation

random()

Utilisation

```
-- Syntaxe Lingo
random(expressionEntière)

// Syntaxe JavaScript
random(expressionEntière);
```

Description

Fonction du niveau supérieur ; renvoie un nombre entier aléatoire dans la plage comprise entre 1 et une valeur spécifiée. Cette fonction peut s'utiliser pour modifier les valeurs d'une animation et peut servir notamment à faire varier les trajectoires dans les jeux, à affecter des nombres aléatoires ou à changer la couleur ou la position des images-objets.

Pour définir un ensemble de nombres aléatoires de façon qu'ils commencent par un nombre autre que 1, soustrayez la quantité appropriée de la fonction `random()`. Par exemple, l'expression `random(n + 1) - 1` utilise une plage allant de 0 au nombre `n`.

Paramètres

expressionEntière Requis. Spécifie la valeur maximum du nombre aléatoire.

Exemple

L'instruction suivante affecte des valeurs aléatoires à la variable `dés` :

```
-- Syntaxe Lingo
dés = (random(6) + random(6))

// Syntaxe JavaScript
var dés = (random(6) + random(6));
```

L'instruction suivante modifie de façon aléatoire la couleur de premier plan de l'image-objet 10 :

```
-- Syntaxe Lingo
sprite(10).foreColor = (random(256) - 1)

// Syntaxe JavaScript
sprite(10).foreColor = (random(256) - 1);
```

Le gestionnaire suivant choisit de manière aléatoire lequel des deux segments de l'animation va être lu :

```
-- Syntaxe Lingo
on sélectionDeScène
  if (random(2) = 2) then
    _movie.go("11a")
  else
    _movie.go("11b")
  end if
end

// Syntaxe JavaScript
function sélectionDeScène() {
  if (random(2) == 1) {
    _movie.go("11a");
  } else {
    _movie.go("11b");
  }
}
```

L'instruction suivante produit un multiple aléatoire de cinq compris entre 5 et 100 :

```
-- Syntaxe Lingo
score = (5 * random(20))

// Syntaxe JavaScript
var score = (5 * random(20));
```

randomVector()

Utilisation

```
-- Syntaxe Lingo
randomVector()

// Syntaxe JavaScript
randomVector();
```

Description

Fonction du niveau supérieur ; renvoie un vecteur unitaire qui décrit un point choisi de manière aléatoire à la surface d'une sphère unitaire.

Cette fonction diffère de `vector(random(10)/10.0, random(10)/10.0, random(10)/10.0,)` dans la mesure où vous êtes assuré que le vecteur résultant de l'utilisation de `randomVector()` sera un vecteur unitaire.

Un vecteur unitaire a toujours une longueur égale à un.

Paramètres

Aucun.

Exemple

Les instructions suivantes créent et affichent deux vecteurs unitaires définis de manière aléatoire dans la fenêtre Messages :

```
-- Syntaxe Lingo
vec1 = randomVector()
vec2 = randomVector()
put(vec1 & RETURN & vec2)

// Syntaxe JavaScript
var vec1 = randomVector();
var vec2 = randomVector();
put(vec1 + "\n" + vec2);
```

Voir aussi

`vector()`

randomVector

Utilisation

`randomVector()`

Description

Commande 3D ; renvoie un vecteur unitaire qui décrit un point choisi de manière aléatoire à la surface d'une sphère unitaire. Cette méthode diffère de `vector( random(10)/10.0, random(10)/10.0, random(10)/10.0)` dans la mesure où il est sûr que le vecteur résultant sera un vecteur unitaire.

Paramètres

Aucun.

Exemple

Les instructions suivantes créent et affichent deux vecteurs unitaires définis de manière aléatoire dans la fenêtre Messages :

```
vec = randomVector()
put vec
-- vector( -0.1155, 0.9833, -0.1408 )
vec2 = randomVector()
put vec2
-- vector( 0.0042, 0.8767, 0.4810 )
```

Voir aussi

`getNormalized`, `generateNormals()`, `normalize`

rawNew()

Utilisation

```
scriptParent.rawNew()
rawNew(scriptParent)
```

Description

Fonction ; crée un objet enfant à partir d'un script parent sans appeler son gestionnaire `on new`. Cela permet la création d'objets enfants sans initialiser leurs propriétés. Cette fonction s'avère particulièrement pratique lors de la création d'un grand nombre d'objets enfants pour une utilisation ultérieure. Pour initialiser les propriétés de l'un de ces objets enfants bruts, appelez son gestionnaire `on new`.

Paramètres

Aucun.

Exemple

L'instruction suivante crée un objet enfant appelé voitureRouge à partir du script parent ScriptParentDeVoiture, sans initialiser ses propriétés :

```
voitureRouge = script("ScriptParentDeVoiture").rawNew()
```

L'instruction suivante initialise les propriétés de l'objet enfant voitureRouge :

```
voitureRouge.new()
```

Voir aussi

`new()`, `script()`

readChar()

Utilisation

```
-- Syntaxe Lingo
réfObjFileio.readChar()

// Syntaxe JavaScript
réfObjFileio.readChar();
```

Description

Méthode de Fileio ; lit le prochain caractère d'un fichier et renvoie son code ASCII.

Vous devez d'abord ouvrir un fichier en appelant `openFile()` avant d'utiliser `readChar()` pour lire un caractère.

Paramètres

Aucun.

Voir aussi

`Fileio`, `openFile()`

readFile()

Utilisation

```
-- Syntaxe Lingo
réfObjFileio.readFile()

// Syntaxe JavaScript
réfObjFileio.readFile();
```

Description

Méthode de Fileio ; lit la position courante à la fin d'un fichier spécifié et renvoie le résultat sous forme d'une chaîne.

Vous devez d'abord ouvrir un fichier en appelant `openFile()` avant d'utiliser `readFile()` pour lire un fichier.

Paramètres

Aucun.

Voir aussi

`Fileio`, `openFile()`

readLine()

Utilisation

```
-- Syntaxe Lingo
réfObjFileio.readLine()

// Syntaxe JavaScript
réfObjFileio.readLine();
```

Description

Méthode de Fileio ; lit la ligne suivante d'un fichier, y compris le RETURN, et la renvoie sous forme d'une chaîne.

Vous devez d'abord ouvrir un fichier en appelant `openFile()` avant d'utiliser `readLine()` pour lire une ligne.

Paramètres

Aucun.

Voir aussi

`Fileio`, `openFile()`

readToken()

Utilisation

```
-- Syntaxe Lingo
réfObjFileio.readToken(chaîneSauter, chaîneCouper)

// Syntaxe JavaScript
réfObjFileio.readToken(chaîneSauter, chaîneCouper);
```

Description

Méthode de Fileio ; lit le jeton suivant et le renvoie sous forme d'une chaîne.

Vous devez d'abord ouvrir un fichier en appelant `openFile()` avant d'utiliser `readToken()` pour lire un jeton.

Paramètres

`chaîneSauter` Requis. Chaîne qui spécifie l'ensemble de caractères après lequel le jeton commence. La chaîne *`chaîneSauter`* n'est pas incluse dans la chaîne renvoyée.

`chaîneCouper` Requis. Chaîne qui spécifie l'ensemble de caractères avant lequel le jeton se termine. La chaîne *`chaîneCouper`* n'est pas incluse dans la chaîne renvoyée.

Voir aussi

`Fileio`, `openFile()`

readWord()

Utilisation

```
-- Syntaxe Lingo
réfObjFileio.readWord()

// Syntaxe JavaScript
réfObjFileio.readWord();
```

Description

Méthode de Fileio ; lit le mot suivant d'un fichier et le renvoie sous forme d'une chaîne.

Vous devez d'abord ouvrir un fichier en appelant `openFile()` avant d'utiliser `readWord()` pour lire un mot.

Paramètres

Aucun.

Voir aussi

`Fileio`, `openFile()`

realPlayerNativeAudio()

Utilisation

```
-- Syntaxe Lingo
realPlayerNativeAudio()

// Syntaxe JavaScript
realPlayerNativeAudio();
```

Description

Fonction RealMedia ; permet d'obtenir ou de définir un indicateur global déterminant si la portion audio de l'acteur RealMedia est traitée par RealPlayer (`TRUE`) ou par Director (`FALSE`). Cette fonction renvoie la valeur précédente de l'indicateur.

Pour être efficace, cet indicateur doit être défini avant le premier chargement de RealPlayer (lors de la rencontre du premier acteur RealMedia dans le scénario ou de la première référence Lingo à un acteur RealMedia). Toute modification apportée à cet indicateur après le chargement de RealPlayer sera ignorée. Cet indicateur devrait être exécuté dans un gestionnaire d'événement `prepareMovie` d'un script d'animation. Cet indicateur est défini pour la session entière (à partir du moment ou le Shockwave Player est chargé jusqu'à sa fermeture et son redémarrage), et non uniquement pour la durée de l'animation courante.

Par défaut, cet indicateur est défini sur `FALSE` et le traitement audio est lancé par Director, ce qui permet de définir la propriété `soundChannel` et d'utiliser les méthodes et propriétés audio standard de Lingo pour le traitement du flux audio d'une image-objet RealMedia, par exemple le mixage d'un élément RealAudio avec d'autres composants audio de Director. Si cet indicateur est défini sur `TRUE`, le contrôle Lingo de la piste audio n'est pas réalisé et le son est traité par RealPlayer.

Paramètres

Aucun.

Exemple

Le code suivant indique que la fonction `realPlayerNativeAudio()` a pour valeur `FALSE`, ce qui signifie que l'audio de l'acteur RealMedia sera traitée par Director.

```
-- Syntaxe Lingo
put(realPlayerNativeAudio())
-- 0

// Syntaxe JavaScript
trace(realPlayerNativeAudio());
// 0
```

Le code suivant donne à la fonction `realPlayerNativeAudio()` la valeur `TRUE`, ce qui signifie que l'audio du flux RealMedia sera traité par RealPlayer et que le contrôle du code sur la piste audio ne sera pas pris en compte.

```
-- Syntaxe Lingo
realPlayerNativeAudio(TRUE)

// Syntaxe JavaScript
realPlayerNativeAudio(1);
```

Voir aussi

`soundChannel (RealMedia)`

realPlayerPromptToInstall()

Utilisation

```
-- Syntaxe Lingo
realPlayerPromptToInstall()

// Syntaxe JavaScript
realPlayerPromptToInstall();
```

Description

Fonction RealMedia ; permet d'obtenir ou de définir un indicateur global déterminant si la détection automatique et les alertes de RealPlayer 8 sont activées (`TRUE`) ou non (`FALSE`).

Par défaut, cette fonction est définie sur `TRUE`, ce qui signifie que si les utilisateurs ne disposent pas de la version 8 de RealPlayer et tentent de charger une animation contenant RealMedia, un message s'affiche automatiquement pour leur demander s'ils veulent accéder au site web de RealNetworks et installer RealPlayer. Vous pouvez définir cet indicateur sur `FALSE` si vous souhaitez créer votre propre système de détection et d'alerte à l'aide de la fonction `realPlayerVersion()`, page 529, et de code personnalisé. Si cet indicateur est défini sur `FALSE` et qu'un autre système de détection et d'alerte n'est pas mis en place, les utilisateurs non équipés de RealPlayer pourront charger des animations contenant des acteurs RealMedia, mais les images-objets RealMedia n'apparaîtront pas.

Cette fonction détecte la version de RealPlayer installée sur le système de l'utilisateur pour déterminer si RealPlayer 8 est installé. Sous Windows, les numéros de version 6.0.8.132 ou supérieure indiquent que RealPlayer 8 est installé. Sur Macintosh, les composants de base de RealPlayer de version 6.0.7.1001 ou supérieure indiquent que RealPlayer 8 est installé.

Cet indicateur devrait être exécuté dans un gestionnaire d'événement `prepareMovie` d'un script d'animation.

Cette fonction renvoie la valeur précédente de l'indicateur.

Paramètres

Aucun.

Exemple

Le code suivant indique que la fonction `realPlayerPromptToInstall()` a pour valeur `TRUE`, ce qui signifie que les utilisateurs qui ne possèdent pas RealPlayer seront invités à l'installer.

```
-- Syntaxe Lingo
put(realPlayerPromptToInstall()) -- 1

// Syntaxe JavaScript
-- Syntaxe Lingo
trace(realPlayerPromptToInstall()); // 1
```

Le code suivant donne à la fonction `realPlayerPromptToInstall()` la valeur `FALSE`, ce qui signifie que les utilisateurs ne seront pas invités à installer RealPlayer à moins que vous n'ayez créé un système de détection et d'alerte.

```
-- Syntaxe Lingo
realPlayerPromptToInstall(FALSE)

// Syntaxe JavaScript
realPlayerPromptToInstall(0);
```

realPlayerVersion()

Utilisation

```
-- Syntaxe Lingo
realPlayerVersion()

// Syntaxe JavaScript
realPlayerVersion();
```

Description

Fonction RealMedia ; renvoie une chaîne identifiant le numéro de version du logiciel RealPlayer installé sur le système de l'utilisateur, ou une chaîne vide si RealPlayer n'est pas installé. RealPlayer 8 ou une version ultérieure doit être installé sur votre ordinateur pour visualiser des animations Director intégrant du contenu RealMedia. Sous Windows, les numéros de version 6.0.8.132 ou supérieure indiquent que RealPlayer 8 est installé. Sur Macintosh, les composants de base de RealPlayer de version 6.0.7.1001 ou supérieure indiquent que RealPlayer 8 est installé.

L'objectif de cette fonction est de permettre la création de votre propre système de détection et d'alerte RealPlayer, si vous ne souhaitez pas utiliser celui qui est fourni par la fonction `realPlayerPromptToInstall()`, page 528.

Si vous choisissez de créer votre propre système de détection et d'alerte à l'aide de la fonction `realPlayerVersion()`, effectuez les opérations suivantes :

- Appelez `realPlayerPromptToInstall(FALSE)` (par défaut, cette fonction est définie sur `TRUE`) avant la référence des acteurs RealMedia dans Lingo ou leur apparition dans le scénario. Cette fonction devrait être définie dans un gestionnaire d'événements `prepareMovie` d'un script d'animation.
- Utilisez la propriété système `xtraList` pour vérifier si l'Xtra pour RealMedia (RealMedia Asset.x32) est répertorié dans la liste des Xtras de la boîte de dialogue Xtras de l'animation. La fonction `realPlayerVersion()` ne fonctionne pas si l'Xtra pour RealMedia est absent.

Le numéro de version renvoyé par cette fonction est identique au numéro de version que vous pouvez afficher dans RealPlayer.

Pour afficher le numéro de version de RealPlayer sous Windows :

1 Démarrez RealPlayer.

2 Choisissez A propos de RealPlayer dans le menu Aide.

Dans la fenêtre qui s'affiche, le numéro de version apparaît dans la partie supérieure de l'écran, au niveau de la seconde ligne.

Pour afficher le numéro de version de RealPlayer sur Macintosh :

1 Démarrez RealPlayer.

2 Choisissez A propos de RealPlayer dans le menu Pomme.

La boîte de dialogue A propos de RealPlayer apparaît. Ignorez le numéro de version indiqué dans la seconde ligne, dans la partie supérieure de l'écran ; il est incorrect.

3 Cliquez sur le bouton d'infos sur la version.

La boîte de dialogue d'informations de version de RealPlayer s'affiche.

4 Sélectionnez Composants de base de RealPlayer dans la liste des composants installés.

Le numéro de version affiché pour le composant de base de RealPlayer (par exemple, 6.0.8.1649) est identique à celui qui est renvoyé par `realPlayerVersion()`.

Paramètres

Aucun.

Exemple

Le code suivant indique que le numéro de version de RealPlayer installé sur le système est 6.0.9.357.

```
-- Syntaxe Lingo
put(realPlayerVersion())

// Syntaxe JavaScript
put(realPlayerVersion());
```

recordFont

Utilisation

```
recordFont(quelActeur, police {[,type]} {,[tailleDesBitmaps]}
  {,sousEnsembleDeCaractères} {, nouveauNom})
```

Description

Commande ; inclut une police TrueType ou Type 1 comme acteur. Une fois incluses, ces polices sont disponibles à l'auteur tout comme les autres polices installées sur le système.

Vous devez créer un acteur police vide à l'aide de la commande `new()` avant d'utiliser `recordFont`.

La commande crée une police shockée dans *`quelActeur`* en utilisant la police nommée dans le paramètre *`police`*. La valeur renvoyée par la commande indique si l'opération a réussi. La valeur zéro indique que l'opération a réussi.

Paramètres

police Requis. Spécifie le nom de la police d'origine à enregistrer.

type Facultatif. Spécifie une liste de symboles indiquant le type de la police d'origine. Les valeurs possibles sont #plain, #bold, #italic. Si vous ne définissez aucune valeur pour ce paramètre, c'est #plain qui sera utilisé par défaut.

tailleDesBitmaps Facultatif. Spécifie une liste d'entiers définissant les tailles pour lesquelles les bitmaps doivent être enregistrés. Ce paramètre peut être vide. Si vous omettez ce paramètre, aucun bitmap n'est généré. Ces bitmaps donnent généralement de meilleurs résultats pour les petites tailles (inférieures à 14 points), mais occupent plus de mémoire.

sousEnsembleDeCaractères Facultatif. Spécifie une chaîne de caractères à encoder. Seuls les caractères spécifiés seront disponibles dans la police. Si ce paramètre est omis, tous les caractères sont encodés. Si seuls certains caractères sont codés mais qu'un caractère non codé est utilisé, ce caractère apparaît comme une case vide.

nouveauNom Facultatif. Spécifie la chaîne utilisée comme nom de l'acteur police nouvellement enregistré.

Exemple

L'instruction suivante crée une police shockée simple n'utilisant que les deux arguments pour l'acteur et la police à enregistrer :

```
monNouvelActeurPolice = new(#font)
recordFont(monNouvelActeurPolice, "Module lunaire")
```

L'instruction suivante spécifie les tailles de bitmaps à générer et les caractères pour lesquels les données de police doivent être créées :

```
monNouvelActeurPolice = new(#font)
recordfont(monNouvelActeur,"Module lunaire", [], [14, 18, 45], "Nom du
  meilleur score du jeu Module lunaire")
```

Remarque : `recordFont` resynthétisant les données de la police au lieu de les utiliser directement, la distribution des polices shockées n'est soumise à aucune restriction légale.

Voir aussi

`newMember()`

rect()

Utilisation

```
-- Syntaxe Lingo
rect(entGauche, entHaut, entDroite, entBas)

// Syntaxe JavaScript
rect(entGauche, entHaut, entDroite, entBas);
```

Description

Fonction du niveau supérieur ; définit un rectangle.

Vous pouvez exécuter des opérations arithmétiques sur les rectangles en Lingo et avec la syntaxe JavaScript. Si vous ajoutez une seule valeur à un rectangle, Lingo et la syntaxe JavaScript l'ajoutent à chaque élément du rectangle.

Vous pouvez faire référence aux composants de rectangles avec les syntaxes de liste ou de propriétés. Par exemple, les affectations suivantes définissent `largeurMonRect1` et `largeurMonRect2` sur 50:

```
// Syntaxe JavaScript
var monRect = rect(40,30,90,70);
var largeurMonRect1 = monRect.right - monRect.left; // 50
var largeurMonRect2 = monRect[3] - monRect[1]; // 50
```

Pour un exemple d'utilisation de `rect()` dans une animation, reportez-vous à l'animation Imaging du dossier Learning/Lingo, lui-même inclus dans le dossier de Director.

Paramètres

`entGauche` Requis. Nombre entier qui spécifie le nombre de pixels entre le côté gauche du rectangle et le bord gauche de la scène.

`entHaut` Requis. Nombre entier qui spécifie le nombre de pixels entre le côté supérieur du rectangle et le bord supérieur de la scène.

`entDroite` Requis. Nombre entier qui spécifie le nombre de pixels entre le côté droit du rectangle et le bord droit de la scène.

`entBas` Requis. Nombre entier qui spécifie le nombre de pixels entre le côté inférieur du rectangle et le bord inférieur de la scène.

Exemple

L'instruction suivante définit la variable `nouvelleZone` comme un rectangle dont le côté gauche est placé à 100, le haut à 150, le côté droit à 300 et le bas à 400 pixels :

```
-- Syntaxe Lingo
nouvelleZone = rect(100, 150, 300, 400)

// Syntaxe JavaScript
var nouvelleZone = rect(100, 150, 300, 400);
```

En Lingo uniquement, l'instruction suivante définit la variable `nouvelleZone` comme un rectangle défini par les points `premierPoint` et `deuxièmePoint`.

```
-- Syntaxe Lingo
premierPoint = point(100, 150)
deuxièmePoint = point(300, 400)
nouvelleZone = rect(premierPoint, deuxièmePoint)
```

En Lingo uniquement, les instructions suivantes ajoutent et soustraient des valeurs aux rectangles :

```
-- Syntaxe Lingo
put(rect(0, 0, 100, 100) + rect(30, 55, 120, 95)) -- rect(30, 55, 220, 195)
put(rect(0, 0, 100, 100) -rect(30, 55, 120, 95)) -- rect(-30, -55, -20, 5)
```

En Lingo uniquement, l'instruction suivante ajoute 80 à chaque coordonnée d'un rectangle :

```
-- Syntaxe Lingo
put(rect(60, 40, 120, 200) + 80) -- rect(140, 120, 200, 280)
```

En Lingo uniquement, l'instruction suivante divise chaque coordonnée d'un rectangle par 3 :

```
-- Syntaxe Lingo
put(rect(60, 40, 120, 200) / 3) -- rect(20, 13, 40, 66)
```

Voir aussi

point(), quad

registerForEvent()

Utilisation

```
member(quelActeur).registerForEvent(nomDévénement, \
   nomDeGestionnaire, objetScript {, début, période, répétitions})
```

Description

Commande 3D ; déclare le gestionnaire spécifié comme étant celui à appeler lorsque l'événement spécifié se produit à l'intérieur de l'acteur spécifié.

Les descriptions de paramètres suivantes s'appliquent aux deux commandes `registerForEvent()` et `registerScript()`.

Remarque : Vous pouvez associer l'enregistrement d'un script à un nœud particulier plutôt qu'à un acteur, à l'aide de la commande `registerScript()`.

Paramètres

`nomDévénement` Requis. Spécifie le nom de l'événement. Il peut s'agir de l'un des événements prédéfinis suivants ou d'un événement personnalisé défini par vous :

- `#collideAny` est un événement de collision.
- `#collideWith` est un événement de collision impliquant ce modèle. La commande `setCollisionCallback()` est un raccourci de la commande `registerScript()` pour l'événement `#collideWith`.
- `#animationStarted` et `#animationEnded` sont des événements de notification utilisés à la lecture ou à l'arrêt d'une animation de segments ou d'images-clés. Le gestionnaire reçoit trois arguments : *`nomDévénement`*, *`mouvement`* et *`position`*. L'argument *`nomDévénement`* a pour valeur `#animationStarted` ou `#animationEnded`. L'argument *`mouvement`* est le nom du mouvement démarré ou arrêté, *`position`* étant la position actuelle du mouvement.
- Pour les animations en boucle, l'événement `#animationStarted` n'est émis que pour la première boucle, pas pour les suivantes. Cet événement est envoyé au début de la fusion entre deux animations.
- Lorsqu'une série d'animations est placée en file d'attente pour le modèle et que la propriété `autoBlend` de l'animation a pour valeur `TRUE`, l'événement `#animationEnded` peut se produire avant la fin apparente d'un mouvement donné. En effet, la propriété `autoBlend` peut encore donner une impression de mouvement alors que l'animation s'est terminée comme prévu.
- `#timeMS` est un événement horaire. Le premier événement `#timeMS` a eu lieu lorsque le nombre de millisecondes spécifié dans le paramètre *`début`* s'est écoulé après l'appel de `registerForEvent`. Le paramètre *`période`* détermine le nombre de millisecondes entre les événements `#timeMS` lorsque la valeur des *`répétitions`* est supérieure à 0. Si la valeur des *`répétitions`* est 0, l'événement `#timeMS` se produit indéfiniment.

nomDeGestionnaire Requis. Spécifie le nom du gestionnaire qui sera appelé lorsque l'événement *nomDévénement* surviendra ; ce gestionnaire se trouve dans l'objet de script indiqué par *objetScript*. Le gestionnaire reçoit les arguments suivants :

- `type` est toujours 0.
- `delta` est le temps (en millisecondes) écoulé depuis le dernier événement `#timeMS`.
- `time` est le nombre de millisecondes écoulées depuis le premier événement `#timeMS`. Par exemple, avec trois itérations d'une période de 500 ms, la première itération sera 0, la deuxième sera 500 et la troisième sera 1000.
- `duration` est le nombre total de millisecondes écoulées entre l'appel `registerForEvent` et le dernier événement `#timeMS`. Par exemple, avec cinq itérations d'une période de 500 ms, la durée est 2500 ms. Pour les tâches avec des itérations illimitées, la durée est 0.
- `systemTime` est la durée absolue, en millisecondes, depuis le début de l'animation Director.

objetScript Requis. Spécifie l'objet de script qui contient le gestionnaire *nomDeGestionnaire*. Si 0 est spécifié pour *objetScript*, c'est le premier gestionnaire d'événement portant le nom donné qui est appelé.

début Facultatif. Spécifie le nombre de millisecondes écoulées entre l'appel de `registerForEvent()` et le premier événement `#timeMS`.

période Facultatif. Spécifie le nombre de millisecondes entre des événements `#timeMS` lorsque la valeur de *répétitions* est supérieure à 0.

répétitions Facultatif. Spécifie le nombre de répétitions pour l'événement `#timeMS`. Si *répétitions* a une valeur égalele à 0, l'événement `#timeMS` se produit indéfiniment.

Exemple

L'instruction suivante enregistre le gestionnaire d'événement `promptUser` détecté dans un script d'animation pour un appel à deux reprises, à un intervalle de cinq secondes :

```
member("Scène").registerForEvent(#timeMS, #promptUser, 0, \
  5000, 5000, 2)
```

L'instruction suivante enregistre le gestionnaire d'événement `promptUser` détecté dans un script d'animation pour un appel à chaque fois qu'une collision a lieu au sein de l'acteur Scène :

```
member("Scène").registerForEvent(#collideAny, #promptUser, 0)
```

L'instruction suivante déclare que le gestionnaire `on promptUser` du même script que celui qui contient la commande `registerForEvent` doit être appelé lorsqu'un objet entre en collision avec le modèle Pluton dans l'acteur nommé Scène :

```
member("Scène").registerForEvent(#collideWith, #promptUser, me, \
  member("Scène").model("Pluton"))
```

Voir aussi

`setCollisionCallback()`, `registerScript()`, `play() (3D)`, `playNext() (3D)`, `autoblend`, `blendTime`, `sendEvent`, `unregisterAllEvents`

registerScript()

Utilisation

```
member(quelActeur).model(quelModèle).registerScript(nomDévénement, \
   nomDeGestionnaire, objetScript {, début, période, répétitions})
member(quelActeur).camera(quelleCaméra).registerScript(nomDévénement, \
   nomDeGestionnaire, objetScript {, début, période, répétitions})
member(quelActeur).light(quelleLumière).registerScript(nomDévénement, \
   nomDeGestionnaire, objetScript {, début, période, répétitions})
member(quelActeur).group(quelGroupe).registerScript(nomDévénement, \
   nomDeGestionnaire, objetScript {, début, période, répétitions})
```

Description

Commande 3D ; enregistre le gestionnaire spécifié comme devant être appelé lorsque l'événement spécifié se produit pour le nœud référencé.

Les descriptions de paramètres suivantes s'appliquent aux deux commandes `registerForEvent()` et `registerScript()`.

Paramètres

`nomDévénement` Requis. Spécifie le nom de l'événement. Il peut s'agir de l'un des événements prédéfinis suivants ou d'un événement personnalisé défini par vous :

- `#collideAny` est un événement de collision.
- `#collideWith` est un événement de collision impliquant ce modèle. La commande `setCollisionCallback()` est un raccourci de la commande `registerScript()` pour l'événement `#collideWith`.
- `#animationStarted` et `#animationEnded` sont des événements de notification utilisés à la lecture ou à l'arrêt d'une animation de segments ou d'images-clés. Le gestionnaire reçoit trois arguments : *`nomDévénement`*, *`mouvement`* et *`position`*. L'argument *`nomDévénement`* a pour valeur `#animationStarted` ou `#animationEnded`. L'argument *`mouvement`* est le nom du mouvement démarré ou arrêté, *`position`* étant la position actuelle du mouvement.

  Pour les animations en boucle, l'événement `#animationStarted` n'est émis que pour la première boucle, pas pour les suivantes. Cet événement est envoyé au début de la fusion entre deux animations.

  Lorsqu'une série d'animations est placée en file d'attente pour le modèle et que la propriété `autoBlend` de l'animation a pour valeur `TRUE`, l'événement `#animationEnded` peut se produire avant la fin apparente d'un mouvement donné. En effet, la propriété `autoBlend` peut encore donner une impression de mouvement alors que l'animation s'est terminée comme prévu.
- `#timeMS` est un événement horaire. Le premier événement `#timeMS` a eu lieu lorsque le nombre de millisecondes spécifié dans le paramètre *`début`* s'est écoulé après l'appel de `registerForEvent`. Le paramètre *`période`* détermine le nombre de millisecondes entre les événements `#timeMS` lorsque la valeur des *`répétitions`* est supérieure à 0. Si la valeur des *`répétitions`* est 0, l'événement `#timeMS` se produit indéfiniment.

nomDeGestionnaire Requis. Spécifie le nom du gestionnaire qui sera appelé lorsque l'événement *nomDévénement* surviendra ; ce gestionnaire se trouve dans l'objet de script indiqué par *objetScript*. Le gestionnaire reçoit les arguments suivants :

- `type` est toujours 0.
- `delta` est le temps (en millisecondes) écoulé depuis le dernier événement `#timeMS`.
- `time` est le nombre de millisecondes écoulées depuis le premier événement `#timeMS`. Par exemple, avec trois itérations d'une période de 500 ms, la première itération sera 0, la deuxième sera 500 et la troisième sera 1000.
- `duration` est le nombre total de millisecondes écoulées entre l'appel `registerForEvent` et le dernier événement `#timeMS`. Par exemple, avec cinq itérations d'une période de 500 ms, la durée est 2500 ms. Pour les tâches avec des itérations illimitées, la durée est 0.
- `systemTime` est la durée absolue, en millisecondes, depuis le début de l'animation Director.

objetScript Requis. Spécifie l'objet de script qui contient le gestionnaire *nomDeGestionnaire*. Si 0 est spécifié pour *objetScript*, c'est le premier gestionnaire d'événement portant le nom donné qui est appelé.

début Facultatif. Spécifie le nombre de millisecondes écoulées entre l'appel de `registerForEvent()` et le premier événement `#timeMS`

période Facultatif. Spécifie le nombre de millisecondes entre des événements `#timeMS` lorsque la valeur de *répétitions* est supérieure à 0.

répétitions Facultatif. Spécifie le nombre de répétitions pour l'événement `#timeMS`. Si *répétitions* a une valeur égalele à 0, l'événement `#timeMS` se produit indéfiniment.

Exemple

L'instruction suivante enregistre le gestionnaire d'événement `messageReçu`, situé dans un script d'animation à appeler lorsque le modèle Lecteur reçoit l'événement personnalisé `#message` défini par l'utilisateur :

```
member("Scène").model("Lecteur").registerScript(#message, \
  #messageReçu, 0)
```

L'instruction suivante enregistre le gestionnaire d'événement `réponseCollision`, situé dans le même script que la commande `registerScript` à appeler chaque fois qu'une collision se produit entre le modèle Lecteur et tout autre modèle utilisant le modificateur `#collision` :

```
member("Scène").model("Lecteur").registerScript(#collideWith, \
  #réponseCollision, me)
```

Voir aussi

`registerForEvent()`, `sendEvent`, `setCollisionCallback()`

removeBackdrop

Utilisation

`member(`*quelActeur*`).camera(`*quelleCaméra*`).removeBackdrop(`*index*`)`

Description

Commande 3D ; supprime le fond situé à une position spécifiée de la liste des fonds de la caméra.

Paramètres

index Requis. Spécifie la position d'index du fond à supprimer de la liste des fonds.

Exemple

L'instruction suivante retire le troisième fond de la liste des fonds de la caméra 1 de l'acteur nommé Scène. Le fond disparaîtra de la scène si des images-objets utilisent actuellement cette caméra.

```
member("Scène").camera[1].removeBackdrop(3)
```

removeFromWorld

Utilisation

```
member(quelActeur).model(quelModèle).removeFromWorld()
member(quelActeur).light(quelleLumière).removeFromWorld()
member(quelActeur).camera(quelleCaméra).removeFromWorld()
member(quelActeur).group(quelGroupe).removeFromWorld()
```

Description

Commande 3D ; pour les modèles, les lumières, les caméras ou les groupes dont la hiérarchie amont se termine dans l'univers, cette commande donne une valeur nulle aux parents et les retire de l'univers.

Pour les objets dont la hiérarchie amont ne se termine pas dans l'univers, cette commande n'a aucun effet.

Paramètres

Aucun.

Exemple

La commande suivante retire le modèle gbCyl de l'univers 3D de l'acteur Scène.

```
member("Scène").model("gbCyl").removeFromWorld()
```

removeLast()

```
member(quelActeur).model(quelModèle).bonesPlayer.removeLast()
member(quelActeur).model(quelModèle).keyframePlayer.\
  removeLast()
```

Description

Commande de modificateur 3D `keyframePlayer` et `bonesPlayer` ; supprime le dernier mouvement de la liste de lecture du modificateur.

Paramètres

Aucun.

Exemple

L'instruction suivante retire le dernier mouvement de la liste de lecture du modificateur `bonesPlayer` pour le modèle Marcheur.

```
member("MonUnivers").model("Marcheur").bonesPlayer.removelast()
```

removeModifier

Utilisation

```
member(quelActeur).model(quelModèle).removeModifier.\
  (#quelModificateur)
```

Description

Commande 3D ; supprime le modificateur spécifié du modèle spécifié.

Cette commande renvoie `TRUE` si elle est exécutée avec succès et `FALSE` si *`#quelModificateur`* n'est pas un modificateur valide ou si le modificateur n'est pas associé au modèle.

Paramètres

`quelModificateur` Requis. Spécifie le modificateur à supprimer.

Exemple

L'instruction suivante retire le modificateur `#toon` du modèle Boîte.

```
member("formes").model("Boîte").removeModifier(#toon)
```

Voir aussi

`addModifier`, `modifier`, `modifier[]`, `modifiers`

removeOverlay

Utilisation

```
member(quelActeur).camera(quelleCaméra).removeOverlay(index)
```

Description

Commande 3D ; retire le recouvrement situé à une position spécifiée de la liste des recouvrements de la caméra.

Paramètres

`index` Requis. Spécifie la position d'index du recouvrement dans la liste des recouvrements.

Exemple

L'instruction suivante retire le premier recouvrement de la liste de recouvrements de la caméra de l'image-objet 5. Le recouvrement disparaîtra de la scène.

```
sprite(5).camera.removeOverlay(1)
```

Voir aussi

`overlay`

removeScriptedSprite()

Utilisation

```
-- Syntaxe Lingo
réfObjPisteDimageObjet.removeScriptedSprite()

// Syntaxe JavaScript
réfObjPisteDimageObjet.removeScriptedSprite();
```

Description

Méthode de piste d'image-objet ; bascule de nouveau le contrôle d'une piste d'image-objet du script vers le scénario.

Paramètres

Aucun.

Exemple

L'instruction suivante supprime l'image-objet scriptée de la piste des images-objets 5 :

```
-- Syntaxe Lingo
channel(5).removeScriptedSprite()

// Syntaxe JavaScript
channel(5).removeScriptedSprite();
```

Voir aussi

makeScriptedSprite(), Piste d'image-objet

resetWorld

Utilisation

```
member(quelActeur).resetWorld()
member(quelActeurTexte).resetWorld()
```

Description

Commande 3D ; redonne aux propriétés de l'acteur 3D référencé les valeurs enregistrées lorsque l'acteur a été mis en mémoire pour la première fois. La propriété `state` de l'acteur doit avoir la valeur 0 (déchargé), 4 (média chargé) ou -1 (erreur) avant que cette commande ne puisse être utilisée afin d'éviter une erreur de script.

Cette commande diffère de `revertToWorldDefaults` dans la mesure où les valeurs utilisées reflètent l'état de l'acteur au moment où il a été mis en mémoire pour la première fois, plutôt qu'au moment où il a été créé.

Paramètres

Aucun.

Exemple

L'instruction suivante redonne aux propriétés de l'acteur Scène les valeurs utilisées lorsque l'acteur a été mis en mémoire pour la première fois.

```
member("Scène").resetWorld()
```

Voir aussi

revertToWorldDefaults

resolveA

Utilisation

collisionData.resolveA(*bRésolution*)

Description

Méthode 3D de collision ; annule le comportement de collision défini par la propriété collision.resolve pour collisionData.modelA. N'appelez cette fonction que si vous souhaitez remplacer le comportement défini pour modelA à l'aide de collision.resolve.

Paramètres

bRésolution Requis. Indique si la collision est résolue pour modelA. Si *bRésolution* est TRUE, la collision est résolue pour modelA ; si *bRésolution* est FALSE, la collision n'est pas résolue pour modelA.

Voir aussi

collisionData, registerScript(), resolve, modelA, setCollisionCallback()

resolveB

Utilisation

collisionData.resolveB(*bRésolution*)

Description

Méthode 3D de collision ; annule le comportement de collision défini par la propriété collision.resolve pour collisionData.modelB. N'appelez cette fonction que si vous souhaitez remplacer le comportement défini pour modelB à l'aide de collision.resolve.

Paramètres

bRésolution Requis. Indique si la collision est résolue pour modelB. Si *bRésolution* est TRUE, la collision est résolue pour modelB ; si *bRésolution* est FALSE, la collision n'est pas résolue pour modelB.

Voir aussi

collisionData, resolve, registerScript(), modelB, setCollisionCallback()

restart()

Utilisation

```
-- Syntaxe Lingo
_system.restart()

// Syntaxe JavaScript
_system.restart();
```

Description

Méthode du système ; ferme toutes les applications ouvertes et redémarre l'ordinateur.

Paramètres

Aucun.

Exemple

L'instruction suivante redémarre l'ordinateur lorsque l'utilisateur appuie sur la combinaison de touches Cmd+R (Macintosh) ou Ctrl+R (Windows) :

```
-- Syntaxe Lingo
if (_key.key = "r" and _key.commandDown) then
  _system.restart()
end if

// Syntaxe JavaScript
if (_key.key = "r" && _key.commandDown) {
  _system.restart();
}
```

Voir aussi

Système

restore()

Utilisation

```
-- Syntaxe Lingo
réfObjFenêtre.restore()

// Syntaxe JavaScript
réfObjFenêtre.restore();
```

Description

Méthode de fenêtre ; restaure une fenêtre lorsqu'elle a été agrandie.

Utilisez cette méthode lorsque vous créez des barres de titre personnalisées pour des animations dans une fenêtre (MIAW).

Paramètres

Aucun.

Exemple

Cette instruction rétablit la fenêtre agrandie intitulée Tableau de commande :

```
-- Syntaxe Lingo
window("Tableau de commande").restore()

// Syntaxe JavaScript
window("Tableau de commande").restore();
```

Voir aussi

maximize(), Fenêtre

result

Utilisation

```
the result
```

Description

Fonction ; affiche la valeur de l'expression renvoyée par le dernier gestionnaire exécuté.

La fonction `result` sert notamment à obtenir des valeurs provenant d'animations lues dans des fenêtres et à suivre l'évolution de Lingo en affichant les résultats des gestionnaires dans la fenêtre Messages pendant la lecture de l'animation.

Pour renvoyer le résultat d'un gestionnaire, affectez ce résultat à une variable, puis vérifiez la valeur de cette dernière. Utilisez une instruction telle que `set myVariable = fonction()`, où *`fonction()`* est le nom d'une fonction spécifique.

Paramètres

Aucun.

Exemple

Le gestionnaire suivant renvoie un résultat aléatoire de deux dés :

```
on lancerLesDés
  return random(6) + random(6)
end
```

Dans l'exemple suivant, les deux instructions

```
lancerLesDés
dés = the result
```

sont équivalentes à l'instruction suivante :

```
set dés = lancerLesDés()
```

L'instruction `set dés = lancerLesDés` n'appelle pas le gestionnaire car il n'existe aucune parenthèse après `lancerLesDés` ; `lancerLesDés` est considéré comme une référence de variable.

Voir aussi

return (mot-clé)

resume()

Utilisation

```
-- Syntaxe Lingo
animGifSpriteRef.resume()

// Syntaxe JavaScript
animGifSpriteRef.resume();
```

Description

Méthode de GIF animé ; reprend la lecture de l'image-objet à partir de l'image suivant celle sur laquelle elle est arrêtée. Cette commande n'a aucun effet si l'image-objet GIF animé n'est pas en pause.

Paramètres

Aucun.

Voir aussi

rewind() (GIF animé, Flash)

returnToTitle()

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.returnToTitle()

// Syntaxe JavaScript
réfObjDvd.returnToTitle();
```

Description

Méthode DVD ; reprend la lecture après l'affichage d'un menu.

Paramètres

Aucun.

Exemple

Cette instruction reprend la lecture après l'affichage d'un menu :

```
-- Lingo syntax
member(1).returnToTitle()

// JavaScript syntax
member(1).returnToTitle();
```

Voir aussi

DVD

revertToWorldDefaults

Utilisation

```
member(quelActeur).revertToWorldDefaults()
```

Description

Commande 3D ; redonne aux propriétés de l'acteur 3D spécifié les valeurs enregistrées lorsque l'acteur a été créé. La propriété `state` de l'acteur doit avoir la valeur 4 (chargé) ou -1 (erreur) avant que cette commande ne puisse être utilisée afin d'éviter une erreur de script.

Cette commande diffère de `resetWorld` dans la mesure où les valeurs utilisées reflètent l'état de l'acteur lorsqu'il a été créé plutôt que lorsqu'il a été mis en mémoire pour la première fois.

Paramètres

Aucun.

Exemple

L'instruction suivante redonne aux propriétés de l'acteur Scène les valeurs enregistrées lorsque l'acteur a été créé.

```
member("Scène").revertToWorldDefaults()
```

Voir aussi

`resetWorld`

rewind() (piste audio)

Utilisation

```
-- Syntaxe Lingo
réfObjPisteAudio.rewind()

// Syntaxe JavaScript
réfObjPisteAudio.rewind();
```

Description

Méthode de piste audio ; interrompt la lecture du son courant et la redémarre à sa position de départ `startTime`.

Si le son est en pause, il reste en pause, avec la propriété `currentTime` définie sur la position de départ `startTime`.

Paramètres

Aucun.

Exemple

L'instruction suivante relance la lecture de l'acteur son dans la piste audio 1 depuis le début.

```
-- Syntaxe Lingo
sound(1).rewind()

// Syntaxe JavaScript
sound(1).rewind();
```

Voir aussi

Piste audio, startTime

rewind() (Windows Media)

Utilisation

```
-- Syntaxe Lingo
réfObjWindowsMedia.rewind()

// Syntaxe JavaScript
réfObjWindowsMedia.rewind();
```

Description

Méthode d'acteur ou d'image-objet Windows Media. Rembobine l'animation jusqu'à la première image d'un acteur ou d'une image-objet Windows Media.

Cette méthode, lorsqu'elle est appelée, n'a aucun effet sur la propriété `mediaStatus`.

Paramètres

Aucun.

Voir aussi

mediaStatus (RealMedia, Windows Media), Windows Media

rewind() (GIF animé, Flash)

Utilisation

```
-- Syntaxe Lingo
animGifSpriteRef.rewind()

// Syntaxe JavaScript
animGifSpriteRef.rewind();
```

Description

Commande ; renvoie une image-objet animation Flash ou GIF animé à l'image 1 lorsque l'image-objet est arrêtée ou en cours de lecture.

Paramètres

Aucun.

Exemple

Le script d'image suivant détermine si l'image-objet animation Flash dans laquelle le comportement était placé est en cours de lecture et, le cas échéant, continue la boucle dans la même image. Lorsque l'animation est terminée, l'image-objet la rembobine (si bien que la première image de l'animation apparaît sur la scène) et permet à la tête de lecture de passer à l'image suivante.

```
-- Syntaxe Lingo
property spriteNum

on exitFrame
  if sprite(spriteNum).playing then
    _movie.go(_movie.frame)
  else
    sprite(spriteNum).rewind()
    _movie.updatestage()
  end if
end

// Syntaxe JavaScript
function exitFrame() {
  var plg = sprite(this.spriteNum).playing;
  if (plg = 1) {
    _movie.go(_movie.frame);
  } else {
    sprite(this.spriteNum).rewind();
    _movie.updatestage();
  }
}
```

rollOver()

Utilisation

```
-- Syntaxe Lingo
_movie.rollOver({intSpriteNum})

// Syntaxe JavaScript
_movie.rollOver({intSpriteNum});
```

Description

Méthode d'animation ; indique si le pointeur (curseur) se trouve sur le rectangle délimitant l'image-objet spécifiée (`TRUE` ou 1) ou non (`FALSE` ou 0).

La méthode `rollOver()`, généralement utilisée dans les scripts d'images, est utile pour créer des gestionnaires qui exécutent une action lorsque l'utilisateur place le pointeur sur une image-objet spécifique.

Si l'utilisateur continue à déplacer la souris, la valeur de `()rollOver` peut changer pendant qu'un script exécute un gestionnaire et peut donner lieu à un comportement inattendu. Vous pouvez vous assurer qu'un gestionnaire utilise une valeur de survol constante en affectant `rollOver()` à une variable au moment de son démarrage.

Lorsque le pointeur survole une zone de la scène où apparaissait précédemment une image-objet, `rollOver()` se produit encore et indique que l'image-objet est toujours présente. Pour éviter ce problème, évitez d'effectuer des survols sur ces emplacements ou placez l'image-objet au-dessus de la barre de menus avant de la supprimer.

Paramètres

intSpriteNum Facultatif. Nombre entier qui spécifie le numéro de l'image-objet.

Exemple

L'instruction suivante change le contenu de l'acteur champ Message en « C'est bien là » lorsque le curseur se trouve sur l'image-objet 6 :

```
-- Syntaxe Lingo
if (_movie.rollOver(6)) then
  member("Message").text = "C'est bien là."
end if

// Syntaxe JavaScript
if (_movie.rollOver(6)) {
  member("Message").text = "C'est bien là.";
}
```

Le gestionnaire suivant positionne la tête de lecture sur d'autres images lorsque le curseur se trouve sur certaines images-objets de la scène. Il affecte d'abord la valeur `rollOver` à une variable. Cela permet au gestionnaire d'utiliser la valeur `rollOver` en vigueur au démarrage du survol, que l'utilisateur continue à déplacer la souris ou non.

```
-- Syntaxe Lingo
on exitFrame
  imageObjetCourante = _movie.rollOver()
  case imageObjetCourante of
    1: _movie.go("Gauche")
    2: _movie.go("Milieu")
    3: _movie.go("Droite")
  end case
end exitFrame

// Syntaxe JavaScript
function exitFrame() {
  var imageObjetCourante = _movie.rollOver();
  switch (imageObjetCourante) {
    cas 1 : _movie.go("Gauche");
      break;
    cas 2 : _movie.go("Milieu");
      break;
    cas 3 : _movie.go("Droite");
      break;
  }
}
```

Voir aussi

Animation

rootMenu()

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.rootMenu()

// Syntaxe JavaScript
réfObjDvd.rootMenu();
```

Description

Méthode de DVD ; affiche le menu racine.

Paramètres

Aucun.

Voir aussi

DVD

rotate

Utilisation

```
member(quelActeur).node(quelNœud).rotate(angleX, angleY, \
  angleZ {, parRapportA})
member(quelActeur).node(quelNœud).rotate(vecteurDeRotation \
  {, parRapportA})
member(quelActeur).node(quelNœud).rotate(position, axe, \
  angle {, parRapportA})
transformation.rotate(angleX, angleY, angleZ {, parRapportA})
transformation.rotate(vecteurDeRotation {, parRapportA})
transformation.rotate(position, axe, angle {, parRapportA})
```

Description

Commande 3D ; applique une rotation après les décalages de position, de rotation et d'échelle de l'objet de transformation d'un nœud ou d'un objet de transformation directement référencé. La rotation doit être spécifiée sous la forme d'un ensemble de trois angles, chacun desquels spécifiant l'angle de rotation autour des trois axes correspondants. Ces angles peuvent être spécifiés de façon explicite sous la forme *angleX*, *angleY* et *angleZ*, ou au moyen d'un *vecteurDeRotation*, où le composant *x* du vecteur correspond à la rotation autour de l'axe des x, *y* autour de l'axe des y et *z* autour de l'axe des z. La rotation peut également être spécifiée autour d'un axe arbitraire passant par un point de l'espace.

Paramètres

angleX Requis si vous appliquez une rotation avec les axes des *x*, des *y* et des *z*. Spécifie l'angle de rotation autour de l'axe des *x*.

angleY Requis si vous appliquez une rotation avec les axes des *x*, des *y* et des *z*. Spécifie l'angle de rotation autour de l'axe des *y*.

angleZ Requis si vous appliquez une rotation avec les axes des *x*, des *y* et des *z*. Spécifie l'angle de rotation autour de l'axe des *z*.

vecteurDeRotation Requis si vous appliquez une rotation avec un vecteur. Spécifie le vecteur qui contient les angles à appliquer.

angle Requis si vous appliquez une rotation autour d'un axe arbitraire qui passe par un point dans l'espace. Spécifie la position dans l'espace.

axe Requis si vous appliquez une rotation autour d'un axe arbitraire qui passe par un point dans l'espace. Spécifie l'axe qui passe par la position *position* spécifiée.

position Requis si vous appliquez une rotation autour d'un axe arbitraire qui passe par un point dans l'espace. Spécifie l'angle de rotation autour de l'axe *axis*.

parRapportA Facultatif. Spécifie les axes du système de coordonnées utilisés pour appliquer les changements de rotation voulus. Le paramètre *parRapportA* peut avoir les valeurs suivantes :

- #self applique les incréments en fonction du système de coordonnées local du nœud (les axes des x, des y et des z spécifiés pour le modèle au cours de la programmation). Cette valeur est utilisée comme valeur par défaut si vous utilisez la commande rotate avec une référence de nœud et que le paramètre *parRapportA* n'est pas spécifié.
- #parent applique les incréments par rapport au système de coordonnées du parent du nœud. Cette valeur est utilisée comme valeur par défaut si vous utilisez la commande rotate avec une référence de transformation et que le paramètre *parRapportA* n'est pas spécifié.
- #world applique les incréments par rapport au système de coordonnées de l'univers. Lorsque le parent d'un modèle est l'univers, ceci est équivalent à l'utilisation de #parent.
- référenceDeNœud permet de spécifier un nœud servant de base à la rotation, la commande appliquant les incréments en fonction du système de coordonnées du nœud spécifié.

Exemple

L'exemple suivant fait d'abord pivoter le modèle Lune autour de son propre axe des z (en le faisant pivoter sur place), puis le fait pivoter autour de son nœud parent, le modèle Terre (ce qui entraîne le déplacement du modèle Lune en orbite autour du modèle Terre).

```
member("Scène").model("Lune").rotate(0,0,15)
member("Scène").model("Lune").rotate(vector(0, 0, 5),
  member("Scène").model("Lune"))
```

L'exemple suivant fait pivoter le modèle Balle autour d'une position de l'espace occupée par le modèle Bâton. L'effet obtenu est le déplacement du modèle Balle en orbite autour du modèle Bâton dans le plan xy.

```
posBâton = member("Scène 3D").model("Bâton").worldPosition
member("Scène 3D").model("Balle").rotate(posBaton, vector(0,0,1), \
  5, #world)
```

Voir aussi

pointAt, preRotate, rotation (transformation), rotation (matériau de gravure), rotation (fond et recouvrement), preScale(), transform (propriété)

runMode

Utilisation

```
the runMode
```

Description

Fonction ; renvoie une chaîne indiquant le mode de lecture de l'animation. Les valeurs possibles sont :

- Author – L'animation est lue dans Director.

- `Projector` – L'animation est lue en tant que projection.
- `BrowserPlugin` – L'animation est lue en tant que module Shockwave Player ou autre environnement de programmation tel que LiveConnect ou ActiveX.

Le moyen le plus sûr de tester des valeurs particulières de cette propriété est d'utiliser l'opérateur `contains`. Cela évite les erreurs et permet les correspondances partielles.

Paramètres

Aucun.

Exemple

L'instruction suivante détermine si des paramètres externes sont disponibles et, le cas échéant, les obtient :

```
-- Syntaxe Lingo
if the runMode contains "Plugin" then
  -- décoder le paramètre embed
  if externalParamName(swURL) = swURL then
    put externalParamValue(swURL) into maVariable
  end if
end if

// Syntaxe JavaScript
if (_movie.runMode.indexOf("Plugin") >=0) {
  // décoder le paramètre embed
  if (externalParamName(swURL) == swURL) {
    maVariable = externalParamValue (swURL);
  }
}
```

Voir aussi

`environmentPropList`, `platform`

save castLib

Utilisation

```
castLib(quelleDistribution).save()
save castLib quelleDistribution {,nomDuChemin&nomDeNouveauFichier}
```

Description

Commande ; enregistre les modifications apportées à l'acteur dans son fichier d'origine ou dans un nouveau fichier. Les opérations ou les références ultérieures à la distribution utilisent l'acteur enregistré.

Cette commande ne fonctionne pas avec les fichiers compressés.

La commande `save CastLib` ne prend pas en charge les adresses URL comme références de fichier.

Paramètres

`nomDuChemin&nomDeNouveauFichier` Facultatif. Spécifie le chemin d'accès et le nom du fichier dans lequel vous voulez enregistrer les modifications. Si ce paramètre est omis, l'acteur d'origine doit être lié.

Exemple

L'instruction suivante demande à Director d'enregistrer la version révisée de la distribution Boutons dans le nouveau fichier BoutonsActualisés au sein du même dossier :

```
castLib("Boutons").save(the moviePath & "BoutonsActualisés.cst")
```

Voir aussi

@ (chemin d'accès)

saveMovie()

Utilisation

```
-- Syntaxe Lingo
_movie.saveMovie({chaîneCheminDeFichier})

// Syntaxe JavaScript
_movie.saveMovie({chaîneCheminDeFichier});
```

Description

Méthode d'animation ; enregistre l'animation courante.

Le fait d'inclure le paramètre *chaîneCheminDeFichier* facultatif permet d'enregistrer l'animation dans le fichier spécifié. Cette commande ne fonctionne pas avec les fichiers compressés. Le nom du fichier spécifié doit comprendre l'extension .dir.

La commande `saveMovie()` ne supporte pas les adresses URL comme références de fichier.

Paramètres

chaîneCheminDeFichier Facultatif. Chaîne qui spécifie le chemin d'accès et le nom du fichier dans lequel l'animation est enregistrée.

Exemple

L'instruction suivante enregistre l'animation courante dans le fichier MiseAjour :

```
-- Syntaxe Lingo
_movie.saveMovie(_movie.path & "MiseAjour.dir")

// Syntaxe JavaScript
_movie.saveMovie(_movie.path + "MiseAjour.dir");
```

Voir aussi

Animation

scale (commande)

Utilisation

```
member(quelActeur).node(quelNœud).scale(échelleX, échelleY, \
  échelleZ)
member(quelActeur).node(quelNœud).scale(échelleUnifome)
transformation.scale(échelleX, échelleY, échelleZ)
transformation.scale(échelleUniforme)
```

Description

Commande 3D de transformation ; applique un redimensionnement après les décalages de position, de rotation et d'échelle courants d'une transformation d'un nœud référencé ou de la transformation directement référencée. Le redimensionnement doit être spécifié soit comme un groupe de trois redimensionnements individuels des axes correspondants, soit comme un redimensionnement unique à appliquer à tous les axes. Vous pouvez spécifier un redimensionnement individuel à l'aide des paramètres *échelleX*, *échelleY* et *échelleZ*, ou spécifier une valeur pour un redimensionnement uniforme à l'aide du paramètre *échelleUniforme*.

Un nœud peut être un objet de caméra, groupe, lumière ou modèle. L'utilisation de la commande `scale` ajuste la propriété `transform.scale` du nœud référencé, mais n'a aucun effet visuel sur les lumières ou les caméras car elles ne contiennent pas de géométrie.

Les valeurs du redimensionnement doivent être supérieures à zéro.

Paramètres

échelleX Requis si vous spécifiez trois redimensionnements. Spécifie l'échelle sur l'axe des *x*.

échelleY Requis si vous spécifiez trois redimensionnements. Spécifie l'échelle sur l'axe des *y*.

échelleZ Requis si vous spécifiez trois redimensionnements. Spécifie l'échelle sur l'axe des *z*.

échelleUnifome Requis si vous spécifiez un seul redimensionnement, uniforme. Spécifie le redimensionnement uniforme.

Exemple

L'exemple suivant affiche d'abord la propriété `transform.scale` du modèle Lune, redimensionne ensuite le modèle à l'aide de la commande `scale`, puis affiche la valeur `transform.scale` résultante.

```
put member("Scène").model("Lune").transform.scale
-- vector( 1.0000, 1.0000, 1.0000 )
member("Scène").model("Lune").scale(2.0,1.0,0.5)
put member("Scène").model("Lune").transform.scale
-- vector( 2.0000, 1.0000, 0.5000 )
```

L'instruction suivante redimensionne le modèle nommé Pluton de façon uniforme sur les trois axes selon la valeur `0.5`, ce qui réduit de moitié la taille du modèle affiché.

```
member("Scène").model("Pluton").scale(0.5)
```

L'instruction suivante redimensionne le modèle Pluton de façon non uniforme, en le modifiant sur l'axe des *z* mais pas sur celui des *x* ou des *y*.

```
member("Scène").model("Pluton").scale(0.0, 0.0, 0.5)
```

Voir aussi

`transform (propriété)`, `preScale()`, `scale (transformation)`

script()

Utilisation

```
-- Syntaxe Lingo
script(nomOuNumDacteur {, nomOuNumDeDistribution})

// Syntaxe JavaScript
script(nomOuNumDacteur {, nomOuNumDeDistribution});
```

Description

Fonction du niveau supérieur ; crée une référence à un acteur donné qui contient un script et, éventuellement, spécifie la distribution qui contient l'acteur.

Une erreur est renvoyée si l'acteur donné ne contient aucun script ou s'il n'existe pas.

Paramètres

nomOuNumDacteur Requis. Chaîne qui spécifie le nom de l'acteur contenant un script ou nombre entier qui spécifie la position d'index de cet acteur contenant le script.

nomOuNumDeDistribution Facultatif. Chaîne qui spécifie le nom de la distribution contenant l'acteur *nomOuNumDacteur* ou nombre entier qui spécifie la position d'index de la distribution contenant l'acteur *nomOuNumDacteur*. Si ce paramètre est omis, `script()` lance la recherche dans la première distribution.

Exemple

En Lingo uniquement, les instructions suivantes vérifient si un objet enfant est une instance du script parent Fourmi :

```
-- Syntaxe Lingo
if (objetInsecte.script = script("Fourmi")) then
  objetInsecte.attaque()
end if
```

Cette instruction affecte la variable `acteurAction` à l'acteur de script Actions :

```
-- Syntaxe Lingo
acteurAction = script("Actions")

// Syntaxe JavaScript
var acteurAction = script("Actions");
```

scrollByLine()

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.scrollByLine(quantité)

// Syntaxe JavaScript
réfObjActeur.scrollByLine(quantité);
```

Description

Commande ; fait défiler l'acteur champ ou texte spécifié vers le haut ou le bas du nombre de lignes spécifié. Les lignes sont définies comme des lignes séparées par des retours chariot ou par des retours à la ligne automatiques.

Paramètres

quantité Requis. Spécifie le nombre de lignes à faire défiler. Lorsque *quantité* a une valeur positive, le champ défile vers le bas. Lorsque *quantité* a une valeur négative, le champ défile vers le haut.

Exemple

L'instruction suivante fait défiler l'acteur champ Nouvelles du jour de cinq lignes vers le bas :

```
-- Syntaxe Lingo
member("Nouvelles du jour").scrollbyline(5)

// Syntaxe JavaScript
member("Nouvelles du jour").scrollbyline(5);
```

L'instruction suivante fait défiler l'acteur champ Nouvelles du jour de cinq lignes vers le haut :

```
-- Syntaxe Lingo
member("Nouvelles du jour").scrollByLine(-5)

// Syntaxe JavaScript
member("Nouvelles du jour").scrollByLine(-5);
```

scrollByPage()

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.scrollByPage(quantité)

// Syntaxe JavaScript
réfObjActeur.scrollByPage(quantité);
```

Description

Commande ; fait défiler l'acteur champ ou texte spécifié vers le haut ou le bas d'un nombre de pages spécifié. Une page correspond au nombre de lignes de texte visibles à l'écran.

Paramètres

quantité Requis. Spécifie le nombre de pages à faire défiler. Lorsque *quantité* a une valeur positive, le champ défile vers le bas. Lorsque *quantité* a une valeur négative, le champ défile vers le haut.

Exemple

L'instruction suivante fait défiler l'acteur champ Nouvelles du jour d'une page vers le bas :

```
-- Syntaxe Lingo
member("Nouvelles du jour").scrollbypage(1)

// Syntaxe JavaScript
member("Nouvelles du jour").scrollbypage(1);
```

L'instruction suivante fait défiler l'acteur champ Nouvelles du jour d'une page vers le haut :

```
-- Syntaxe Lingo
member("Nouvelles du jour").scrollbypage(-1)

// Syntaxe JavaScript
member("Nouvelles du jour").scrollbypage(-1);
```

Voir aussi

scrollTop

seek()

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.seek(millisecondes)

// Syntaxe JavaScript
réfObjActeurOuImageObjet.seek(millisecondes);
```

Description

Méthode d'acteur ou d'image-objet RealMedia ; change l'emplacement de lecture du flux multimédia et sélectionne l'emplacement spécifié par le nombre de millisecondes écoulées depuis le début du flux. La valeur `mediaStatus` devient généralement `#seeking`, puis `#buffering`.

Vous pouvez utiliser cette méthode pour initialiser la lecture à des points autres que le début du flux RealMedia, ou pour avancer ou reculer dans le flux. Le nombre entier spécifié dans *millisecondes* correspond au nombre de millisecondes écoulées depuis le début du flux. Ainsi, pour reculer, vous devez spécifier un nombre inférieur de millisecondes, et non un nombre négatif.

Si la commande `seek` est appelée lorsque `mediaStatus` a pour valeur `#paused`, le flux repasse en mémoire tampon et revient à la valeur `#paused` au nouvel emplacement spécifié par `seek`. Si `seek` est appelé lorsque `mediaStatus` a pour valeur `#playing`, le flux repasse en tampon et sa lecture démarre automatiquement au nouvel emplacement du flux. Si `seek` est appelé lorsque `mediaStatus` a pour valeur `#closed`, rien ne se passe.

Si vous tentez de lancer une recherche au-delà de la valeur `duration` du flux, l'argument entier spécifié est ajouté à la plage à partir de 0 pour la durée du flux. Vous ne pouvez pas accéder directement à une image-objet RealMedia qui est en cours de lecture en flux continu.

L'instruction `x.seek(n)` est identique à l'instruction `x.currentTime = n` ; dans les deux cas, le flux sera replacé en mémoire tampon.

Paramètres

millisecondes Requis. Un nombre entier spécifiant le nombre de millisecondes à partir du début du flux.

Exemple

Les exemples suivants définissent la position de lecture actuelle du flux sur 10 000 millisecondes (10 secondes) :

```
-- Syntaxe Lingo
sprite(2).seek(10000)
member("Real").seek(10000)

// Syntaxe JavaScript
sprite(2).seek(10000);
member("Real").seek(10000);
```

Voir aussi

duration (RealMedia, SWA), currentTime (RealMedia), play() (RealMedia, SWA, Windows Media), pause() (RealMedia, SWA, Windows Media), stop() (RealMedia, SWA, Windows Media), mediaStatus (RealMedia, Windows Media)

selectAtLoc()

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.selectAtLoc(point(x, y))

// Syntaxe JavaScript
réfObjDvd.selectAtLoc(point(x, y));
```

Description

Méthode de DVD ; déclenche le bouton sous un point spécifié.

Cette méthode a la même fonctionnalité qu'un clic de souris sur un bouton.

Paramètres

point(x, y) Requis. Point qui spécifie l'emplacement sous lequel un bouton est déclenché.

Exemple

Cette instruction déclenche le bouton sous un point spécifié :

```
-- Syntaxe Lingo
member(10).selectAtLoc(point(50, 75))

// Syntaxe JavaScript
member(10).selectAtLoc(point(50, 75));
```

Voir aussi

DVD

selectButton()

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.selectButton(entBouton)

// Syntaxe JavaScript
réfObjDvd.selectButton(entBouton);
```

Description

Méthode de DVD ; sélectionne un bouton spécifié.

Cette méthode renvoie `TRUE` (1) en cas de réussite.

Paramètres

`entBouton` Requis. Nombre entier qui spécifie le bouton à sélectionner.

Exemple

Cette instruction sélectionne le bouton 5 :

```
-- Syntaxe Lingo
sprite(11).selectButton(5)

// Syntaxe JavaScript
sprite(11).selectButton(5);
```

Voir aussi

`DVD`

selectButtonRelative()

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.selectButtonRelative(direction)

// Syntaxe JavaScript
réfObjDvd.selectButtonRelative(direction);
```

Description

Méthode de DVD ; sélectionne un bouton par rapport à la position du bouton courant dans le menu.

Paramètres

`direction` Requis. Symbole (Lingo) ou chaîne (syntaxe JavaScript) qui spécifie la direction du déplacement par rapport à la position du bouton courant. Les valeurs correctes sont `left` et `right`.

Exemple

Cette instruction spécifie le bouton qui se trouve à gauche du bouton actuel :

```
-- Syntaxe Lingo
sprite(12).member.selectButtonRelative(#left)

// Syntaxe JavaScript
sprite(12).member.selectButtonRelative("left");
```

Voir aussi

DVD

selection() (fonction)

Utilisation

```
the selection
```

Description

Fonction ; renvoie une chaîne de caractères contenant la partie sélectionnée du champ modifiable courant. Elle permet notamment de tester la sélection faite par l'utilisateur dans un champ.

La fonction `selection` indique uniquement la chaîne de caractères sélectionnée ; vous ne pouvez pas utiliser `selection` pour sélectionner une chaîne de caractères.

Paramètres

Aucun.

Exemple

L'instruction suivante vérifie si des caractères sont sélectionnés et, à défaut, affiche le message d'alerte « Veuillez sélectionner un mot. » :

```
if the selection = EMPTY then alert "Veuillez sélectionner un mot."
```

Voir aussi

selStart, selEnd

sendAllSprites()

Utilisation

```
-- Syntaxe Lingo
_movie.sendAllSprites(chaîneMessageDévénement {, args})

// Syntaxe JavaScript
_movie.sendAllSprites(chaîneMessageDévénement {, args});
```

Description

Méthode d'animation ; envoie un message d'événement personnalisé à toutes les images-objets, et pas seulement à l'image-objet impliquée dans l'événement. Comme tout autre message, celui-ci est envoyé à chaque script associé à l'image-objet, à moins que la méthode `stopEvent()` ne soit utilisée.

Pour de meilleurs résultats, n'envoyez le message qu'aux images-objets capables de le gérer correctement par le biais de la méthode `sendSprite()`. Aucune erreur ne se produira si le message est envoyé à toutes les images-objets, mais cela nuira néanmoins à la performance. La présence d'un gestionnaire identique dans un comportement donné et pour des images-objets différentes risquant également de poser des problèmes, il est important d'utiliser des noms uniques pour les messages qui seront diffusés, ceci afin d'éviter tout conflit éventuel.

Une fois le message passé à tous les comportements, l'événement suit la hiérarchie de messages classique: script d'acteur, script d'image, puis script d'animation.

Lorsque vous utilisez la méthode `sendAllSprites()`, effectuez les opérations suivantes :

- Remplacez *`chaîneMessageDévénement`* par le message.
- Remplacez *`arguments`* par tout argument devant être envoyé avec le message.

Si aucune image-objet ne possède de comportement associé contenant le gestionnaire donné, `sendAllSprites() renvoie la valeur` FALSE.

Paramètres

`chaîneMessageDévénement` Requis. Chaîne qui spécifie le message à envoyer à toutes les images-objets.

`args` Facultatif. Argument ou arguments à envoyer avec le message.

Exemple

Le gestionnaire suivant envoie le message personnalisé `toutesLesImagesObjetsDoiventAugmenterLeCompteur` et l'argument `2` à toutes les images-objets lorsque l'utilisateur clique avec la souris :

```
-- Syntaxe Lingo
on mouseDown me
  _movie.sendAllSprites(#toutesLesImagesObjetsDoiventAugmenterLeCompteur, 2)
end

// Syntaxe JavaScript
function mouseDown() {
  _movie.sendAllSprites("toutesLesImagesObjetsDoiventAugmenterLeCompteur",
  2);
}
```

Voir aussi

`Animation`, `sendSprite()`, `stopEvent()`

sendEvent

Utilisation

`member(`*`quelActeur`*`).sendEvent(`*`#nomDévénement, arg1, arg2,...`*`)`

Description

Commande 3D ; envoie un événement et un nombre arbitraire d'arguments à tous les scripts enregistrés pour recevoir l'événement. Utilisez `registerForEvent()` ou `setCollisionCallback()` pour enregistrer les scripts pour les événements.

Paramètres

nomDévénement Requis. Spécifie le nom de l'événement à envoyer.

arg1, arg2, ... Requis. Un ou plusieurs arguments qui sont envoyés avec l'événement *nomDévénement*.

Exemple

La première ligne de cet exemple crée une instance d'un script parent appelé "Testeur". La seconde ligne définit le gestionnaire de l'instance de script, sautPluton, comme gestionnaire à appeler lorsque l'événement #saut est envoyé. La troisième ligne définit le gestionnaire d'un script d'animation sautMars comme un autre gestionnaire à appeler lorsque l'événement #saut est envoyé. La quatrième ligne envoie l'événement #saut. Le gestionnaire #sautMars d'un script d'animation et le gestionnaire #sautPluton sont appelés, ainsi que tout autre gestionnaire enregistré pour l'événement #saut. Une valeur d'instance de script de 0 indique que vous enregistrez le gestionnaire d'un script d'animation, par opposition au gestionnaire d'une instance de comportement ou de l'enfant d'un script parent.

```
t = new (script "Testeur")
member("Scène").registerForEvent(#saut, #sautPluton, t)
member("Scène").registerForEvent(#saut, #sautMars, 0)
member("Scène").sendEvent(#saut)
```

Voir aussi

registerScript(), registerForEvent(), setCollisionCallback()

sendSprite()

Utilisation

```
-- Syntaxe Lingo
_movie.sendSprite(nomOuNumDimageObjet, événement {, args})

// Syntaxe JavaScript
_movie.sendSprite(nomOuNumDimageObjet, événement {, args});
```

Description

Méthode d'animation ; envoie un message à tous les scripts associés à une image-objet spécifiée.

Les messages envoyés à l'aide de sendSprite() sont envoyés à chacun des scripts associés à l'image-objet. Ces messages suivent ensuite la hiérarchie de messages classique : script d'acteur, script d'image et script d'animation.

Si l'image-objet donnée ne possède pas de comportement associé contenant le gestionnaire donné, sendSprite() renvoie la valeur FALSE.

Paramètres

nomOuNumDimageObjet Requis. Chaîne ou nombre entier qui spécifie le nom ou le numéro de l'image-objet qui recevra l'événement.

événement Requis. Symbole ou chaîne qui spécifie l'événement à envoyer à l'image-objet spécifiée.

args Facultatif. Argument ou arguments à envoyer avec le message.

Exemple

Le gestionnaire suivant envoie le message personnalisé `augmenterLeCompteur` et l'argument `2` à l'image-objet 1 lorsque l'utilisateur clique avec la souris :

```
-- Syntaxe Lingo
on mouseDown me
  _movie.sendSprite(1, #augmenterLeCompteur, 2)
end

// Syntaxe JavaScript
function mouseDown() {
  _movie.sendSprite(1, "augmenterLeCompteur", 2);
}
```

Voir aussi

`Animation`

setAlpha()

Utilisation

`objetImage``.setAlpha(`*`niveauAlpha`*`)`
`objetImage``.setAlpha(`*`objetImageAlpha`*`)`

Description

Fonction ; donne à la couche alpha d'une image-objet un niveauAlpha plat ou un *`objetImageAlpha`* existant. Le *`niveauAlpha`* doit être un nombre compris entre 0 et 255. Des valeurs inférieures font apparaître l'image plus transparente. Des valeurs supérieures font apparaître l'image plus opaque. La valeur 255 a le même effet que la valeur 0. Pour que le *`niveauAlpha`* puisse prendre effet, la propriété `useAlpha()` de l'image-objet doit être définie sur `TRUE`.

L'objet image doit être de 32 bits. Si vous spécifiez un objet image alpha, il doit être de 8 bits. Les deux images doivent avoir les mêmes dimensions. Si ces conditions ne sont pas remplies, `setAlpha()` n'a aucun effet et renvoie la valeur FALSE. La fonction renvoie TRUE si elle réussit.

Exemple

L'instruction Lingo suivante rend l'image de l'acteur bitmap Premier plan opaque et désactive simultanément la couche alpha. Cette méthode est efficace pour supprimer la couche alpha d'une image :

```
member("Premier plan").image.setAlpha(255)
member("Premier plan").image.useAlpha = FALSE
```

L'instruction Lingo suivante récupère la couche alpha de l'acteur Lever de soleil et la place dans la couche alpha de l'acteur Coucher de soleil :

```
alphaTemp = member("Lever de soleil").image.extractAlpha()
member("Coucher de soleil").image.setAlpha(alphaTemp)
```

Voir aussi

`useAlpha`, `extractAlpha()`

setaProp

Utilisation

```
setaProp list, propriétéDeListe, nouvelleValeur
setaProp (objetEnfant, propriétéDeListe, nouvelleValeur)
liste.propriétéDeListe = nouvelleValeur
liste[propriétéDeListe] = nouvelleValeur
objetEnfant.propriétéDeListe = nouvelleValeur
```

Description

Commande ; remplace la valeur affectée à *propriétéDeListe* par la valeur spécifiée par *nouvelleValeur*. La commande `setaProp` fonctionne uniquement avec des listes de propriétés et des objets enfants. L'utilisation de `setaProp` avec une liste linéaire produit une erreur de script.

- Pour les listes de propriétés, `setaProp` remplace une propriété dans la liste spécifiée par *liste*. Lorsque la propriété ne se trouve pas déjà dans la liste, le code ajoute la nouvelle propriété et sa valeur.
- Pour les objets enfants, `setaProp` remplace une propriété de l'objet enfant. Lorsque la propriété ne se trouve pas déjà dans l'objet, le code ajoute la nouvelle propriété et sa valeur.
- La commande `setaProp` peut également définir des propriétés `ancestor`.

Paramètres

propriétéDeListe Requis. Symbole (Lingo uniquement) ou chaîne qui spécifie le nom de la propriété dont la valeur change.

nouvelleValeur Requis. Nouvelle valeur de la propriété *propriétéDeListe*.

Exemple

Les instructions suivantes créent une liste de propriétés, puis ajoutent l'élément `#c:10` à la liste :

```
nouvelleListe = [#a:1, #b:5]
put nouvelleListe
-- [#a:1, #b:5]
setaProp nouvelleListe, #c, 10
put nouvelleListe
```

L'opérateur point permet de modifier la valeur de propriété d'une propriété figurant déjà dans une liste sans utiliser `setaProp` :

```
nouvelleListe = [#a:1, #b:5]
put nouvelleListe
-- [#a:1, #b:5]
nouvelleListe.b = 99
put nouvelleListe
-- [#a:1, #b:99]
```

Remarque : Pour pouvoir utiliser l'opérateur point en vue de manipuler une propriété, la propriété doit déjà exister dans la liste, dans l'objet enfant ou dans le comportement.

Voir aussi

`ancestor`, `property`, `. (opérateur point)`

setAt

Utilisation

```
setAt liste, numéroDordre, valeur
liste[numéroDordre] = valeur
```

Description

Commande ; remplace l'élément spécifié par *numéroDordre* par la valeur spécifiée par *valeur* dans la liste spécifiée par *liste*. Lorsque *numéroDordre* est supérieur au nombre d'éléments d'une liste de propriétés, la commande `setAt` renvoie une erreur de script. Lorsque *numéroDordre* est supérieur au nombre d'éléments d'une liste linéaire, Director insère des entrées vierges dans la liste pour ajouter le nombre d'emplacements spécifiés par *numéroDordre*.

Exemple

Le gestionnaire suivant affecte un nom à la liste [12, 34, 6, 7, 45], remplace le quatrième élément de la liste par la valeur 10, puis affiche le résultat dans la fenêtre Messages :

```
on enterFrame
  set vNombres = [12, 34, 6, 7, 45]
  setAt vNombres, 4, 10
  put vNombres
end enterFrame
```

Lorsque le gestionnaire est exécuté, la fenêtre Messages affiche le message suivant :

```
[12, 34, 6, 10, 45]
```

La même opération peut être exécutée en utilisant des crochets d'accès de la manière suivante :

```
on enterFrame
  set vNombres = [12, 34, 6, 7, 45]
  vNombres[4] = 10
  put vNombres
end enterFrame
```

Lorsque le gestionnaire est exécuté, la fenêtre Messages affiche le message suivant :

```
[12, 34, 6, 10, 45]
```

Voir aussi

`[ ] (crochets d'accès)`

setCallback()

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.setCallback(objetAS, nomDévénAS, #gestLingo, \
  objetLingo)

// Syntaxe JavaScript
réfObjImageObjet.setCallback(objetAS, nomDévénAS, symbol(gestLingo),
  objetLingo);
```

Description

Commande Flash ; cette commande peut être utilisée en tant qu'image-objet ou comme méthode globale pour définir un appel de gestionnaire Lingo pour un événement particulier généré par l'objet spécifié. Lorsque ActionScript déclenche l'événement dans l'objet, cet événement est redirigé vers le gestionnaire Lingo spécifié, avec tous les arguments transmis avec l'événement.

Si l'objet ActionScript a été initialement créé dans une image-objet Flash, utilisez la syntaxe *réfDimageObjetFlash*. Si l'objet a été créé globalement, utilisez la syntaxe globale.

Remarque : Si vous n'avez pas importé d'acteur Flash, vous devrez ajouter manuellement l'Xtra Flash à la liste des Xtras de votre animation pour permettre aux objets Flash globaux de fonctionner correctement. Vous pouvez ajouter les Xtras à la liste des Xtras en choisissant Modification › Animation › Xtras. Pour plus d'informations sur la gestion des Xtras pour les animations distribuées, consultez les rubriques Utilisation de Director dans le panneau d'aide de Director.

Paramètres

objetAS Requis. Spécifie l'objet ActionScript qui contient l'événement *nomDévénAS*.

nomDévénAS Requis. Spécifie l'événement ActionScript qui est survenu.

gestLingo Requis. Spécifie le gestionnaire Lingo qui prend en charge l'événement *nomDévénAS*.

objetLingo Requis. Spécifie l'objet script Lingo qui contient le gestionnaire *gestLingo*.

Exemple

L'instruction suivante définit l'appel du gestionnaire Lingo appelé `monOnStatus` dans l'objet de script Lingo `me` lorsqu'un événement `onStatus` est généré par l'objet ActionScript `tObjetConnexionLocale` dans l'animation Flash, au niveau de l'image-objet 3 :

```
-- Syntaxe Lingo
sprite(3).setCallback(tObjetConnexionLocale, "onStatus", #monOnStatus, me)

// Syntaxe JavaScript
sprite(3).setCallback(tObjetConnexionLocale, "onStatus",
  symbol("monOnStatus"), me);
```

Les instructions suivantes créent un nouvel objet XML global, ainsi qu'un gestionnaire d'appel qui analyse les données XML à leur arrivée. La troisième ligne entraîne le chargement d'un fichier XML. Le gestionnaire d'appel est inclus.

```
-- Syntaxe Lingo
gXMLCB = newObject("XML")
setCallback( gXMLCB, "onDonnées", #donnéesTrouvées, 0 )
gXMLCB.load( "monFichier.xml" )

-- gestionnaire d'appel appelé lors de l'arrivée des données xml
on donnéesTrouvées me, obj, source
   obj.parseXML(source)
   obj.loaded = 1
   obj.onload(TRUE)
end donnéesTrouvées

// Syntaxe JavaScript
gXMLCB = newObject("XML");
setCallback( gXMLCB, "onDonnées", symbol("donnéesTrouvées"), 0 );
gXMLCB.load( "monFichier.xml" );

// gestionnaire d'appel appelé lors de l'arrivée des données xml
function donnéesTrouvées(me, obj, source) {
   obj.parseXML(source);
   obj.loaded = 1;
   obj.onload(1);
}
```

Voir aussi

`newObject()`, `clearAsObjects()`

setCollisionCallback()

Utilisation

```
member(quelActeur).model(quelModèle).collision.\
  setCollisionCallback (#nomDeGestionnaire, instanceDeScript)
```

Description

Commande de collision 3D ; enregistre un gestionnaire spécifié, dans une instance de script donnée, à appeler lorsque *quelModèle* est impliqué dans une collision.

Cette commande ne fonctionne que si la propriété `collision.enabled` du modèle a pour valeur `TRUE`. Le comportement par défaut est déterminé par la valeur de `collision.resolve`, que vous pouvez remplacer en utilisant les commandes `collision.resolveA` et/ou `collision.resolveB`. N'utilisez pas la commande `updateStage` dans le gestionnaire spécifié.

Cette commande est une alternative plus courte à la commande `registerScript` pour les collisions, et le résultat n'est globalement pas différent. Cette commande peut être envisagée pour exécuter une petite partie de la fonctionnalité de la commande `registerScript`.

Paramètres

nomDeGestionnaire Requis. Spécifie le gestionnaire appelé lorsqu'un modèle est impliqué dans une collision.

instanceDeScript Requis. Spécifie l'instance de script qui contient le gestionnaire spécifié par *nomDeGestionnaire*.

Exemple

L'instruction suivante entraîne l'appel du gestionnaire `#rebond` de l'acteur scriptDeCollision lorsque le modèle Sphère entre en collision avec un autre modèle.

```
member("Univers 3D").model("Sphère").collision.\
  setCollisionCallback\
(#rebond, member("scriptDeCollision"))
```

Voir aussi

`collisionData`, `collision (modificateur)`, `resolve`, `resolveA`, `resolveB`, `registerForEvent()`, `registerScript()`, `sendEvent`

setFilterMask()

Utilisation

```
-- Syntaxe Lingo
réfObjFileio.setFilterMask(chaîneMasque)

// Syntaxe JavaScript
réfObjFileio.setFilterMask(chaîneMasque);
```

Description

Méthode de Fileio ; définit le masque de filtrage pour le champ *Type* d'une boîte de dialogue pour spécifier le type des fichiers affichés à l'ouverture de la boîte de dialogue.

Paramètres

chaîneMasque Requis. Chaîne qui spécifie le masque de filtrage.

Voir aussi

`Fileio`

setFinderInfo()

Utilisation

```
-- Syntaxe Lingo
réfObjFileio.setFinderInfo(chaîneAttrs)

// Syntaxe JavaScript
réfObjFileio.setFinderInfo(chaîneAttrs)
```

Description

Méthode de Fileio (Macintosh uniquement) ; définit les informations du Finder pour un fichier ouvert.

Paramètres

chaîneAttrs Requis. Chaîne qui spécifie les informations du Finder.

Voir aussi

`Fileio`

setFlashProperty()

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.setFlashProperty(nomDeCible, #propriété, nouvelleValeur)

// Syntaxe JavaScript
réfObjImageObjet.setFlashProperty(nomDeCible, symbol(propriété),
  nouvelleValeur);
```

Description

Fonction ; permet à Lingo d'invoquer la fonction script d'action Flash `setProperty()` dans l'image-objet Flash donnée. Utilisez la fonction `setFlashProperty()` pour définir les propriétés des propriétés des clips ou des niveaux dans une animation Flash. Cette fonction est similaire à la définition des propriétés d'images-objets dans Director.

Pour définir une propriété globale de l'image-objet Flash, passez une ligne vide comme *`nomDeCible`*. Vous pouvez définir les propriétés Flash globales suivantes : `#focusRect` et `#spriteSoundBufferTime`.

Consultez la documentation de Flash pour plus d'informations sur ces propriétés.

Paramètres

`nomDeCible` Requis. Spécifie le nom du clip ou du niveau de l'animation dont vous souhaitez définir la propriété dans l'image-objet Flash donnée.

`propriété` Requis. Spécifie le nom de la propriété à définir. Vous pouvez définir les propriétés suivantes : `#posX`, `#posY`, `#scaleX`, `#scaleY`, `#visible`, `#rotate`, `#alpha` et `#name`.

`nouvelleValeur` Requis. Spécifie la nouvelle valeur.

Exemple

L'instruction suivante définit la valeur de la propriété `#rotate` du clip Etoile dans l'acteur Flash de l'image-objet 3 sur 180.

```
-- Syntaxe Lingo
sprite(3).setFlashProperty("Etoile", #rotate, 180)

// Syntaxe JavaScript
sprite(3).setFlashProperty("Etoile", symbol("rotate"), 180);
```

Voir aussi

`getFlashProperty()`

setNewLineConversion()

Utilisation

```
-- Syntaxe Lingo
réfObjFileio.setNewLineConversion(entActivéDésactivé)

// Syntaxe JavaScript
réfObjFileio.setNewLineConversion(entActivéDésactivé)
```

Description

Méthode de Fileio (Macintosh uniquement) ; spécifie si la conversion automatique des caractères de changement de ligne est activée ou désactivée.

Paramètres

entActivéDésactivé Requis. Nombre entier qui spécifie si la conversion automatique est activée ou désactivée. Les valeurs correctes incluent 0 (désactivé) et 1 (activé).

Voir aussi

Fileio

setPixel()

Utilisation

```
-- Syntaxe Lingo
réfObjImage.setPixel(x, y, objOuEntierCouleur)
réfObjImage.setPixel(point(x, y), objOuEntierCouleur)

// Syntaxe JavaScript
réfObjImage.setPixel(x, y, objOuEntierCouleur);
réfObjImage.setPixel(point(x, y), objOuEntierCouleur);
```

Description

Méthode d'image. Définit la valeur de couleur du pixel à un point spécifié dans une image donnée.

Si vous définissez un grand nombre de pixels sur la couleur d'un autre pixel à l'aide de `getPixel()`, il est plus rapide de les définir en tant que nombres entiers.

Pour obtenir des performances maximales avec les objets couleur, utilisez un objet couleur indexée pour les images 8 bits (ou d'une qualité inférieure) et un objet couleur RVB pour des images 16 bits (ou d'une qualité supérieure).

Cette méthode renvoie FALSE si le pixel spécifié se trouve en dehors de l'image spécifiée.

Pour un exemple d'utilisation de cette méthode dans une animation, reportez-vous à l'animation Imaging du dossier Learning/Lingo, lui-même inclus dans le dossier de Director.

Paramètres

x Requis si vous spécifiez un pixel avec des coordonnées `x` et `y`. Entier qui spécifie la coordonnée `x` du pixel.

y1 Requis si vous spécifiez un pixel avec des coordonnées `x` et `y`. Entier qui spécifie la coordonnée `y` du pixel.

`point(x, y)` Requis si vous spécifiez un pixel en utilisant un point. Point qui spécifie le pixel.

objOuEntierCouleur Requis si vous définissez la couleur sur un objet couleur ou un nombre entier. Référence à un objet couleur qui spécifie la couleur du pixel ou nombre entier qui spécifie la valeur de couleur du pixel.

Exemple

L'instruction Lingo suivante dessine une ligne noire horizontale de 50 pixels, de gauche à droite, dans l'acteur 5 :

Voir aussi

`color()`, `draw()`, `fill()`, `getPixel()`, `image()`

setPlayList()

Utilisation

```
-- Syntaxe Lingo
réfObjPisteAudio.setPlayList(listeLinéaireDeListesDePropriétés)

// Syntaxe JavaScript
réfObjPisteAudio.setPlayList(listeLinéaireDeListesDePropriétés);
```

Description

Méthode de piste audio ; définit ou réinitialise la liste de lecture d'une piste audio.

Cette méthode est pratique pour placer plusieurs sons en file d'attente en une seule opération.

Pour un exemple d'utilisation de `setPlaylist()` dans une animation, reportez-vous à l'animation Sound Control du dossier Learning/Lingo, lui-même inclus dans le dossier de Director.

Paramètres

listeLinéaireDeListesDePropriétés Requis. Liste linéaire de listes de propriétés qui spécifie les paramètres d'une liste de lecture. Vous pouvez définir ces paramètres pour chaque son à placer en file d'attente :

Propriété	Description
`#member`	L'acteur à placer en file d'attente. Cette propriété doit être spécifiée. Toutes les autres sont facultatives.
`#startTime`	Position temporelle de départ de la lecture du son, en millisecondes. La valeur par défaut est le début du son. Pour plus d'informations, consultez `startTime`.
`#endTime`	Position temporelle de fin de la lecture du son, en millisecondes. La valeur par défaut est la fin du son. Pour plus d'informations, consultez `endTime`.
`#loopCount`	Nombre de répétitions d'une boucle défini avec `#loopStartTime` et `#loopEndTime`. La valeur par défaut est 1. Voir `loopCount`.
`#loopStartTime`	Position temporelle de départ de la boucle, en millisecondes. Pour plus d'informations, consultez `loopStartTime`.
`#loopEndTime`	Position temporelle de fin de la boucle, en millisecondes. Pour plus d'informations, consultez `loopEndTime`.
`#preloadTime`	Quantité de son à placer en mémoire tampon avant la lecture, en millisecondes. Pour plus d'informations, consultez `preloadTime`.

Exemple

Le gestionnaire suivant place en file d'attente et lit l'acteur son Intro, à partir du point 3 secondes, exécute cinq lectures en boucle successives, du point 8 secondes au point 8,9 secondes, et s'arrête au point 10 secondes ;

```
-- Syntaxe Lingo
on playMusic
  sound(2).queue([#member:member("intro"), #startTime:3000, \
  #endTime:10000, #loopCount:5, #loopStartTime:8000, #loopEndTime:8900])
  sound(2).play()
end playMusic

// Syntaxe JavaScript
function playMusic() {
  sound(2).queue(propList("member",member("Intro"),
  "startTime",3000, "endTime",10000, "loopCount",5, "loopStartTime",8000,
  "loopEndTime",8900));
  sound(2).play();
}
```

Voir aussi

endTime, getPlayList(), loopCount, loopEndTime, loopStartTime, Acteur, member, preLoadTime, queue(), Piste audio, startTime

setPosition()

Utilisation

```
-- Syntaxe Lingo
réfObjFileio.setPosition(entPosition)

// Syntaxe JavaScript
réfObjFileio.setPosition(entPosition);
```

Description

Méthode de Fileio ; définit la position d'un fichier.

Paramètres

entPosition Requis. Nombre entier qui spécifie la nouvelle position du fichier.

Voir aussi

Fileio

setPref()

Utilisation

```
-- Syntaxe Lingo
_player.setPref(chaîneNomDePréf, chaîneDePréf)

// Syntaxe JavaScript
_player.setPref(chaîneNomDePréf, chaîneDePréf);
```

Description

Méthode de lecteur ; écrit une chaîne donnée dans un fichier spécifié sur le disque local de l'ordinateur. Le fichier est un fichier texte standard.

Après l'exécution de la méthode `setPref()`, si l'animation est lue dans un navigateur, un dossier nommé Prefs est créé à l'intérieur du dossier Plug-In Support. La méthode `setPref()` ne peut écrire que dans ce dossier.

Si l'animation est lue dans une projection ou dans Director, un dossier est créé dans le même dossier que l'application. Le dossier est appelé *Prefs*.

N'utilisez pas cette méthode pour écrire sur un média en lecture seule. Selon la plate-forme et la version du système d'exploitation utilisé, vous pourriez rencontrer des erreurs ou autres problèmes.

Dans un navigateur, les données écrites par `setPref()` ne sont pas confidentielles. Toute animation avec un contenu Shockwave est en mesure de lire ces informations et de les télécharger vers un serveur. Les informations confidentielles ne doivent donc pas être stockées à l'aide de la commande `setPref()`.

Sous Windows, la méthode `setPref()` échoue si l'utilisateur dispose de droits d'accès restreints.

Pour un exemple d'utilisation de `setPref()` dans une animation, reportez-vous à l'animation Read and Write Text du dossier Learning/Lingo, lui-même inclus dans le dossier de Director.

Paramètres

`chaîneNomDePréf` Requis. Chaîne qui spécifie le fichier dans lequel les données seront enregistrées. Le paramètre *`chaîneNomDePréf`* doit être un nom de fichier valide. Pour vous assurer de sa validité sur toutes les plates-formes, n'utilisez pas plus de huit caractères alphanumériques dans ce nom de fichier.

`chaîneDePréf` Requis. Chaîne qui spécifie le texte à écrire dans le fichier *`chaîneNomDePréf`*.

Exemple

Le gestionnaire suivant enregistre le contenu de l'acteur champ Saisie dans un fichier nommé PréfsActuelles :

```
-- Syntaxe Lingo
on mouseUp me
  _player.setPref("PréfsActuelles", member("Saisie").text)
end

// Syntaxe JavaScript
function mouseUp() {
  _player.setPref("PréfsActuelles", member("Saisie").text);
}
```

Voir aussi

`getPref()`, `Lecteur`

setProp

Utilisation

```
setProp liste, propriété, nouvelleValeur
liste.propriétéDeListe = nouvelleValeur
liste[propriétéDeListe] = nouvelleValeur
```

Description

Commande ; dans une liste, remplace la valeur affectée à une propriété spécifiée par une nouvelle valeur. Si la liste ne contient pas la propriété spécifiée, `setProp` renvoie une erreur de script.

La commande `setProp` fonctionne uniquement avec les listes de propriétés. L'utilisation de `setProp` avec une liste linéaire produit une erreur de script.

Cette commande est similaire à la commande `setaProp`, sauf que `setProp` renvoie une erreur lorsque la propriété ne se trouve pas dans la liste.

Paramètres

`propriété` Requis. Symbole (Lingo uniquement) ou chaîne qui spécifie la propriété dont la valeur est remplacée par *`nouvelleValeur`*.

`nouvelleValeur` Requis. Nouvelle valeur de la propriété spécifiée par *`propriété`*.

Exemple

L'instruction suivante fait passer à 11 la valeur affectée à la propriété âge de la liste de propriétés x :

```
setProp x, #âge, 11
```

L'opérateur point permet de modifier la valeur de propriété d'une propriété figurant déjà dans une liste, tout comme ci-dessus :

```
x.âge = 11
```

Voir aussi

`setaProp`

setScriptList()

Utilisation

```
réfDimageObjet.setScriptList(listeDeScripts)
sprite(quelleImageObjet).setScriptList(listeDeScripts)
```

Description

Cette commande définit la liste `scriptList` de l'image-objet donnée. La liste `scriptList` indique les scripts attachés à l'image-objet, ainsi que les paramètres de chaque propriété de script. La définition de cette liste permet de modifier les paramètres attachés à une image-objet ou de modifier les propriétés de comportement.

La liste prend la forme suivante :

```
[ [ (quelActeurComportement), " [ #prop1: valeur, #prop2: valeur, . . . ] ",
[(quelActeurComportement), " [ #prop1: valeur, #prop2: valeur, . . . ] " ] ]
```

Cette commande ne peut pas être utilisée pendant la création du scénario. Utilisez `setScriptList()` pour les images-objets ajoutées pendant l'enregistrement du scénario après la session d'enregistrement du scénario.

Paramètres

`listeDeScripts` Requis. Spécifie la liste de scripts pour une image-objet donnée.

Voir aussi

`scriptList`, `value()`, `string()`

settingsPanel()

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.settingsPanel({entierIndexPanneau})

// Syntaxe JavaScript
réfObjImageObjet.settingsPanel({entierIndexPanneau});
```

Description

Commande d'image-objet Flash ; invoque la boîte de dialogue des paramètres de Flash, à l'index de panneau indiqué. Il s'agit de la même boîte de dialogue qui est accessible en cliquant du bouton droit (Windows) ou en cliquant avec la touche Ctrl enfoncée (Macintosh) sur une animation Flash lue dans un navigateur.

La boîte de dialogue des paramètres ne s'affiche pas si les dimensions du rectangle de l'image-objet Flash ne peuvent s'y adapter.

Si vous souhaitez émuler le lecteur Flash en appelant la boîte de dialogue des paramètres lorsque l'utilisateur clique du bouton droit (Windows) ou clique avec la touche Ctrl enfoncée (Macintosh), vous pouvez utiliser cette commande dans un gestionnaire `mouseDown` qui teste la propriété `rightMouseDown` ou `controlDown`.

Pour émuler Flash Player en activant la boîte de dialogue des paramètres dans une animation Director exécutée dans un navigateur, vous devez d'abord désactiver le menu contextuel Shockwave Player, auquel vous accédez avec un clic sur le bouton droit (Windows) ou en cliquant avec la touche Ctrl enfoncée (Macintosh) dans une animation avec un contenu Shockwave lue dans un navigateur. Pour plus d'informations sur la désactivation de ce menu, consultez les rubriques Utilisation de Director dans le panneau d'aide de Director.

Paramètres

`entierIndexPanneau` Facultatif. Spécifie quel panneau doit être activé à l'ouverture de la boîte de dialogue. Les valeurs correctes sont 0, 1, 2 ou 3. Toutes les quatre ouvrent la boîte de dialogue, la valeur 0 affichant l'onglet Contrôle de l'accès, la valeur 1 l'onglet Enregistrement local, la valeur 2 l'onglet Microphone et la valeur 3 l'onglet Caméra. L'index de panneau par défaut est 0.

Exemple

L'instruction suivante ouvre le panneau des paramètres de Flash, qui présente l'onglet Enregistrement local :

```
-- Syntaxe Lingo
sprite(3).settingsPanel(1)

// Syntaxe JavaScript
sprite(3).settingsPanel(1);
```

Voir aussi

on mouseDown (gestionnaire d'événement), rightMouseDown, controlDown

setPref()

Utilisation

```
-- Syntaxe Lingo
_player.setPref(chaîneNomDePréf, chaîneDePréf)

// Syntaxe JavaScript
_player.setPref(chaîneNomDePréf, chaîneDePréf);
```

Description

Méthode de lecteur ; écrit la chaîne spécifiée par *chaîneDePréf* dans le fichier spécifié par *chaîneNomDePréf* sur le disque local de l'ordinateur.

L'argument *chaîneNomDePréf* doit correspondre à un nom de fichier valide. Pour vous assurer de sa validité sur toutes les plates-formes, n'utilisez pas plus de huit caractères alphanumériques dans ce nom de fichier.

Après l'exécution de la méthode `setPref`, si l'animation est lue dans un navigateur, un dossier nommé Prefs est créé à l'intérieur du dossier Plug-In Support. La méthode `setPref()` ne peut écrire que dans ce dossier.

Si l'animation est lue dans une projection ou dans Director, un dossier est créé dans le même dossier que l'application. Le dossier est appelé *Prefs*.

N'utilisez pas cette méthode pour écrire sur un média en lecture seule. Selon la plate-forme et la version du système d'exploitation utilisé, vous pourriez rencontrer des erreurs ou autres problèmes.

Cette méthode n'exécute pas de manipulation complexe sur les données de la chaîne ou son formatage. Toute opération de formatage ou autre manipulation doit être effectuée parallèlement à `getPref()` ; les données peuvent ensuite être traitées en mémoire et inscrites dans l'ancien fichier par le biais de `setPref()`.

Dans un navigateur, les données écrites par `setPref()` ne sont pas confidentielles. Toute animation avec un contenu Shockwave est en mesure de lire ces informations et de les télécharger vers un serveur. Les informations confidentielles ne doivent donc pas être stockées à l'aide de la commande `setPref()`.

Sous Windows, la commande `setPref()` échoue si l'utilisateur dispose de droits d'accès restreints.

Pour un exemple d'utilisation de `setPref()` dans une animation, reportez-vous à l'animation Read and Write Text du dossier Learning/Lingo, lui-même inclus dans le dossier de Director.

Paramètres

chaîneNomDePréf Requis. Chaîne qui spécifie le nom du fichier dans lequel la chaîne *chaîneDePréf* est écrite. Le fichier est un fichier texte standard.

chaîneDePréf Requis. Chaîne à écrire dans le fichier spécifié par *chaîneNomDePréf*.

Exemple

Le gestionnaire suivant enregistre le contenu de l'acteur champ Saisie dans un fichier nommé PréfsActuelles :

Voir aussi

getPref(), Lecteur

setTrackEnabled()

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.setTrackEnabled(quellePiste, trueOrFalse)

// Syntaxe JavaScript
réfObjImageObjet.setTrackEnabled(quellePiste, trueOrFalse);
```

Description

Commande ; définit si la lecture de la piste spécifiée d'une vidéo numérique est activée.

- Si `setTrackEnabled` a la valeur `TRUE`, la piste indiquée est activée et lue.
- Si `setTrackEnabled` a la valeur `FALSE`, la piste indiquée est désactivée et n'est pas lue. Pour les vidéos numériques, cela signifie qu'elles ne seront plus mises à jour à l'écran.

Pour tester si une piste est déjà activée, testez la propriété d'image-objet `trackEnabled`.

Paramètres

quellePiste Requis. Spécifie la piste à tester.

trueOuFalse Requis. Spécifie si la piste dans la vidéo numérique est activée (`TRUE`) ou non (`FALSE`).

Exemple

L'instruction suivante active la piste 3 de la vidéo numérique affectée à la piste d'image-objet 8 :

```
-- Syntaxe Lingo
sprite(8).setTrackEnabled(3, TRUE)

// Syntaxe JavaScript
sprite(8).setTrackEnabled(3, 1);
```

Voir aussi

trackEnabled

setVariable()

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.setVariable(nomDeVariable, nouvelleValeur)

// Syntaxe JavaScript
réfObjImageObjet.setVariable(nomDeVariable, nouvelleValeur);
```

Description

Fonction ; définit la valeur de la variable spécifiée dans l'image-objet donnée. Les variables Flash ont été introduites dans la version 4 de Flash.

Paramètres

nomDeVariable Requis. Spécifie le nom de la variable.

nouvelleValeur Requis. Saisissez la nouvelle valeur de la variable.

Exemple

Les instructions suivantes définissent la valeur de la variable URLcourante dans l'acteur Flash, au niveau de l'image-objet 3. La nouvelle valeur de la variable URLcourante sera « http://www.macromedia.com/fr/software/flash/ ».

```
-- Syntaxe Lingo
sprite(3).setVariable("URLCourante", \
  "http://www.macromedia.com/fr/software/flash/")

// Syntaxe JavaScript
sprite(3).setVariable("URLCourante",
  "http://www.macromedia.com/fr/software/flash/");
```

Voir aussi

hitTest(), getVariable()

shader()

Utilisation

```
member(quelActeur).shader(quelMatériau)
member(quelActeur).shader[index]
member(quelActeur).model(quelModèle).shader
member(quelActeur).modelResource(quelleRessDeMod).\
  face[index].shader
```

Description

Propriété 3D d'élément, de modèle et de face ; objet utilisé pour définir l'apparence de la surface du modèle. Le matériau est la « peau » qui entoure la ressource de modèle utilisée par le modèle.

Le matériau même n'est pas une image. Le composant visible d'un matériau est créé avec un maximum de huit couches de `textures`. Ces huit couches de texture ont été créées à partir d'acteurs bitmap ou d'objets image dans Director ou importées avec des modèles de programmes de modélisation 3D. Pour plus d'informations, consultez `texture`.

Tout modèle dispose d'une liste linéaire de matériaux appelée `shaderList`. Le nombre d'entrées de la liste est égal au nombre de mailles de la ressource utilisée par le modèle. Un seul matériau peut être appliqué à chaque maille.

L'acteur 3D a un matériau par défaut nommé `DefaultShader`, qui ne peut pas être supprimé. Ce matériau est utilisé lorsque aucun matériau n'est affecté à un modèle et lorsqu'un matériau utilisé par un modèle est supprimé.

La syntaxe `member(`*`quelActeur`*`).model(`*`quelModèle`*`).shader` donne accès au premier matériau de la liste des matériaux du modèle et est équivalent à `member(`*`quelActeur`*`).model(`*`quelModèle`*`).shaderList[1]`.

Les matériaux sont créés et supprimés avec les commandes `newShader()` et `deleteShader()`.

Les matériaux sont enregistrés dans la palette des matériaux de l'acteur 3D. Ils peuvent être référencés par nom (*`quelMatériau`*) ou par index de palette (*`indexDeMatériau`*). Un matériau peut être utilisé par n'importe quel nombre de modèles. Les modifications apportées à un matériau apparaissent dans tous les modèles qui l'utilisent.

Il existe quatre types de matériaux :

Les matériaux `#standard` présentent leurs textures de façon réaliste.

Les matériaux `#painter`, `#engraver` et `#newsprint` stylisent leurs textures pour qu'elles aient l'apparence d'une peinture, d'une gravure ou d'un journal. Ils ont des propriétés spéciales en plus des propriétés de matériau `#standard`.

Les matériaux utilisés par les faces individuelles des primitives `#mesh` peuvent être définis à l'aide de la syntaxe `member(`*`quelActeur`*`).modelResource(`*`quelleRessDeMod`*`).face[`*`index`*`].shader`. Pour modifier cette propriété, il est nécessaire d'appeler la commande `build()`.

Exemple

L'instruction suivante affecte le matériau surfaceDuMur à la propriété de matériau du modèle Mur.

```
member("Pièce").model("Mur").shader = \
  member("Pièce").shader("surfaceDuMur")
```

Voir aussi

`shaderList`, `newShader`, `deleteShader`, `face[ ]`, `texture()`

showLocals()

Utilisation

```
-- Syntaxe Lingo
showLocals()
```

Description

Fonction du niveau supérieur (Lingo uniquement) ; affiche toutes les variables locales dans la fenêtre Messages. Elle n'est utile que dans les gestionnaires ou les scripts parents contenant des variables locales à afficher. Toutes les variables utilisées dans la fenêtre Messages sont automatiquement globales.

Les variables locales des gestionnaires sont supprimées après l'exécution de ces derniers. L'insertion de l'instruction `showLocals()` dans un gestionnaire a pour effet d'afficher les variables locales de ce gestionnaire dans la fenêtre Messages.

Cette commande est pratique pour déboguer les scripts.

Paramètres

Aucun.

Voir aussi

clearGlobals(), global, showGlobals()

showProps()

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.showProps()

// Syntaxe JavaScript
réfObjActeurOuImageObjet.showProps();
```

Description

Commande ; affiche une liste des paramètres de propriétés actuels d'une animation Flash, d'un acteur vectoriel ou d'un son en cours de lecture dans la fenêtre Messages. Cette commande n'est utile que pour la programmation et ne fonctionne pas dans les projections ou les animations avec un contenu Shockwave.

Paramètres

Aucun.

Exemple

Le gestionnaire suivant accepte le nom d'une distribution comme paramètre, recherche les acteurs animation Flash de cette distribution et affiche le nom, le numéro et les propriétés de l'acteur dans la fenêtre Messages :

```
-- Syntaxe Lingo
on AfficherPropDistribution(quelleDistribution)
  repeat with i = 1 to castLib(quelleDistribution).member.count
    typeDeDistribution = member(i, quelleDistribution).type
    if (typeDeDistribution = #flash) OR (typeDeDistribution = #vectorShape)
  then
      put typeDeDistribution&&"acteur" && i & " :" && member(i,
  quelleDistribution).name
      put RETURN
      member(i ,quelleDistribution).showProps()
    end if
  end repeat
end
```

```
// Syntaxe JavaScript
function AfficherPropDistribution(quelleDistribution) {
  i = 1;
  while( i < (castLib(quelleDistribution).member.count) +1 ) {
    typeDeDistribution = member(i, quelleDistribution).type;
    if ((typeDeDistribution = "flash") || (typeDeDistribution =
"vectorShape")) {
      trace (typeDeDistribution + " acteur " + i + ": " + member(i,
quelleDistribution).name) + \n;
      member(i ,quelleDistribution).showProps();
      i++;
    }
  }
}
```

Voir aussi

queue(), setPlayList()

showGlobals()

Utilisation

```
-- Syntaxe Lingo
_global.showGlobals()

// Syntaxe JavaScript
_global.showGlobals();
```

Paramètres

Aucun.

Description

Méthode globale ; affiche toutes les variables globales dans la fenêtre Messages.

Cette méthode est pratique pour déboguer les scripts.

Exemple

L'instruction suivante affiche toutes les variables globales dans la fenêtre Messages :

```
-- Syntaxe Lingo
on mouseDown
  _global.showGlobals()
end

// Syntaxe JavaScript
function mouseDown() {
  _global.showGlobals();
}
```

Voir aussi

Global

shutDown()

Utilisation

```
-- Syntaxe Lingo
_system.shutDown()

// Syntaxe JavaScript
_system.shutDown();
```

Description

Méthode du système ; ferme toutes les applications ouvertes et éteint l'ordinateur.

Paramètres

Aucun.

Exemple

L'instruction suivante vérifie si l'utilisateur a utilisé le raccourci clavier Ctrl-S (Windows) ou Cmd-S (Macintosh) et, le cas échéant, éteint l'ordinateur :

Voir aussi

Système

sin()

Utilisation

`sin(`*`angle`*`)`

Description

Fonction mathématique (Lingo uniquement) ; calcule le sinus de l'angle spécifié. Celui-ci doit être exprimé en radians sous forme de nombre à virgule flottante.

Dans la syntaxe JavaScript, utilisez la fonction `sin()` de l'objet Math.

Paramètres

`angle` Requis. Spécifie l'angle.

Exemple

L'instruction suivante calcule le sinus de `pi`/2 :

```
put sin (PI/2.0)
-- 1
```

Voir aussi

PI

sort

Utilisation

```
liste.sort()
sort liste
```

Description

Commande ; trie les éléments d'une liste par ordre alphabétique.

- S'il s'agit d'une liste linéaire, elle est triée par valeurs.
- S'il s'agit d'une liste de propriétés, elle est triée alphabétiquement par propriété.

Une fois que la liste est triée, elle le reste, même si vous ajoutez des variables avec la commande `add`.

Paramètres

Aucun.

Exemple

L'instruction suivante trie la liste Valeurs qui consiste en [`#a: 1, #d: 2, #c: 3`], par ordre alphabétique. Le résultat apparaît sous l'instruction.

```
put Valeurs
-- [#a: 1, #d: 2, #c: 3]
Valeurs.sort()
put Valeurs
-- [#a: 1, #c: 3, #d: 2]
```

sound()

Utilisation

```
-- Syntaxe Lingo
sound(entPisteAudio)

// Syntaxe JavaScript
sound(entPisteAudio);
```

Description

Fonction du niveau supérieur ; renvoie une référence à une piste audio spécifiée.

La fonctionnalité de cette méthode est identique à celle de la méthode `channel()` de l'objet Son.

Paramètres

`entPisteAudio` Requis. Nombre entier qui spécifie la piste audio à référencer.

Exemple

L'exemple suivant affecte la piste audio 1 à une variable `music` et lit un son.

```
-- Syntaxe Lingo
music = sound(1)
music.play(member("valse1"))

// Syntaxe JavaScript
var music = sound(1);
music.play(member("valse1"));
```

Voir aussi

`channel() (audio)`, `Piste audio`

sprite()

Utilisation

```
-- Syntaxe Lingo
sprite(nomOuNum)

// Syntaxe JavaScript
sprite(nomOuNum);
```

Description

Fonction du niveau supérieur ; renvoie une référence à une image-objet donnée dans le scénario.

Si le scénario ne contient pas cette image-objet, la fonction renvoie une chaîne vide.

Paramètres

nomOuNum Requis. Chaîne ou nombre entier qui spécifie le nom ou la position d'index de l'image-objet.

Exemple

Cette instruction affecte la variable `cetteImageObjet` à l'image-objet Grotte :

```
-- Syntaxe Lingo
cetteImageObjet = sprite("Grotte")

// Syntaxe JavaScript
var cetteImageObjet = sprite("Grotte");
```

Voir aussi

`Piste d'image-objet`

spriteSpaceToWorldSpace

Utilisation

```
sprite(quelleImageObjet).camera.spriteSpaceToWorldSpace(emplacement)
sprite(quelleImageObjet).camera(index).spriteSpaceToWorldSpace(emplacement)
```

Description

Commande 3D ; renvoie une position d'univers située sur le plan de projection de la caméra spécifiée, qui correspond à un emplacement dans l'image-objet référencée.

Le plan de projection est défini par les axes des x et y de la caméra, et sa distance devant la caméra est telle qu'un pixel représente une unité de mesure d'univers. C'est ce plan de projection qui est utilisé pour l'affichage de l'image-objet sur la scène.

La forme `camera.spriteSpaceToWorldSpace()` de cette commande est un raccourci de `camera(1).spriteSpaceToWorldSpace()`.

Toutes les caméras utilisées par l'image-objet référencée répondront à la commande `spriteSpaceToWorldSpace` comme si leur rectangle d'affichage était de la même taille que l'image-objet.

Paramètres

`emplacement` Requis. Spécifie l'emplacement dans l'image-objet référencée. Cet emplacement doit être un point relatif au coin supérieur gauche de l'image-objet.

Exemple

L'instruction suivante indique que le point (50, 50) de l'image-objet 5 équivaut à `vector(1993.6699, 52.0773, 2263.7446)` sur le plan de projection de la caméra de l'image-objet 5.

```
put sprite(5).camera.spriteSpaceToWorldSpace(point(50, 50))
-- vector( -1993.6699, 52.0773, 2263.7446 )
```

Voir aussi

`worldSpaceToSpriteSpace`, `rect (caméra)`, `camera`

sqrt()

Utilisation

```
sqrt(nombre)
the sqrt of nombre
```

Description

Fonction mathématique (Lingo uniquement) ; renvoie la racine carrée d'un nombre spécifié.

La valeur doit être un nombre décimal supérieur à 0. Les valeurs négatives renvoient la valeur 0.

Dans la syntaxe JavaScript, utilisez la fonction `sqrt()` de l'objet Math.

Paramètres

`nombre` Requis. Spécifie la valeur numérique. Il peut s'agir soit d'un nombre à virgule flottante soit d'un nombre entier arrondi au nombre entier le plus proche.

Exemple

L'instruction suivante affiche la racine carrée de 3.0 dans la fenêtre Messages :

```
put sqrt(3.0)
-- 1.7321
```

L'instruction suivante affiche la racine carrée de 3 dans la fenêtre Messages :

```
put sqrt(3)
-- 2
```

Voir aussi

floatPrecision

stageBottom

Utilisation

```
the stageBottom
```

Description

Fonction ; utilisée conjointement à `stageLeft`, `stageRight` et `stageTop`, indique la position de la scène sur le bureau. Elle renvoie la coordonnée verticale du bord inférieur de la scène, par rapport au coin supérieur gauche de l'écran principal. La hauteur de la scène en pixels est déterminée par la formule `the stageBottom - the stageTop`.

Lorsque l'animation est lue en tant qu'applet, la propriété `stageBottom` correspond à la hauteur de cet applet en pixels.

Cette fonction peut être testée, mais pas définie.

Paramètres

Aucun.

Exemple

Les instructions suivantes placent l'image-objet 3 à une distance de 50 pixels à partir du bord inférieur de la scène :

```
hauteurDeLaScène = the stageBottom - the stageTop
sprite(3).locV = hauteurDeLaScène - 50
```

Les coordonnées des images-objets sont exprimées par rapport au coin supérieur gauche de la scène. Pour plus d'informations, consultez les rubriques Utilisation de Director dans le panneau d'aide de Director.

Voir aussi

stageLeft, stageRight, stageTop, locH, locV

stageLeft

Utilisation

```
the stageLeft
```

Description

Fonction ; utilisée conjointement à `stageRight`, `stageTop` et `stageBottom`, indique la position de la scène sur le bureau. Elle renvoie la coordonnée horizontale gauche de la scène, par rapport au coin supérieur gauche de l'écran principal. Lorsque le bord de la scène coïncide avec le côté gauche de l'écran principal, cette coordonnée est 0.

Lorsque l'animation est lue sous forme d'applet, la propriété `stageLeft` est 0, qui correspond à l'emplacement du côté gauche de l'applet.

Cette propriété peut être testée, mais pas définie.

Les coordonnées des images-objets sont exprimées par rapport au coin supérieur gauche de la scène.

Paramètres

Aucun.

Exemple

L'instruction suivante vérifie si le bord gauche de la scène dépasse le bord gauche de l'écran et, le cas échéant, appelle le gestionnaire *`aGaucheDuMoniteur`* :

```
if the stageLeft < 0 then aGaucheDuMoniteur
```

Voir aussi

`stageBottom`, `stageRight`, `stageTop`, `locH`, `locV`

stageRight

Utilisation

```
the stageRight
```

Description

Fonction ; utilisée conjointement à `stageLeft`, `stageTop` et `stageBottom`, indique la position de la scène sur le bureau. Elle renvoie la coordonnée horizontale droite de la scène, par rapport au coin supérieur gauche de l'écran principal. La largeur de la scène en pixels est déterminée par la formule `the stageRight - the stageLeft`.

Lorsque l'animation est lue sous forme d'applet, la propriété `stageRight` est la largeur de l'applet, en pixels.

Cette fonction peut être testée, mais pas définie.

Les coordonnées des images-objets sont exprimées par rapport au coin supérieur gauche de la scène.

Paramètres

Aucun.

Exemple

Les deux instructions suivantes placent l'image-objet 3 à une distance de 50 pixels du bord droit de la scène :

```
largeurDeLaScène = the stageRight - the stageLeft
  sprite(3).locH = largeurDeLaScène - 50
```

Voir aussi

`stageLeft`, `stageBottom`, `stageTop`, `locH`, `locV`

stageToFlash()

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.stageToFlash(pointDeLaScèneDeDirector)

// Syntaxe JavaScript
réfObjImageObjet.stageToFlash(pointDeLaScèneDeDirector);
```

Description

Fonction ; renvoie la coordonnée d'une animation Flash correspondant à une coordonnée spécifiée sur la scène de Director. Cette fonction accepte la coordonnée de la scène Director et renvoie la coordonnée de l'animation Flash sous la forme de valeurs de point Director : par exemple, point (300,300).

Les coordonnées de l'animation Flash sont mesurées en pixels d'animation Flash, déterminés par la taille d'origine de l'animation lorsqu'elle a été créée dans Flash. Le point (0,0) d'une animation Flash est toujours placé dans son coin supérieur gauche. La propriété `originPoint` de l'acteur n'intervient pas dans le calcul des coordonnées d'une animation, mais seulement pour la rotation et la mise à l'échelle.

La fonction `stageToFlash()` et la fonction `flashToStage()` correspondante sont pratiques pour déterminer la coordonnée d'une animation Flash se trouvant à une coordonnée spécifique de la scène Director. Pour Flash et Director, le point (0,0) est le coin supérieur gauche de la scène Flash ou Director. Ces coordonnées peuvent ne pas coïncider sur la scène Director si une image-objet Flash est étirée, mise à l'échelle ou a pivoté.

Paramètres

pointDeLaScèneDeDirector Requis. Spécifie le point sur la scène Director.

Exemple

Le gestionnaire suivant vérifie si la souris (dont l'emplacement est suivi dans les coordonnées de la scène Director) est placée sur une coordonnée spécifique (130,10) dans une image-objet animation Flash placée dans la piste 5. Si la souris est placée sur cette coordonnée, le script interrompt l'animation Flash.

```
-- Syntaxe Lingo
on checkFlashRollover
  if (sprite(5).stageToFlash(point(_mouse.mouseH, _mouse.mouseV)) =
  point(130,10) then
    sprite(5).stop()
  end if
end

// Syntaxe JavaScript
function checkFlashRollover() {
  var stf = sprite(5).stageToFlash(point(_mouse.mouseH,_mouse.mouseV));
  if (stf = point(130,10)) {
    sprite(5).stop();
  }
}
```

Voir aussi

`flashToStage()`

stageTop

Utilisation

```
the stageTop
```

Description

Fonction ; utilisée conjointement à `stageBottom`, `stageLeft` et `stageRight`, indique la position de la scène sur le bureau. Elle renvoie la coordonnée verticale supérieure de la scène par rapport au coin supérieur gauche de l'écran principal. Si la scène est dans le coin supérieur gauche de l'écran principal, cette coordonnée est 0.

Lorsque l'animation est lue en tant qu'applet, la valeur de la propriété `stageTop` est toujours zéro, l'emplacement du bord supérieur de l'applet.

Cette fonction peut être testée, mais pas définie.

Les coordonnées des images-objets sont exprimées par rapport au coin supérieur gauche de la scène.

Paramètres

Aucun.

Exemple

L'instruction suivante vérifie si le bord supérieur de la scène dépasse le bord supérieur de l'écran et, le cas échéant, appelle le gestionnaire *`débordementEnHaut`* :

```
if the stageTop < 0 then débordementEnHaut
```

Voir aussi

`stageLeft`, `stageRight`, `stageBottom`, `locH`, `locV`

status()

Utilisation

```
-- Syntaxe Lingo
réfObjFileio.status()

// Syntaxe JavaScript
réfObjFileio.status();
```

Description

Méthode de Fileio ; renvoie le code d'erreur de la dernière méthode appelée.

Paramètres

Aucun.

Voir aussi

`Fileio`

stop() (DVD)

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.stop()

// Syntaxe JavaScript
réfObjDvd.stop();
```

Description

Méthode de DVD ; arrête la lecture.

Cette méthode renvoie `TRUE` (1) en cas de réussite.

Paramètres

Aucun.

Voir aussi

DVD

stop() (piste audio)

Utilisation

```
-- Syntaxe Lingo
réfObjPisteAudio.stop()

// Syntaxe JavaScript
réfObjPisteAudio.stop();
```

Description

Méthode de piste audio ; arrête le son qui est en cours de lecture sur une piste audio.

L'émission d'une méthode `play()` commence la lecture du premier son encore présent dans la file d'attente de la piste audio donnée.

Pour un exemple d'utilisation de `stop()` dans une animation, reportez-vous à l'animation Sound Control du dossier Learning/Lingo, lui-même inclus dans le dossier de Director.

Paramètres

Aucun.

Exemple

L'instruction suivante arrête la lecture de l'acteur son en cours de lecture dans la piste audio 1 :

```
-- Syntaxe Lingo
sound(1).stop()

// Syntaxe JavaScript
sound(1).stop();
```

Voir aussi

getPlayList(), pause() (piste audio), play() (piste audio), playNext() (piste audio), rewind() (piste audio), Piste audio

stop() (Flash)

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.stop()

// Syntaxe JavaScript
réfObjImageObjet.stop();
```

Description

Commande Flash ; interrompt une image-objet animation Flash lue dans l'image courante.

Paramètres

Aucun.

Exemple

Le script d'image suivant interrompt les images de l'animation Flash lues dans les pistes 5 à 10 :

```
-- Syntaxe Lingo
on enterFrame
  repeat with i = 5 to 10
    sprite(i).stop()
  end repeat
end

// Syntaxe JavaScript
function enterFrame() {
  var i = 5;
  while (i < 11) {
    sprite(i).stop();
    i++;
  }
}
```

Voir aussi

hold()

stop() (RealMedia, SWA, Windows Media)

Utilisation

```
-- Syntaxe Lingo
réfObjWindowsMedia.stop()
réfObjRealMedia.stop()

// Syntaxe JavaScript
réfObjWindowsMedia.stop();
réfObjRealMedia.stop();
```

Description

Méthode d'acteur ou d'image-objet Windows Media ou RealMedia. Arrête la lecture d'un acteur ou d'une image-objet Windows Media ou RealMedia.

Paramètres

Aucun.

Exemple

Les exemples suivants arrêtent la lecture de l'image-objet 2 et de l'acteur Real :

```
-- Syntaxe Lingo
sprite(2).stop()
member("Real").stop()

// Syntaxe JavaScript
sprite(2).stop();
member("Real").stop();
```

Voir aussi

RealMedia, Windows Media

stopEvent()

Utilisation

```
-- Syntaxe Lingo
_movie.stopEvent()

// Syntaxe JavaScript
_movie.stopEvent();
```

Description

Méthode d'animation ; empêche les scripts de passer un message d'événement aux emplacements suivants dans la hiérarchie de messages.

Cette méthode s'applique également aux scripts d'images-objets.

Utilisez la méthode `stopEvent()` pour arrêter le message dans un gestionnaire d'événement principal ou un script d'image-objet, et rendre ainsi le message non disponible pour les scripts d'images-objets suivants.

Par défaut, les messages sont d'abord mis à la disposition d'un gestionnaire d'événement principal (s'il existe), puis à celle de tous les scripts associés à une image-objet impliquée dans l'événement. Si plusieurs scripts sont associés à l'image-objet, le message est mis à la disposition de tous les scripts de l'image-objet. Si aucun script d'image-objet ne répond au message, celui-ci passe à un script d'acteur, puis à un script d'image et enfin à un script d'animation.

La méthode `stopEvent()` ne s'applique qu'à l'événement courant en cours de traitement. Elle n'affecte pas les événements futurs. La commande `stopEvent()` ne s'applique qu'aux gestionnaires d'événements principaux, aux gestionnaires appelés par des gestionnaires d'événements principaux ou à plusieurs scripts d'images-objets. Elle n'a aucun effet ailleurs.

Paramètres

Aucun.

Exemple

L'instruction suivante montre l'événement `mouseUp` interrompu dans un comportement lorsque la variable globale `totalGénéral` est égale à 500 :

```
-- Syntaxe Lingo
global totalGénéral

on mouseUp me
  if (totalGénéral = 500) then
    _movie.stopEvent()
  end if
end

// Syntaxe JavaScript
_global.totalGénéral;

function mouseUp() {
  if (_global.totalGénéral == 500) {
    _movie.stopEvent();
  }
}
```

Les autres comportements ou les scripts ultérieurs sur l'image-objet ne reçoivent pas l'événement s'il est interrompu de cette manière.

Voir aussi

`Animation`

stream()

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.stream(nombreDoctets)

// Syntaxe JavaScript
réfObjActeur.stream(nombreDoctets);
```

Description

Commande ; permet de faire passer manuellement en mémoire une partie d'un acteur animation Flash spécifié.

La commande `stream` renvoie le nombre d'octets réellement chargés. En fonction d'un ensemble de conditions variées (telles que la vitesse du réseau ou la disponibilité des données requises), le nombre d'octets réellement chargés peut être inférieur au nombre d'octets requis.

Vous pouvez toujours utiliser la commande `stream` pour un acteur, quelle que soit sa propriété `streamMode`.

Paramètres

`nombreDoctets` Facultatif. Nombre entier qui spécifie combien d'octets à lire en flux continu. Si vous omettez le paramètre *`nombreDoctets`*, Director essaie de lire le nombre d'octets défini par la propriété `bufferSize` de l'acteur.

Exemple

Le script d'image suivant vérifie si un acteur animation Flash liée a été complètement chargé en mémoire en contrôlant sa propriété `percentStreamed`. Si l'acteur n'est pas chargé en mémoire à cent pour cent, le script tente de charger 32 000 octets de l'animation en mémoire.

Le script enregistre également le nombre réel d'octets chargés dans une variable intitulée `octetsReçus`. Si le nombre d'octets chargés ne correspond pas au nombre d'octets requis, le script met à jour un acteur texte pour indiquer le nombre d'octets réellement reçus. Le script contraint la tête de lecture à passer en boucle dans l'image courante jusqu'à ce que l'acteur soit complètement chargé en mémoire.

```
-- Syntaxe Lingo
on exitFrame
  if member(10).percentStreamed < 100 then
    octetsReçus = member(10).stream(32000)
    if octetsReçus < 32000 then
      member("Ligne de message").text = "Reçus :" && octetsReçus \
  && "sur les 32 000 demandés."
      _movie.updateStage()
    else
      member("Ligne de message").text = "Les 32 000 octets ont été reçus."
    end if
    _movie.go(_movie.frame)
  end if
end

// Syntaxe JavaScript
function exitFrame() {
  var pctStm = member(10).percentStreamed;
  if (pctStm < 100) {
    var octetsReçus = member(10).stream(32000);
    if (octetsReçus < 32000) {
      member("Ligne de message").text = "Reçus :" + octetsReçus + " sur les
  32 000 demandés.";
      _movie.updateStage();
    } else {
      member("Ligne de message").text = "Les 32 000 octets ont été reçus.";
    }
    _movie.go(_movie.frame);
  }
}
```

string()

Utilisation

`string(`*`expression`*`)`

Description

Fonction ; convertit en chaîne un nombre entier, un nombre à virgule flottante, une référence d'objet, une liste, un symbole ou toute autre expression n'existant pas sous la forme d'une chaîne.

Paramètres

`expression` Requis. Expression à convertir en chaîne.

Exemple

L'instruction suivante additionne 2.0 + 2.5 et insère le résultat dans l'acteur champ Total :

```
member("Total").text = string(2.0 + 2.5)
```

L'instruction suivante convertit le symbole `#rouge` en une chaîne et l'insère dans l'acteur champ Couleur :

```
member("Couleur").text = string(#rouge)
```

Voir aussi

`value()`, `stringP()`, `float()`, `integer()`, `symbol()`

stringP()

Utilisation

`stringP(`*`expression`*`)`

Description

Fonction ; détermine si une expression est une chaîne (`TRUE`) ou non (`FALSE`).

Le *P* de `stringP` signifie *prédicat*.

Paramètres

`expression` Requis. Expression à tester.

Exemple

L'instruction suivante vérifie si *3* est une chaîne :

```
put stringP("3")
```

Le résultat est 1, équivalent numérique de `TRUE`.

L'instruction suivante vérifie si le nombre à virgule flottante 3,0 est une chaîne :

```
put stringP(3.0)
```

Puisque 3,0 est un nombre à virgule flottante et non une chaîne, le résultat est 0, équivalent numérique de `FALSE`.

Voir aussi

`floatP()`, `ilk()`, `integerP()`, `objectP()`, `symbolP()`

subPictureType()

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.subPictureType(entFlux)

// Syntaxe JavaScript
réfObjDvd.subPictureType(entFlux);
```

Description

Méthode de DVD ; spécifie le type d'un flux de sous-images donné.

Cette méthode peut renvoyer les valeurs suivantes :

Symbole	Description
#unknown	Le type de sous-image est inconnu.
#Language	La sous-image inclut un contenu lié à la langue, par exemple, des sous-titres ou d'autres textes.
#Other	La sous-image inclut un contenu non lié à la langue, par exemple, une balle qui rebondit entre les titres d'un karaoké.

Paramètres

entFlux Requis. Nombre entier qui spécifie le flux à tester.

Exemple

Cette instruction renvoie le type de sous-image du flux 2 :

```
-- Syntaxe Lingo
sprite(12).member.subPictureType(2)

// Syntaxe JavaScript
sprite(12).member.subPictureType(2);
```

Voir aussi

DVD

substituteFont

Utilisation

```
TextMemberRef.substituteFont(policeDorigine, nouvellePolice)
substituteFont(réfDacteurTexte, policeDorigine, nouvellePolice)
```

Description

Commande d'acteur texte ; remplace toutes les instances d'une police par une autre police dans un acteur texte.

Paramètres

policeDorigine Requis. Police à remplacer.

nouvellePolice Requis. La nouvelle police remplace la police spécifiée par *policeDorigine*.

Exemple

Le script suivant vérifie si la police Bonneville est disponible dans un acteur texte et, dans la négative, la remplace par Arial :

```
-- Syntaxe Lingo
property spriteNum

on beginSprite me
  acteurCourant = sprite(spriteNum).member
  if acteurCourant.missingFonts contains "Bonneville" then
    acteurCourant.substituteFont("Bonneville", "Arial")
  end if
end
```

```
// Syntaxe JavaScript
function beginSprite() {
  acteurCourant = sprite(spriteNum).member;
  if (acteurCourant.missingFonts contains "Bonneville") { //vérifier la
  syntaxe
    acteurCourant.substituteFont("Bonneville", "Arial");
  }
}
```

Voir aussi

missingFonts

swing()

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.swing(pan, inclin, champDeVue, vitesseDeMouv)

// Syntaxe JavaScript
réfObjImageObjet.swing(pan, inclin, champDeVue, vitesseDeMouv);
```

Description

Fonction d'image-objet QuickTime VR ; déplace une image-objet QuickTime 3 contenant un panoramique autour de nouveaux réglages de vue. Cette fonction produit un effet de travelling régulier.

réfObjImageObjet désigne le numéro de l'image-objet contenant l'acteur QuickTime VR.

Cette fonction renvoie immédiatement une valeur, mais l'image-objet continue de changer de vue jusqu'à ce que la vue finale soit atteinte. La durée requise pour arriver aux paramètres finaux varie en fonction du type d'ordinateur, de la taille du rectangle de l'image-objet, du codage des couleurs et d'autres paramètres affectant la performance.

Pour vérifier si le mouvement est terminé, vérifiez si la propriété `pan` de l'image-objet est arrivée à la valeur finale.

Paramètres

pan Requis. Spécifie la nouvelle position panoramique, en degrés.

inclin Requis. Spécifie la nouvelle inclinaison, en degrés.

champDeVue Requis. Spécifie le nouveau champ de vue, en degrés.

vitesseDeMouv Requis. Spécifie la vitesse à laquelle le mouvement doit s'effectuer. Les valeurs correctes sont comprises entre 1 (lent) et 10 (rapide).

Exemple

Ceci permet d'ajuster graduellement l'affichage de l'image-objet QuickTime VR dans une position panoramique de 300°, une inclinaison de -15° et un champ de vue de 40°.

```
-- Syntaxe Lingo
sprite(1).swing(300, -15, 40, 1)

// Syntaxe JavaScript
sprite(1).swing(300, -15, 40, 1);
```

Voir aussi

pan (propriété QTVR)

symbol()

Utilisation

```
-- Syntaxe Lingo
symbol(valeurDeChaîne)

// Syntaxe JavaScript
symbol(valeurDeChaîne);
```

Description

Fonction du niveau supérieur ; prend une chaîne et renvoie un symbole.

Paramètres

valeurDeChaîne Requis. Chaîne à convertir en symbole.

Exemple

L'instruction suivante affiche le symbole #bonjour :

```
-- Syntaxe Lingo
put(symbol("bonjour"))

// Syntaxe JavaScript
put(symbol("bonjour"));
```

L'instruction suivante affiche le symbole #auRevoir :

```
-- Syntaxe Lingo
x = "auRevoir"
put(symbol(x))

// Syntaxe JavaScript
var x = "auRevoir";
put(symbol(x));
```

Voir aussi

value(), string()

symbolP()

Utilisation

```
expression.symbolP
symbolP(expression)
```

Description

Fonction ; détermine si une expression spécifiée est un symbole (`TRUE`) ou non (`FALSE`).

Le *P* de `symbolP` signifie prédicat.

Paramètres

`expression` Requis. Spécifie l'expression à tester.

Exemple

L'instruction suivante vérifie si la variable `maVariable` est un symbole :

```
put maVariable.symbolP
```

Voir aussi

`ilk()`

tan()

Utilisation

```
tan(angle)
```

Description

Fonction mathématique ; renvoie la tangente de l'angle spécifié exprimée en radians sous forme de nombre à virgule flottante.

Dans la syntaxe JavaScript, utilisez la fonction `tan()` de l'objet Math.

Paramètres

`angle` Requis. Spécifie l'angle à partir duquel une tangente est calculée.

Exemple

La fonction suivante renvoie la tangente de pi/4 :

```
tan (PI/4.0) = 1
```

Le symbole π ne peut pas être utilisé dans une expression Lingo.

Voir aussi

`PI`

tellStreamStatus()

Utilisation

`tellStreamStatus(`*`booléenActivéOuDésactivé`*`)`

Description

Fonction ; active (`TRUE`) ou désactive (`FALSE`) le rapport de l'état de la lecture en flux continu.

La forme `tellStreamStatus()` détermine l'état du gestionnaire.

Lorsque `streamStatusHandler` a la valeur `TRUE`, l'activité de lecture en flux continu sur Internet provoque des appels périodiques du script de l'animation, déclenchant le gestionnaire `streamStatusHandler`. Le gestionnaire est alors exécuté, Director remplissant automatiquement les paramètres avec des informations concernant l'évolution des téléchargements.

Paramètres

`booléenActivéOuDésactivé` Facultatif. Spécifie l'état du gestionnaire.

Exemple

Le gestionnaire `on prepareMovie` suivant active le gestionnaire `on streamStatus` au démarrage de l'animation :

```
-- Syntaxe Lingo
on prepareMovie
  tellStreamStatus(TRUE)
end

// Syntaxe JavaScript
function prepareMovie() {
  tellStreamStatus(TRUE);
}
```

L'instruction suivante détermine l'état du gestionnaire d'état de la lecture en flux continu :

```
-- Syntaxe Lingo
on mouseDown
  put tellStreamStatus()
end

// Syntaxe JavaScript
function mouseDown() {
  put(tellStreamStatus());
}
```

Voir aussi

`on streamStatus`

tellTarget()

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.tellTarget(nomDeCible)

// Syntaxe JavaScript
réfObjImageObjet.tellTarget(nomDeCible);
```

Description

Commande ; équivalente aux méthodes `beginTellTarget` et `endTellTarget` de Flash. La commande `tellTarget()` permet à l'utilisateur de configurer un scénario principal sur lequel agiront les commandes d'image-objet suivantes. Une fois qu'un clip d'animation ou un niveau contenant une animation Flash chargée a été configuré en tant que cible, certaines commandes agissent sur le composant cible, plutôt que sur le scénario principal. Pour les obliger à agir sur le scénario principal, appelez la commande `endTellTarget()`.

Le seul argument valide de `tellTarget` est le nom de la cible. Il n'existe aucun argument valide pour `endTellTarget`.

Les fonctions d'image-objet Flash affectées par `tellTarget` sont `stop`, `play`, `getProperty`, `setProperty`, `gotoFrame`, `call(image)` et `find(libellé)`. En outre, la propriété d'image-objet `frame` (qui renvoie l'image courante) est affectée par `tellTarget`.

Paramètres

`nomDeCible` Requis. Spécifie le nom de la cible.

Exemple

La commande suivante définit le clip comme cible :

```
-- Syntaxe Lingo
sprite(1).tellTarget("\monClip")

// Syntaxe JavaScript
sprite(1).tellTarget("\monClip");
```

La commande suivante arrête le clip :

```
-- Syntaxe Lingo
sprite(1).stop()

// Syntaxe JavaScript
sprite(1).stop();
```

La commande suivante entraîne la lecture du clip :

```
-- Syntaxe Lingo
sprite(1).play()

// Syntaxe JavaScript
sprite(1).play();
```

La commande suivante entraîne le retour au scénario principal :

```
-- Syntaxe Lingo
sprite(1).endTellTarget()

// Syntaxe JavaScript
sprite(1).endTellTarget();
```

La commande suivante arrête l'animation principale :

```
-- Syntaxe Lingo
sprite(1).stop()

// Syntaxe JavaScript
sprite(1).stop();
```

texture()

Utilisation

```
member(quelActeur).texture(quelleTexture)
member(quelActeur).texture[index]
member(quelActeur).shader(quelMatériau).texture
member(quelActeur).model(quelModèle).shader.texture
member(quelActeur).model(quelModèle).shaderList.texture
member(quelActeur).model(quelModèle).shaderList[index].texture
member(quelActeur).modelResource(quelModèleDeSystèmeDeParticules\
  Ressource).texture
```

Description

Propriété 3D d'élément et de matériau ; objet image utilisé par un matériau pour définir l'apparence de la surface d'un modèle. L'image est enrobée sur la géométrie du modèle par le matériau.

Le composant visible d'un matériau est créé avec un maximum de huit couches de textures. Ces huit couches de texture ont été créées à partir d'acteurs bitmap ou d'objets image dans Director ou importées avec des modèles de programmes de modélisation 3D.

Créez et supprimez des textures avec les commandes `newTexture()` et `deleteTexture()`.

Les textures sont enregistrées dans la palette des textures de l'acteur 3D. Elles peuvent être référencées par nom (*quelleTexture*) ou par index de palette (*index*). Une texture peut être utilisée par n'importe quel nombre de matériaux. Les modifications apportées à une texture apparaissent dans tous les matériaux qui l'utilisent.

Il existe trois types de textures :

`#fromCastmember` ; la texture est créée à partir d'un acteur bitmap avec la commande `newTexture()`.

`#fromImageObject` ; la texture est créée à partir d'un objet image Lingo avec la commande `newTexture()`.

`#importedFromFile` ; la texture est importée avec un modèle à partir d'un programme de modélisation 3D.

La texture d'un système de particules est une propriété de la ressource de modèle, dont le type est `#particle`.

Exemple

L'instruction suivante définit la propriété `texture` du matériau surfaceDuMur sur la texture peintureBleue :

```
member("Pièce").shader("surfaceDuMur").texture = \
  member("Pièce").texture("peintureBleue")
```

Voir aussi

`newTexture`, `deleteTexture`

time() (Système)

Utilisation

```
-- Syntaxe Lingo
_system.time()

// Syntaxe JavaScript
_system.time();
```

Description

Méthode du système ; renvoie l'heure courante de l'horloge du système sous forme de chaîne.

L'heure est renvoyée dans le format suivant :

```
1:30 PM
```

Paramètres

Aucun.

Exemple

Le gestionnaire suivant renvoie l'heure courante dans un champ texte :

```
-- Syntaxe Lingo
on exitFrame
  member("horloge").text = _system.time()
end

// Syntaxe JavaScript
function exitFrame() {
  member("horloge").text = _system.time();
}
```

Voir aussi

date() (Système), Système

timeout()

Utilisation

```
-- Syntaxe Lingo
timeout(nomObjDeTemporisation)

// Syntaxe JavaScript
timeout(nomObjDeTemporisation);
```

Description

Fonction du niveau supérieur ; renvoie un objet de temporisation donné.

Utilisez la méthode `new()` pour créer un objet de temporisation et l'ajouter à la liste `timeoutList`.

Paramètres

nomObjDeTemporisation Requis. Chaîne qui spécifie le nom de l'objet de temporisation à renvoyer.

Exemple

Le gestionnaire suivant supprime l'objet de temporisation appelé Foudre aléatoire :

```
-- Syntaxe Lingo
on exitFrame
  timeout("Foudre aléatoire").forget()
end

// Syntaxe JavaScript
function exitFrame() {
  timeout("Foudre aléatoire").forget();
}
```

Voir aussi

new(), timeoutList

titleMenu()

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.titleMenu()

// Syntaxe JavaScript
réfObjDvd.titleMenu();
```

Description

Méthode de DVD ; affiche le menu des titres.

Paramètres

Aucun.

Voir aussi

DVD

top (3D)

Utilisation

référenceDobjetDeRessourceDeModèle.top

Description

Commande 3D ; utilisée avec une ressource de modèle de type #box, cette commande permet d'obtenir et de définir la propriété top de la ressource de modèle.

La propriété top détermine si le haut de la boîte est fermé (TRUE) ou ouvert (FALSE). La valeur par défaut est TRUE.

Paramètres

Aucun.

Exemple

L'instruction suivante donne à la propriété `top` de la ressource de modèle boîteAcadeau la valeur `FALSE`, ce qui signifie que le haut de la boîte sera ouvert.

```
member("Univers 3D").modelResource("boîteAcadeau").top = FALSE
```

Voir aussi

`back`, `bottom (3D)`, `front`

topCap

Utilisation

`référenceDobjetDeRessourceDeModèle``.topCap`

Description

Commande 3D ; utilisée avec une ressource de modèle de type `#cylinder`, cette propriété permet d'obtenir ou de définir la propriété `topCap` de la ressource de modèle.

La propriété `topCap` détermine si l'extrémité du cylindre est fermée (`TRUE`) ou ouverte (`FALSE`). La valeur par défaut de cette propriété est `FALSE`.

Paramètres

Aucun.

Exemple

L'instruction suivante donne à la propriété `topCap` de la ressource de modèle Tube la valeur `FALSE`, ce qui signifie que l'extrémité de ce cylindre sera ouverte.

```
member("Univers 3D").modelResource("Tube").topCap = FALSE
```

topRadius

Utilisation

`référenceDobjetDeRessourceDeModèle``.topRadius`

Description

Commande 3D ; utilisée avec une ressource de modèle de type `#cylinder`, cette commande permet d'obtenir et de définir la propriété `topRadius` de la ressource de modèle, sous la forme d'une valeur à virgule flottante.

La propriété `topRadius` détermine le rayon de l'extrémité supérieure du cylindre. Cette propriété doit toujours être supérieure ou égale à 0.0. La valeur par défaut est 25.0. Le fait de donner à `topRadius` la valeur 0.0 produit un cône.

Paramètres

Aucun.

Exemple

L'instruction suivante donne à la propriété `topRadius` de la ressource de modèle Tube la valeur 0.0. Si le rayon inférieur a une valeur supérieure à 0, les modèles utilisant Tube seront coniques.

```
member("Univers 3D").modelResource("tube").topRadius = 0.0
```

trace()

Utilisation

```
-- Syntaxe Lingo
trace(valeur)

// Syntaxe JavaScript
trace(valeur);
```

Description

Fonction du niveau supérieur ; évalue une expression et affiche le résultat dans la fenêtre Messages.

La fonctionnalité de cette méthode est identique à celle de la méthode `put()` du niveau supérieur, disponible à la fois pour Lingo et la syntaxe JavaScript.

Cette méthode peut servir d'outil de débogage pour suivre la valeur des variables au cours de la lecture d'une animation.

Paramètres

`valeur` Requis. Expression à évaluer.

Exemple

L'instruction suivante renvoie la valeur de la variable `compteur` à la fenêtre Messages.

```
-- Syntaxe Lingo
compteur = (_system.milliseconds / 1000)
trace(compteur)

// Syntaxe JavaScript
var compteur = (_system.milliseconds / 1000);
trace(compteur);
```

Voir aussi

`put()`

transform (commande)

Utilisation

```
transform()
transform(n1,n2,n3, ... ,n14,n15,n16)
```

Description

Commande 3D ; crée un objet de transformation. Lorsque cette commande est utilisée sans fournir d'autres paramètres, elle crée un objet de transformation égal à la transformation d'identité. La transformation d'identité possède des composants de position et de rotation de vector(0,0,0) et un composant d'échelle de vector(1,1,1). Lorsque cette commande est utilisée en fournissant 16 paramètres sous la forme de *`n1`*,*`n2`*,*`n3`*, ... ,*`n14`*,*`n15`*,*`n16`*, elle crée un objet de transformation utilisant les 16 entrées.

Paramètres

`n1, n2, n3, ... ,n14, n15, n16` Facultatif. Spécifie les seize paramètres utilisés comme données de transformation lors de la création d'un objet de transformation.

Exemple

L'instruction suivante crée une transformation d'identité et l'enregistre dans la variable tTransformation :

```
tTransformation = transform()
```

L'instruction suivante crée une transformation d'identité en spécifiant ses 16 éléments et enregistre la nouvelle transformation dans la variable tTransformation.

```
tTransformation = transform(1.0000,0.0000,0.0000,0.0000, \
  0.0000,1.0000,0.0000,0.0000, 0.0000,0.0000,1.0000,0.0000, \
  0.0000,0.0000,0.0000,1.0000)
```

L'instruction suivante crée une transformation personnalisée en spécifiant ses 16 éléments et enregistre la nouvelle transformation dans la variable tTransformation. La transformation créée a une propriété `position` de `vector(19.2884, 1.7649, 4.2426)`, une propriété `rotation` de `vector(75.7007, 0.0000, -6.5847)` et une propriété `scale` de `vector(0.4904, 0.7297, 0.3493)`.

```
tTransformation = transform(0.4872,-0.0562,0.0000,0.0000, \
  0.0795,0.1722,0.7071,0.0000, -0.0795,-0.1722,0.7071,0.0000, \
  19.2884,1.7649,4.2426,1.0000)
```

Voir aussi

transform (propriété), preRotate, preTranslate(), preScale(), rotate, translate, scale (commande)

translate

Utilisation

```
member(quelActeur).node(quelNœud).translate(incrémentX, \
  incrémentY, incrémentZ {, parRapportA})
member(quelActeur).node(quelNœud).translate\
  (vecteurDeTranslation {, parRapportA})
transformation.translate(incrémentX, incrémentY, incrémentZ\
  {, parRapportA})
transformation.translate(vecteurDeTranslation {, parRapportA})
```

Description

Commande 3D ; applique une translation après les décalages de position, de rotation et d'échelle courants d'un objet de transformation d'un nœud référencé ou d'un objet de transformation directement référencé. La translation doit être spécifiée sous la forme d'un jeu de trois incréments le long des trois axes correspondants. Ces incréments peuvent être spécifiés explicitement sous la forme *incrémentX*, *incrémentY* et *incrémentZ*, ou par un *vecteurDeTranslation*, dont le composant x correspond à la translation sur l'axe des x, y sur l'axe des y et z sur l'axe des y.

Un nœud peut être une caméra, un modèle, une lumière ou un groupe.

Paramètres

incrémentX Requis si vous spécifiez un ensemble de trois incréments. Spécifie l'incrément sur l'axe des *x*.

incrémentY Requis si vous spécifiez un ensemble de trois incréments. Spécifie l'incrément sur l'axe des *y*.

incrémentZ Requis si vous spécifiez un ensemble de trois incréments. Spécifie l'incrément sur l'axe des *z*.

vecteurDeTranslation Requis si vous spécifiez un vecteur. Spécifie le vecteur qui contient les composants *x*, *y* et *z*.

parRapportA Facultatif. Détermine les axes du système de coordonnées utilisés pour appliquer les modifications de translation voulues. Le paramètre *parRapportA* peut avoir les valeurs suivantes :

- #self applique les incréments en fonction du système de coordonnées local du nœud (les axes des x, des y et des z spécifiés pour le modèle au cours de la programmation). Cette valeur est utilisée comme valeur par défaut si vous utilisez la commande translate avec une référence de nœud et que le paramètre *parRapportA* n'est pas spécifié.
- #parent applique les incréments par rapport au système de coordonnées du parent du nœud. Cette valeur est utilisée comme valeur par défaut si vous utilisez la commande translate avec une référence de transformation et que le paramètre *parRapportA* n'est pas spécifié.
- #world applique les incréments par rapport au système de coordonnées de l'univers. Lorsque le parent d'un modèle est l'univers, ceci est équivalent à l'utilisation de #parent.
- nodeReference permet de spécifier un nœud servant de base à la translation, la commande appliquant les translations en fonction du système de coordonnées du nœud spécifié.

Exemple

L'exemple suivant construit une transformation à l'aide de la commande transform, puis initialise la position et l'orientation de la transformation dans l'espace avant d'affecter la transformation au modèle nommé Mars. Cet exemple indique ensuite la position résultante du modèle.

```
t = transform()
t.transform.identity()
t.transform.rotate(0, 90, 0)
t.transform.translate(100, 0, 0)
gbModèle = member("Scène").model("Mars")
gbModel.transform = t
put gbModèle.transform.position
-- vector( 0.0000, 100.0000, 0.0000 )
```

L'instruction suivante déplace le modèle Bip de 20 unités le long de l'axe des x de son nœud parent.

```
put member("Scène").model("Bip").position
-- vector( -38.5000, 21.2500, 2.0000 )
member("Scène").model("Bip").translate(20, 10, -0.5)
put member("Scène").model("Bip").position
-- vector( -18.5000, 31.2500, 1.5000 )
```

Voir aussi

transform (propriété), preTranslate(), scale (commande), rotate

union()

Utilisation

```
rect(1).union(rect(2))
union (rect1, rect2)
```

Description

Fonction ; renvoie le plus petit rectangle qui inclut les deux rectangles.

Paramètres

rect2 Requis. Spécifie le second rectangle.

Exemple

L'instruction suivante renvoie le rectangle qui inclut les rectangles spécifiés :

```
put union (rect (0, 0, 10, 10), rect (15, 15, 20, 20))
--rect (0, 0, 20, 20)
```

or

```
put rect(0, 0, 10, 10).union(rect(15, 15, 20, 20))
--rect (0, 0, 20, 20)
```

Voir aussi

map(), rect()

unLoad() (acteur)

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.unLoad({réfObjDernierActeur})

// Syntaxe JavaScript
réfObjActeur.unLoad({réfObjDernierActeur});
```

Description

Méthode d'acteur ; force Director à purger les acteurs spécifiés de la mémoire.

Director purge automatiquement les acteurs les moins récemment utilisés pour permettre l'utilisation des méthodes `preLoad()` ou le chargement normal de la distribution.

- Si elle est utilisée sans argument, la méthode `unLoad()` purge de la mémoire les acteurs présents dans toutes les images d'une animation.
- Si elle est utilisée avec le paramètre *réfObjDernierActeur*, la méthode `unLoad()` purge de la mémoire tous les acteurs dans la plage spécifiée.

Si elle est utilisée dans une nouvelle animation sans acteurs chargés, cette méthode renvoie une erreur.

Les acteurs que vous avez modifiés au cours de la programmation ou en définissant `picture`, `pasteClipBoadInto()`, et ainsi de suite, ne peuvent pas être purgés.

Paramètres

réfObjDernierActeur Facultatif. Référence au dernier acteur de la plage à purger de la mémoire.

Exemple

L'instruction suivante purge de la mémoire l'acteur Navires :

```
-- Syntaxe Lingo
member("Navires").unLoad()

// Syntaxe JavaScript
member("Navires").unLoad();
```

Cette instruction purge de la mémoire les acteurs 10 à 15 :

```
-- Syntaxe Lingo
member(10).unLoad(15)

// Syntaxe JavaScript
member(10).unLoad(15);
```

Voir aussi

Acteur

unLoad() (animation)

Utilisation

```
-- Syntaxe Lingo
_movie.unLoad({entDeNumDimage} {, entANumDimage})

// Syntaxe JavaScript
_movie.unLoad({entDeNumDimage} {, entANumDimage});
```

Description

Méthode d'animation ; supprime l'animation préchargée spécifiée de la mémoire.

Cette commande est utile pour forcer la purge des animations lorsque la mémoire diminue.

Vous pouvez utiliser une adresse URL comme référence de fichier.

Si l'animation n'est pas encore dans la RAM, le résultat est -1.

Paramètres

entDeNumDimage Facultatif. Nombre entier qui spécifie le numéro de la première image d'une plage à purger de la mémoire.

entANumDimage Facultatif. Nombre entier qui spécifie le numéro de la dernière image d'une plage à purger de la mémoire.

Exemple

Les instructions suivantes purgent les images 10 à 25 de la mémoire.

```
-- Syntaxe Lingo
_movie.unLoad(10, 25)

// Syntaxe JavaScript
_movie.unLoad(10, 25);
```

Voir aussi

Animation

unLoadMember()

Utilisation

```
-- Syntaxe Lingo
_movie.unLoadMember({réfObjActeur})
_movie.unLoadMember(deNomOuNumDacteur, àNomOuNumDacteur)

// Syntaxe JavaScript
_movie.unLoadMember({réfObjActeur});
_movie.unLoadMember(deNomOuNumDacteur, àNomOuNumDacteur);
```

Description

Méthode d'animation ; force Director à purger de la mémoire les acteurs utilisés dans une image spécifiée. Director purge automatiquement les acteurs les moins récemment utilisés pour permettre l'utilisation des méthodes `preLoad()` ou le chargement normal de la distribution.

- Si vous ne précisez pas d'argument, la méthode `unLoadMember()` purge de la mémoire les acteurs figurant dans toutes les images d'une animation.
- Si vous utilisez un argument, *`réfObjActeur`*, la méthode `unLoadMember()` purge de la mémoire les acteurs contenus dans cette image.
- Si vous utilisez deux arguments, *`deNomOuNumDacteur`* et *`àNomOuNumDacteur`*, la méthode `unLoadMember()` purge de la mémoire tous les acteurs contenus dans la plage spécifiée. Vous pouvez spécifier une plage de membres par numéros ou par libellés d'images.

Paramètres

`réfObjActeur` Facultatif. Référence à l'acteur qui doit être purgé de la mémoire.

`deNomOuNumDacteur` Requis si vous purgez une plage d'acteurs. Chaîne ou nombre entier qui spécifie le nom ou le numéro du premier acteur d'une plage à purger de la mémoire.

`àNomOuNumDacteur` Requis si vous purgez une plage d'acteurs. Chaîne ou nombre entier qui spécifie le nom ou le numéro du dernier acteur d'une plage à purger de la mémoire.

Exemple

L'instruction suivante purge de la mémoire l'acteur Ecran1 :

```
-- Syntaxe Lingo
_movie.unLoadMember(member("Ecran1"))

// Syntaxe JavaScript
_movie.unLoadMember(member("Ecran1"));
```

L'instruction suivante purge de la mémoire tous les acteurs compris entre l'acteur 1 et l'acteur Cinémascope :

```
-- Syntaxe Lingo
_movie.unLoadMember(member(1), member("Cinémascope"))

// Syntaxe JavaScript
_movie.unLoadMember(member(1), member("Cinémascope"));
```

Voir aussi

Animation

unLoadMovie()

Utilisation

```
-- Syntaxe Lingo
_movie.unLoadMovie(chaîneNomDanimation)

// Syntaxe JavaScript
_movie.unLoadMovie(chaîneNomDanimation);
```

Description

Méthode d'animation ; supprime l'animation préchargée spécifiée de la mémoire.

Cette commande est utile pour forcer la purge des animations lorsque la mémoire diminue.

Vous pouvez utiliser une adresse URL comme référence de fichier.

Si l'animation n'est pas encore dans la RAM, le résultat est -1.

Paramètres

chaîneNomDanimation Requis. Chaîne qui spécifie le nom de l'animation à purger de la mémoire.

Exemple

L'instruction suivante vérifie si le plus grand bloc contigu de mémoire disponible est inférieur à 100 Ko et, le cas échéant, purge l'animation Parsifal :

```
-- Syntaxe Lingo
if (_system.freeBlock < (100*1024)) then
  _movie.unLoadMovie("Parsifal")
end if

// Syntaxe JavaScript
if (_system.freeBlock < (100*1024)) {
  _movie.unLoadMovie("Parsifal");
}
```

L'instruction suivante purge l'animation à http://www.cbDemille.com/SunsetBlvd.dir :

```
-- Syntaxe Lingo
_movie.unLoadMovie("http://www.cbDemille.com/SunsetBlvd.dir")

// Syntaxe JavaScript
_movie.unLoadMovie("http://www.cbDemille.com/SunsetBlvd.dir");
```

Voir aussi

Animation

unregisterAllEvents

Utilisation

```
-- Syntaxe Lingo
member(quelActeur).unregisterAllEvents()

// Syntaxe JavaScript
member(quelActeur).unregisterAllEvents();
```

Description

Commande 3D ; annule l'enregistrement de l'acteur référencé pour toutes les notifications d'événements. Tous les gestionnaires précédemment enregistrés pour répondre aux événements utilisant la commande `registerForEvent` ne seront donc plus déclenchés lors de ces événements.

Paramètres

Aucun.

Exemple

L'instruction suivante annule l'enregistrement de l'acteur Scène pour toutes les notifications d'événements.

```
-- Syntaxe Lingo
member("Scène").unregisterAllEvents()

// Syntaxe JavaScript
member("Scène").unregisterAllEvents();
```

Voir aussi

`registerForEvent()`

update

Utilisation

```
-- Syntaxe Lingo
member(quelActeur).model(quelModèle).update

// Syntaxe JavaScript
member(quelActeur).model(quelModèle).update();
```

Description

Commande 3D ; entraîne la mise à jour des animations du modèle sans rendu. Utilisez cette commande pour déterminer la position exacte d'un modèle animé avec un script.

Paramètres

Aucun.

updateFrame()

Utilisation

```
-- Syntaxe Lingo
_movie.updateFrame()

// Syntaxe JavaScript
_movie.updateFrame();
```

Description

Méthode d'animation ; pendant la création du scénario uniquement, entre les changements apportés à l'image courante pendant l'enregistrement du scénario et passe à l'image suivante. N'importe quel objet qui était déjà dans l'image quand la session de mise à jour a commencé reste dans l'image. Vous devez émettre une méthode `updateFrame()` pour chaque image que vous mettez à jour.

Paramètres

Aucun.

Exemple

Lorsque utilisée dans le gestionnaire suivant, la commande `updateFrame` entre les changements apportés à l'image courante et passe à l'image suivante à chaque fois que le code atteint la fin de la boucle de répétition. Le nombre des images est déterminé par l'argument `nombreDimages`.

```
-- Syntaxe Lingo
on animBalle(nombreDimages)
  _movie.beginRecording()
  horizontal = 0
  vertical = 100
  repeat with i = 1 to nombreDimages
    _movie.go(i)
    sprite(20).member = member("Balle").number
    sprite(20).locH = horizontal
    sprite(20).locV = vertical
    sprite(20).forecolor = 255
    horizontal = horizontal + 3
    vertical = vertical + 2
    _movie.updateFrame()
  end repeat
  _movie.endRecording()
end animBalle
```

```
// Syntaxe JavaScript
function animBalle(nombreDimages) {
  _movie.beginRecording();
  var horizontal = 0;
  var vertical = 100;
  for (var i = 1; i <= nombreDimages; i++) {
    _movie.go(1);
    sprite(20).member = member("Balle");
    sprite(20).locH = horizontal;
    sprite(20).locV = vertical;
    sprite(20).foreColor = 255;
    horizontal = horizontal + 3;
    vertical = vertical + 2;
    _movie.updateFrame();
  }
  _movie.endRecording();
}
```

Voir aussi

beginRecording(), endRecording(), Animation, scriptNum, tweened

updateStage()

Utilisation

```
-- Syntaxe Lingo
_movie.updateStage()

// Syntaxe JavaScript
_movie.updateStage();
```

Description

Méthode d'animation ; rafraîchit la scène immédiatement au lieu de la rafraîchir entre les images.

La méthode `updateStage()` rafraîchit les images-objets, effectue les transitions, lit des sons, envoie un message `prepareFrame` (affectant les scripts d'animations et de comportements) et un message `stepFrame` (affectant `actorList`).

Paramètres

Aucun.

Exemple

Le gestionnaire suivant change la position horizontale et verticale de l'image-objet et rafraîchit la scène de façon à ce que l'image-objet apparaisse à son nouvel emplacement sans devoir attendre le déplacement de la tête de lecture :

```
-- Syntaxe Lingo
on déplacementVersLaDroite(quelleImageObjet, quelleDistance)
  sprite(quelleImageObjet).locH = sprite(quelleImageObjet).locH +
  quelleDistance
  _movie.updateStage()
end déplacementVersLaDroite
```

```
// Syntaxe JavaScript
function déplacementVersLaDroite(quelleImageObjet, quelleDistance) {
  sprite(quelleImageObjet).locH = sprite(quelleImageObjet).locH +
  quelleDistance;
  _movie.updateStage();
}
```

Voir aussi

actorList, Animation, on prepareFrame, on stepFrame

URLEncode

Utilisation

URLEncode(*listeDePropriétésOuChaîne* {, *chaîneOSduServeur*} {, *jeuDeCaractères*})

Description

Fonction ; renvoie la chaîne en codage URL pour son premier argument. Permet l'utilisation de paramètres CGI dans d'autres commandes. La même conversion que pour postNetText et getNetText() est effectuée lorsqu'ils reçoivent une liste de propriétés.

Paramètres

listeDePropriétésOuChaîne Requis. Spécifie la liste de propriétés ou la chaîne à transformer en codage URL.

chaîneOSduServeur Facultatif. Encode les caractères de retour chariot dans *listeDePropriétésOuChaîne*. Par défaut, ce paramètre a la valeur "Unix", mais il peut également prendre la valeur "Win" ou "Mac" ; les retours chariot inclus dans *listeDePropriétésOuChaîne* sont alors convertis dans les caractères utilisés par le serveur. Pour la plupart des applications, ce paramètre n'est pas nécessaire, les ruptures de ligne n'étant généralement pas utilisées dans les réponses de formulaires.

jeuDeCaractères Facultatif. Ce paramère ne s'applique que si l'utilisateur travaille sur un système Shift-JIS (japonais). Ses valeurs possibles sont JIS, EUC, ASCII et AUTO. Les données récupérées sont converties de Shift-JIS dans le jeu de caractères désigné. Les données renvoyées sont traitées de la même façon qu'avec getNetText() (converties du jeu de caractères nommé à Shift-JIS). Si AUTO est utilisé, les données affichées dans le jeu de caractères local ne sont pas converties ; les résultats renvoyés par le serveur sont convertis comme pour getNetText(). ASCII est la valeur par défaut si *jeuDeCaractères* est omis. ASCII n'offre aucune conversion pour l'envoi ou les résultats.

Exemple

Dans l'exemple suivant, la fonction URLEncode fournit la chaîne en codage URL à une requête CGI à l'emplacement spécifié.

```
URL = "http://unServeur/cgi-bin/requête.cgi"
gotonetpage URL & "?" & URLEncode( [#name: "Jean", #hobby: "Quoi ?"] )
```

Voir aussi

getNetText(), postNetText

value()

Utilisation

`value(`*`expressionChaîne`*`)`

Description

Fonction ; renvoie la valeur d'une chaîne. Lorsque `value()` est appelé, Lingo analyse l'*`expressionChaîne`* fournie et renvoie sa valeur logique.

Toute expression Lingo pouvant être affichée (`put`) dans la fenêtre Messages ou définie comme valeur d'une variable peut également être utilisée avec `value()`.

Les deux instructions Lingo suivantes sont équivalentes :

```
put sprite(2).member.duration * 5
put value("sprite(2).member.duration * 5")
```

Les deux instructions Lingo suivantes sont également équivalentes :

```
x = (the mouseH - 10) / (the mouseV + 10)
x = value("(the mouseH - 10) / (the mouseV + 10)")
```

Les expressions que Lingo ne peut pas analyser produiront des résultats inattendus, mais ne produiront pas d'erreurs Lingo. Le résultat est la valeur de la portion initiale de l'expression jusqu'à la première erreur de syntaxe détectée dans la chaîne.

La fonction `value()` est pratique pour l'analyse d'expressions placées dans le texte par les utilisateurs, les expressions de chaîne passées à Lingo par des Xtras ou toute autre expression nécessaire à la conversion d'une chaîne en valeur Lingo.

Gardez à l'esprit que dans certaines situations, l'utilisation de `value()`, soumise aux actions des utilisateurs, peut être dangereuse, par exemple lors de la saisie du nom d'un gestionnaire personnalisé dans le champ. Ceci engendrerait l'exécution du gestionnaire lors de son transfert à `value()`.

Ne confondez pas les actions de la fonction value() avec celles des fonctions `integer()` et `float()`.

Paramètres

`expressionChaîne` Requis. Spécifie la chaîne à partir de laquelle une valeur est renvoyée. Cette chaîne peut être constituée de n'importe quelle expression reconnue par Lingo.

Exemple

L'instruction suivante affiche la valeur numérique de la chaîne `"the sqrt of" && "2.0"` :

```
put value("the sqrt of" && "2.0")
```

Le résultat est 1.4142.

L'instruction suivante affiche la valeur numérique de la chaîne Centime :

```
put value("Centime")
```

Le résultat affiché dans la fenêtre Messages est le mot `VOID`, *Centime* n'ayant pas de valeur numérique.

Vous pouvez utiliser la syntaxe suivante pour convertir une chaîne ayant le format d'une liste en liste véritable :

```
maChaîne = "[" & QUOTE & "chat" & QUOTE & ", " & QUOTE & "chien" & QUOTE & "]"
maListe = value(maChaîne)
put maListe
-- ["chat", "chien"]
```

Cela permet de placer une liste dans un champ ou un acteur texte, puis de l'extraire et de la reformater facilement sous forme de liste.

L'instruction suivante analyse la chaîne "3 5" et renvoie la valeur de la portion de la chaîne reconnue par Lingo :

```
put value("3 5")
-- 3
```

Voir aussi

`string()`, `integer()`, `float()`

vector()

Utilisation

```
-- Syntaxe Lingo
vector()
vector(entX, entY, entZ)

// Syntaxe JavaScript
vector();
vector(entX, entY, entZ);
```

Description

Fonction du niveau supérieur et type de données. Décrit un point dans l'espace 3D en fonction de trois paramètres, qui sont les distances spécifiques des points de référence le long des axes des *x*, des *y* et des *z*, respectivement.

Si le vecteur se trouve dans l'espace de l'univers, le point de référence est l'origine de l'univers, `vector(0, 0, 0)`. Si le vecteur se trouve dans l'espace de l'objet, le point de référence est la position et l'orientation de l'objet.

Cette méthode renvoie un objet vectoriel.

Les valeurs des vecteurs peuvent être manipulées à l'aide des opérateurs `+`, `-`, `*` et `/`. Consultez la définition des différents opérateurs pour plus d'informations.

Paramètres

`entX` Facultatif. Nombre entier qui spécifie le point sur l'axe des *x*.

`entY` Facultatif. Nombre entier qui spécifie le point sur l'axe des *y*.

`entZ` Facultatif. Nombre entier qui spécifie le point sur l'axe des *z*.

Exemple

L'instruction suivante crée un vecteur et l'affecte à la variable `monVecteur` :

```
-- Syntaxe Lingo
monVecteur = vector(10.0, -5.0, 0.0)

// Syntaxe JavaScript
var monVecteur = vector(10.0, -5.0, 0.0);
```

En Lingo uniquement, l'instruction suivante ajoute deux vecteurs et affecte la valeur résultante à la variable `ceVecteur` :

```
-- Syntaxe Lingo
ceVecteur = vector(1.0, 0.0, 0.0) + vector(0.0, -12.5, 2.0)
```

version()

Utilisation

```
-- Syntaxe Lingo
réfObjFileio.version()

// Syntaxe JavaScript
réfObjFileio.version();
```

Description

Méthode de Fileio ; affiche les informations de version de Fileio dans la fenêtre Messages.

Paramètres

Aucun.

Voir aussi

`Fileio`

voiceCount()

Utilisation

```
voiceCount()
```

Description

Fonction : renvoie le nombre de voix installées disponibles pour le logiciel de conversion de texte en voix. La valeur renvoyée est un nombre entier. Ce nombre de voix peut être utilisé avec `voiceSet()` et `voiceGet()` pour activer une voix spécifique.

Paramètres

Aucun.

Exemple

L'instruction suivante définit la variable `nombreDeVoix` sur le nombre de voix de la fonction de conversion de texte en voix :

```
nombreDeVoix = voiceCount()
```

Voir aussi

`voiceInitialize()`, `voiceSet()`, `voiceGet()`

voiceGet()

Utilisation

```
voiceGet()
```

Description

Fonction ; renvoie une liste de propriétés décrivant la voix courante utilisée pour la conversion de texte en voix. La liste contient les propriétés suivantes :

- `#name` indique le nom de la voix installée.
- `#age` indique l'âge de la voix. Cette valeur est une chaîne. Les valeurs possibles sont « Teen », « Adult », « Toddler » et « Senior », ainsi que des valeurs numériques telles que « 35 ». Les valeurs réelles dépendent du système d'exploitation, de la version du logiciel de conversion de texte en voix et des voix installées.
- `#gender` indique si la voix est celle d'une femme ou d'un homme. Cette valeur est une chaîne.
- `#index` indique la position de la voix dans la liste des voix installées. Vous pouvez faire référence à une voix par son index lors de l'utilisation de la commande `voiceSet()`.

Utilisez `voiceCount()` pour déterminer le nombre de voix disponibles.

Paramètres

Aucun.

Exemple

L'instruction suivante définit la variable `vieilleVoix` sur la liste de propriétés décrivant la voix courante de la fonction de conversion de texte en voix :

```
vieilleVoix = voiceGet()
```

L'instruction suivante affiche la liste des propriétés de la voix courante de la fonction de conversion de texte en voix :

```
put voiceGet()
-- [#name: "Marie", #age: "teen", #gender: "female", #index: 5]
```

Voir aussi

`voiceInitialize()`, `voiceCount()`, `voiceSet()`, `voiceGet()`

voiceGetAll()

Utilisation

```
voiceGetAll()
```

Description

Fonction ; renvoie la liste des voix disponibles installées sur l'ordinateur. Cette liste est composée de listes de propriétés, une pour chaque voix disponible.

Chaque liste de propriétés contient les propriétés suivantes :

- `#name` indique le nom de la voix installée.

- `#age` indique l'âge de la voix. Cette valeur est une chaîne. Les valeurs possibles sont « Teen », « Adult », « Toddler » et « Senior », ainsi que des valeurs numériques telles que « 35 ». Les valeurs réelles dépendent du système d'exploitation, de la version du logiciel de conversion de texte en voix et des voix installées.
- `#gender` indique si la voix est celle d'une femme ou d'un homme.
- `#index` indique la position de la voix dans la liste des voix installées. Vous pouvez faire référence à une voix par son index lors de l'utilisation de la commande `voiceSet()`.

Vous pouvez également utiliser `voiceCount()` pour déterminer le nombre de voix disponibles.

Paramètres

Aucun.

Exemple

L'instruction suivante définit la variable `voixActuelles` sur la liste des voix installées sur l'ordinateur :

```
voixActuelles = voiceGetAll()
```

L'instruction suivante affiche la liste des propriétés décrivant chacune des voix actuellement installées de la fonction de conversion de texte en voix :

```
put voiceGetAll()
-- [[#name: "Marie", #age: "teen", #gender: "female", #index: 1], [#name:
  "Joe", #age: "adult", #gender: "male", #index: 2]]
```

Voir aussi

`voiceInitialize()`, `voiceCount()`, `voiceSet()`, `voiceGet()`

voiceGetPitch()

Utilisation

```
voiceGetPitch()
```

Description

Fonction ; renvoie la tonalité actuelle de la voix courante, sous forme de nombre entier. La plage des valeurs valides dépend du système d'exploitation et du logiciel vocal.

Paramètres

Aucun.

Exemple

Les instructions suivantes contrôlent si la tonalité de la voix courante est supérieure à 10 et la redéfinissent sur 10 si c'est le cas :

```
-- Syntaxe Lingo
if voiceGetPitch() > 10 then
  voiceSetPitch(10)
end if

// Syntaxe JavaScript
if (voiceGetPitch() > 10) {
  voiceSetPitch(10);
}
```

Voir aussi

voiceSpeak(), voicePause(), voiceResume(), voiceStop(), voiceGetRate(), voiceSetRate(), voiceSetPitch(), voiceGetVolume(), voiceSetVolume(), voiceState(), voiceWordPos()

voiceGetRate()

Utilisation

```
voiceGetRate()
```

Description

Fonction ; renvoie la cadence de lecture courante du logiciel de conversion de texte en voix. La valeur renvoyée est un nombre entier. La plage des valeurs valides dépend du système d'exploitation et du logiciel vocal. Il s'agit généralement de valeurs comprises entre -10 et 10.

Paramètres

Aucun.

Exemple

Les instructions suivantes vérifient que la cadence de synthèse vocale est inférieure à 50 et la définit à 50 le cas échéant :

```
-- Syntaxe Lingo
if voiceGetRate() < 50 then
  voiceSetRate(50)
end if

// Syntaxe JavaScript
if (voiceGetRate() < 50) {
  voiceSetRate(50);
}
```

Voir aussi

voiceSpeak(), voicePause(), voiceResume(), voiceStop(), voiceSetRate(), voiceGetPitch(), voiceSetPitch(), voiceGetVolume(), voiceSetVolume(), voiceState(), voiceWordPos()

voiceGetVolume()

Utilisation

```
voiceGetVolume()
```

Description

Fonction : renvoie le volume courant de la fonction de conversion de texte en voix. La valeur renvoyée est un nombre entier. La plage des valeurs valides dépend du système d'exploitation.

Paramètres

Aucun.

Exemple

Les instructions suivantes vérifient si le volume de la conversion de texte en voix s'élève au moins à 55 et le définissent sur 55 s'il est inférieur :

```
-- Syntaxe Lingo
if voiceGetVolume() < 55 then
  voiceSetVolume(55)
end if

// Syntaxe JavaScript
if (voiceGetVolume() < 55) {
  voiceSetVolume(55);
}
```

Voir aussi

```
voiceSpeak(), voicePause(), voiceResume(), voiceStop(), voiceGetRate(),
  voiceSetRate(), voiceGetPitch(), voiceSetPitch(), voiceSetVolume(),
  voiceState(), voiceWordPos()
```

voiceInitialize()

Utilisation

```
voiceInitialize()
```

Description

Commande ; charge le logiciel de conversion de texte en voix. Si la commande `voiceInitialize()` renvoie la valeur 0, cela signifie que le logiciel de conversion de texte en voix n'est pas présent ou que son chargement a échoué.

Cette commande renvoie la valeur 1 s'il le détecte, et 0 dans le cas contraire.

Paramètres

Aucun.

Exemple

Ces instructions chargent le logiciel de conversion de texte en voix et testent ensuite si ce chargement est complet avant d'utiliser la commande `voiceSpeak()`, qui prononcera la phrase « Bienvenue dans Shockwave. » :

```
-- Syntaxe Lingo
err = voiceInitialize()
if err = 1 then
  voiceSpeak("Bienvenue dans Shockwave")
else
  alert "Le chargement du logiciel vocal a échoué."
end if

// Syntaxe JavaScript
err = voiceInitialize();
if (err == 1) {
  voiceSpeak("Bienvenue dans Shockwave");
} else {
  alert("Le chargement du logiciel vocal a échoué.");
}
```

Voir aussi

`voiceCount()`, `voiceSet()`, `voiceGet()`

voicePause()

Utilisation

`voicePause()`

Description

Commande ; met la sortie vocale en pause. La commande renvoie la valeur 1 en cas de réussite, et 0 dans le cas contraire.

Paramètres

Aucun.

Exemple

Les instructions suivantes provoquent l'interruption de la fonction de conversion de texte en voix lorsque l'utilisateur clique avec la souris :

```
-- Syntaxe Lingo
on mouseUp
  voicePause()
end mouseUp

// Syntaxe JavaScript
function mouseUp() {
  voicePause();
}
```

Voir aussi

`voiceSpeak()`, `voiceResume()`, `voiceStop()`, `voiceGetRate()`, `voiceSetRate()`, `voiceGetPitch()`, `voiceSetPitch()`, `voiceGetVolume()`, `voiceSetVolume()`, `voiceState()`, `voiceWordPos()`

voiceResume()

Utilisation

`voiceResume()`

Description

Commande ; reprend la sortie vocale. La commande renvoie la valeur 1 en cas de réussite, et 0 dans le cas contraire.

Paramètres

Aucun.

Exemple

Les instructions suivantes réactivent la fonction vocale lorsque la tête de lecture se déplace sur l'image suivante dans le scénario :

```
-- Syntaxe Lingo
on exitFrame
  voiceResume()
end exitFrame

// Syntaxe JavaScript
function exitFrame() {
  voiceResume();
}
```

Voir aussi

`voiceSpeak()`, `voicePause()`, `voiceStop()`, `voiceGetRate()`, `voiceSetRate()`, `voiceGetPitch()`, `voiceSetPitch()`, `voiceGetVolume()`, `voiceSetVolume()`, `voiceState()`, `voiceWordPos()`

voiceSet()

Utilisation

`voiceSet(`*`entier`*`)`

Description

Commande : définit la voix courante de la fonction de conversion de texte en voix. Dans ce cas, la commande renvoie la nouvelle valeur définie. Utilisez `voiceCount()` pour déterminer le nombre de voix disponibles.

Paramètres

`entier` Requis. Nombre entier qui spécifie le numéro de la voix à utiliser pour la fonction de conversion de texte en voix. La plage des valeurs valides dépend du nombre de voix installées sur l'ordinateur de l'utilisateur. Si une valeur non admise est spécifiée, la voix sera définie sur la valeur admise la plus proche.

Exemple

L'instruction suivante définit la voix courante de la conversion de texte en voix sur la troisième voix installée sur l'ordinateur :

```
voiceSet(3)
```

Voir aussi

`voiceInitialize()`, `voiceCount()`, `voiceGet()`

voiceSetPitch()

Utilisation

`voiceSetPitch(`*`entier`*`)`

Description

Commande ; définit la tonalité de la voix courante de la fonction de conversion de texte en voix sur la valeur spécifiée. La valeur renvoyée est la nouvelle valeur de tonalité définie.

Paramètres

`entier` Requis. Nombre entier qui spécifie la tonalité de la voix pour la fonction de conversion de texte en voix. La plage des valeurs valides dépend du système d'exploitation et du logiciel vocal.

Exemple

L'instruction suivante définit la tonalité de la voix courante sur 75 :

```
voiceSetPitch(75)
```

Voir aussi

`voiceSpeak()`, `voicePause()`, `voiceResume()`, `voiceStop()`, `voiceGetRate()`, `voiceSetRate()`, `voiceGetPitch()`, `voiceGetVolume()`, `voiceSetVolume()`, `voiceState()`, `voiceWordPos()`

voiceSetRate()

Utilisation

`voiceSetRate(`*`entier`*`)`

Description

Commande ; définit la cadence de la voix courante de la fonction de conversion de texte en voix sur la valeur entière spécifiée. Cette commande renvoie la nouvelle valeur définie.

Paramètres

`entier` Requis. Nombre entier qui spécifie la cadence de lecture pour la fonction de conversion de texte en voix. La plage des valeurs valides dépend du système d'exploitation. En règle générale, les valeurs comprises entre -10 et 10 conviennent à la plupart des logiciels de conversion de texte en voix. Si une valeur non admise est spécifiée, la cadence sera définie sur la valeur admise la plus proche.

Exemple

L'instruction suivante définit la cadence de lecture de la fonction de conversion de texte en voix sur 7.

```
voiceSetRate(7)
```

Voir aussi

voiceSpeak(), voicePause(), voiceResume(), voiceStop(), voiceGetRate(), voiceGetPitch(), voiceSetPitch(), voiceGetVolume(), voiceSetVolume(), voiceState(), voiceWordPos()

voiceSetVolume()

Utilisation

```
voiceSetVolume(entier)
```

Description

Commande ; définit le volume courant de la fonction de conversion de texte en voix.

Paramètres

entier Requis. Nombre entier qui spécifie le volume pour la fonction de conversion de texte en voix. La plage des valeurs valides dépend du système d'exploitation. Dans ce cas, la commande renvoie la nouvelle valeur définie. Si une valeur non admise est spécifiée, le volume sera défini sur la valeur admise la plus proche.

Exemple

L'instruction suivante définit le volume de la fonction de conversion de texte en voix sur 55.

```
voiceSetVolume(55)
```

Voir aussi

voiceSpeak(), voicePause(), voiceResume(), voiceStop(), voiceGetRate(), voiceSetRate(), voiceGetPitch(), voiceSetPitch(), voiceGetVolume(), voiceState(), voiceWordPos()

voiceSpeak()

Utilisation

```
-- Syntaxe Lingo
voiceSpeak("chaîne")

// Syntaxe JavaScript
voiceSpeak("chaîne"); // pas encore documenté
```

Description

Commande ; active la lecture de la chaîne spécifiée par le logiciel de conversion de texte en voix. Si cette commande est utilisée, toute lecture en cours est automatiquement interrompue par la nouvelle chaîne.

Paramètres

chaîne Requis. Chaîne qui doit être lue par la fonction de conversion de texte en voix.

Exemple

L'instruction suivante entraîne la lecture de la chaîne « Bienvenue dans Shockwave » par le logiciel de conversion de texte en voix :

```
voiceSpeak("Bienvenue dans Shockwave")
```

Voir aussi

voiceSpeak(), voicePause(), voiceResume(), voiceStop(), voiceGetRate(), voiceSetRate(), voiceGetPitch(), voiceSetPitch(), voiceGetVolume(), voiceSetVolume(), voiceState(), voiceWordPos()

voiceState()

Utilisation

```
-- Syntaxe Lingo
voiceState()

// Syntaxe JavaScript
voiceState(); // pas encore documenté
```

Description

Fonction ; renvoie l'état actuel de la voix sous forme d'un symbole. Les valeurs possibles sont #playing, #paused et #stopped.

Paramètres

Aucun.

Exemple

Les instructions suivantes vérifient si le logiciel de conversion de texte en voix est actuellement actif et définissent la voix sur 1 dans le cas contraire :

```
-- Syntaxe Lingo
if voiceState() <> #playing then
  voiceSet(1)
end if

// Syntaxe JavaScript
if (voiceState() != symbol("playing")) {
  voiceSet(1);
}
```

Voir aussi

voiceSpeak(), voicePause(), voiceResume(), voiceStop(), voiceGetRate(), voiceSetRate(), voiceGetPitch(), voiceSetPitch(), voiceGetVolume(), voiceSetVolume(), voiceWordPos(), voiceSpeak()

voiceStop()

Utilisation

```
-- Syntaxe Lingo
voiceStop()

// Syntaxe JavaScript
voiceStop(); // pas encore documenté
```

Description

Commande ; arrête la sortie vocale de la conversion de texte en voix et vide la mémoire tampon de cette fonction. La commande renvoie la valeur 1 en cas de réussite, et 0 dans le cas contraire.

Paramètres

Aucun.

Exemple

Les instructions suivantes arrêtent la fonction vocale lorsque la tête de lecture se déplace sur l'image suivante dans le scénario :

```
-- Syntaxe Lingo
on exitFrame
  voiceStop()
end exitFrame

// Syntaxe JavaScript
function exitFrame() {
  voiceStop();
}
```

Voir aussi

voiceSpeak(), voicePause(), voiceResume(), voiceGetRate(), voiceSetRate(), voiceGetPitch(), voiceSetPitch(), voiceGetVolume(), voiceSetVolume(), voiceState(), voiceWordPos(), voiceSpeak()

voiceWordPos()

Utilisation

```
-- Syntaxe Lingo
voiceWordPos()

// Syntaxe JavaScript
voiceWordPos();
```

Description

Fonction ; renvoie un nombre entier indiquant la position du mot actuellement en cours de lecture au sein de la chaîne qui le contient. Par exemple, si un acteur contenant 15 mots est lu et que c'est le cinquième mot de l'acteur qui est en cours de lecture lorsque cette fonction est activée, la valeur renvoyée est 5.

Paramètres

Aucun.

Exemple

Les instructions suivantes entraînent la lecture de la phrase « Bonjour, comment allez-vous ? » et affichent la position du mot courant dans la fenêtre Messages. La fonction `voiceWordPos()` étant appelée immédiatement après l'utilisation de la commande `voiceSpeak()`, la valeur renvoyée sera 1.

```
-- Syntaxe Lingo
voiceSpeak("Bonjour, comment allez-vous ?")
put voiceWordPos()
-- 1

// Syntaxe JavaScript
voiceSpeak("Bonjour, comment allez-vous ?")
put(voiceWordPos());
// 1
```

Voir aussi

`voiceSpeak()`, `voicePause()`, `voiceResume()`, `voiceStop()`, `voiceGetRate()`, `voiceSetRate()`, `voiceGetPitch()`, `voiceSetPitch()`, `voiceGetVolume()`, `voiceSetVolume()`, `voiceState()`, `voiceSpeak()`

voidP()

Utilisation

```
-- Syntaxe Lingo
voidP(nomDeVariable)

// Syntaxe JavaScript
nomDeVariable == null
```

Description

Fonction ; détermine si une variable spécifiée a une valeur. Si la variable n'a aucune valeur initiale ou est `VOID`, cette fonction renvoie `TRUE`. Si la variable a une valeur différente de `VOID`, cette fonction renvoie `FALSE`.

Paramètres

`nomDeVariable` Requis. Spécifie la variable à tester.

Exemple

L'instruction suivante vérifie si la variable `réponse` a une valeur initiale :

```
-- Syntaxe Lingo
put voidP(réponse)

// Syntaxe JavaScript
put(réponse == null));
```

Voir aussi

`ilk()`, `VOID`

window()

Utilisation

```
-- Syntaxe Lingo
window(chaîneNomDeFenêtre)

// Syntaxe JavaScript
window(chaîneNomDeFenêtre);
```

Description

Fonction du niveau supérieur ; renvoie une référence à une fenêtre spécifiée.

La fenêtre spécifiée doit contenir une animation Director.

Les fenêtres contenant les animations sont pratiques pour créer des palettes flottantes, des tableaux de commande indépendants et des fenêtres de formes différentes. L'utilisation de fenêtres contenant les animations vous permet d'ouvrir plusieurs animations simultanément et de les faire dialoguer.

Paramètres

chaîneNomDeFenêtre Requis. Chaîne qui spécifie le nom de la fenêtre à référencer.

Exemple

Cette instruction affecte la variable `maFenêtre` à la fenêtre Collections :

```
-- Syntaxe Lingo
maFenêtre = window("Collections")

// Syntaxe JavaScript
var maFenêtre = window("Collections");
```

Voir aussi

Fenêtre

windowPresent()

Utilisation

```
-- Syntaxe Lingo
_player.windowPresent(chaîneNomDeFenêtre)

// Syntaxe JavaScript
_player.windowPresent(chaîneNomDeFenêtre);
```

Description

Méthode de lecteur ; indique si l'objet spécifié par *chaîneNomDeFenêtre* est exécuté sous la forme d'une animation dans une fenêtre (`TRUE`) ou non (`FALSE`).

Si une fenêtre a été ouverte, `windowPresent()` conserve la valeur `TRUE` pour cette fenêtre jusqu'à ce qu'elle soit supprimée de la propriété `windowList`.

L'argument *chaîneNomDeFenêtre* doit correspondre au nom de la fenêtre tel qu'il apparaît dans la propriété `windowList`.

Paramètres

chaîneNomDeFenêtre Requis. Chaîne qui spécifie le nom de la fenêtre à obtenir.

Exemple

L'instruction suivante vérifie si l'objet `maFenêtre` est une animation dans une fenêtre (MIAW) et affiche le résultat dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(_player.windowPresent(maFenêtre))

// Syntaxe JavaScript
put(_player.windowPresent(maFenêtre));
```

Voir aussi

`Lecteur`, `windowList`

worldSpaceToSpriteSpace

Utilisation

```
-- Syntaxe Lingo
member(quelActeur).camera(quelleCaméra).worldSpaceToSpriteSpace(vecteur)

// Syntaxe JavaScript
member(quelActeur).camera(quelleCaméra).worldSpaceToSpriteSpace(vecteur);
```

Description

Commande 3D ; renvoie le point du cadre de la caméra où une position donnée, relative à l'univers, apparaîtrait. La position renvoyée par cette commande est relative au coin supérieur gauche du cadre de la caméra.

Si la position spécifiée est en-dehors du champ de la caméra, cette commande renvoie `void`.

Paramètres

`vecteur` Requis. Spécifie la position relative à l'univers qui apparaîtraît.

Exemple

L'instruction suivante indique que l'origine de l'univers, spécifiée par vector(0, 0, 0), apparaît au point (250,281) du cadre de la caméra.

```
-- Syntaxe Lingo
put sprite(5).camera.worldSpaceToSpriteSpace(vector(0, 0, 0))
-- point(250, 281)

// Syntaxe JavaScript
put sprite(5).camera.worldSpaceToSpriteSpace(vector(0,0,0)));
```

Voir aussi

`spriteSpaceToWorldSpace`, `rect (caméra)`

writeChar()

Utilisation

```
-- Syntaxe Lingo
réfObjFileio.writeChar(chaîneCaractère)

// Syntaxe JavaScript
réfObjFileio.writeChar(chaîneCaractère)
```

Description

Méthode de Fileio ; écrit un seul caractère ASCII spécifié dans un fichier.

Vous devez d'abord ouvrir un fichier en appelant `openFile()` avant d'utiliser `writeChar()` pour lire un caractère.

Paramètres

chaîneCaractère Requis. Spécifie le caractère ASCII à écrire dans le fichier.

Voir aussi

Fileio

writeReturn()

Utilisation

```
-- Syntaxe Lingo
réfObjFileio.writeReturn()

// Syntaxe JavaScript
réfObjFileio.writeReturn();
```

Description

Méthode de Fileio ; insère un retour à la ligne dans un fichier.

Paramètres

Aucun.

Voir aussi

Fileio

writeString()

Utilisation

```
-- Syntaxe Lingo
réfObjFileio.writeString(chaîne)

// Syntaxe JavaScript
réfObjFileio.writeString(chaîne)
```

Description

Méthode de Fileio ; écrit une chaîne qui se termine par une valeur nulle dans un fichier.

Paramètres

chaîne Requis. Chaîne à écrire dans un fichier.

Voir aussi

Fileio

xtra()

Utilisation

```
-- Syntaxe Lingo
xtra(nomOuNumXtra)

// Syntaxe JavaScript
xtra(nomOuNumXtra);
```

Description

Fonction du niveau supérieur ; renvoie une instance d'un Xtra spécifié.

Une référence à un objet vide est renvoyée si l'Xtra spécifié est introuvable.

Pour un exemple d'`xtra` dans une animation, consultez l'animation Read and Write Text du dossier Learning/Lingo, lui-même inclus dans le dossier de Director.

Paramètres

nomOuNumXtra Requis. Chaîne qui spécifie le nom de l'Xtra à renvoyer ou nombre entier qui spécifie sa position d'index. Les noms ne font pas la distinction entre les majuscules et les minuscules.

Exemple

L'instruction suivante affecte la variable `monNetLingo` à l'Xtra NetLingo :

```
-- Syntaxe Lingo
monNetLingo = xtra("netlingo")

// Syntaxe JavaScript
var monNetLingo = xtra("netlingo");
```

zoomBox

Utilisation

```
-- Syntaxe Lingo
zoomBox imageObjetSource, imageObjetDestination {,délaiEnBattements}

// Syntaxe JavaScript
zoomBox imageObjetSource, imageObjetDestination {,délaiEnBattements}); // pas
  encore documenté
```

Description

Commande ; crée un effet de zoom semblable à celui généré par le Finder (Macintosh) lors de l'ouverture d'une fenêtre. L'effet de zoom va du rectangle de délimitation d'une image-objet de début donnée jusqu'au rectangle de délimitation d'une image-objet de fin donnée. La commande `zoomBox` utilise la logique suivante lors de l'exécution :

1 Rechercher *`imageObjetDestination`* dans l'image courante : sinon,

2 Rechercher *`imageObjetDestination`* dans l'image suivante.

Notez que la commande `zoomBox` ne fonctionne pas si l'*`imageObjetDestination`* se trouve dans la même piste que l'*`imageObjetSource`*.

Paramètres

`imageObjetSource` Requis. Spécifie l'image-objet de début.

`imageObjetDestination` Requis. Spécifie l'image-objet de fin.

`délaiEnBattements` Facultatif. Spécifie le délai mesuré en battements entre chaque mouvement des rectangles de zoom. Si *`délaiEnBattements`* n'est pas spécifié, la valeur par défaut est 1.

Exemple

L'instruction suivante crée un effet de zoom entre les images-objets 7 et 3 :

```
-- Syntaxe Lingo
zoomBox 7, 3

// Syntaxe JavaScript
zoomBox(7, 3); // pas encore documenté
```

CHAPITRE 13
Opérateurs

Cette section propose une liste alphabétique de tous les opérateurs disponibles dans Macromedia Director MX 2004.

La plupart de ces opérateurs ne s'appliquent qu'à Lingo. La syntaxe JavaScript contient des opérateurs qui sont similaires ou identiques aux opérateurs Lingo énumérés ici ; nous vous fournirons donc des explications sur l'usage de la Syntaxe JavaScript ainsi que des exemples pour vous aider à associer les fonctionnalités des opérateurs Lingo à leurs équivalents les plus proches dans la syntaxe JavaScript. Pour plus d'informations sur les opérateurs de la syntaxe JavaScript, consultez le Chapitre 2, *Principes de base du scripting dans Director*, page 9.

(symbole)

Utilisation

```
-- Syntaxe Lingo
#nomDeSymbole

// Syntaxe JavaScript
symbol("nomDeSymbole");
```

Description

Opérateur de symbole ; définit un symbole, une unité autonome pouvant représenter une condition ou un indicateur. La valeur *nomDeSymbole* commence par un caractère alphabétique et peut être suivie de plusieurs caractères alphabétiques ou numériques.

Un symbole vous permet d'effectuer les opérations suivantes :

- Affecter une valeur à une variable.
- Comparer des chaînes, des entiers, des rectangles et des points.
- Passer un paramètre à un gestionnaire ou à une méthode.
- Renvoyer une valeur à partir d'un gestionnaire ou d'une méthode.

Les symboles prennent moins de place que les chaînes et sont plus facilement manipulables, mais ne sont pas formés de caractères individuels de la même manière qu'une chaîne. Vous pouvez convertir un symbole en chaîne pour l'afficher, à l'aide de la fonction `string`.

Les points suivants concernant la syntaxe des symboles sont très importants :

- La différence entre les majuscules et les minuscules n'a pas d'importance dans les symboles.
- Les symboles ne peuvent pas commencer pas un chiffre.
- Vous ne pouvez pas utiliser d'espaces, mais vous pouvez employer des caractères de soulignement pour les simuler.
- Les symboles utilisent les 128 caractères ASCII. Les lettres portant des marques diacritiques ou d'accent sont traitées en fonction de leur lettre de base.
- Vous ne pouvez pas utiliser de points dans les symboles.

Tous les symboles, variables globales et noms de paramètres transmis aux variables globales sont conservés dans une table commune.

Exemple

Cette instruction affecte le symbole `#Lecture` à la variable nommée état :

```
-- Syntaxe Lingo
état = #Lecture

// Syntaxe JavaScript
var état = symbole("Lecture");
```

Voir aussi

`ilk()`, `string()`, `symbol()`, `symbolP()`

. (opérateur point)

Utilisation

```
-- Syntaxe Lingo
référenceDobjet.propriétéDobjet
expressionTest.propriétéDobjet
objet.commandeOuFonction()

// Syntaxe JavaScript
RéférenceDobjet.propriétéDobjet;
expressionDetexte.propriétéDobjet;
objet.commandeOuFonction();
```

Description

Opérateur ; utilisé pour tester ou définir les propriétés des objets, émettre une commande ou exécuter une fonction de l'objet. L'objet peut être un acteur, une image-objet, une liste de propriétés, un objet enfant d'un script parent ou un comportement.

Exemple

Cette instruction indique l'acteur courant contenu dans l'image-objet de la piste 10 :

```
-- Syntaxe Lingo
put(sprite(10.member)

// Syntaxe JavaScript
put(sprite(10).member);
```

Pour utiliser une autre syntaxe et appeler une fonction, utilisez la forme :

```
-- Syntaxe Lingo
monObjetCouleur = color(124, 22, 233)
put(monObjetCouleur.ilk())
-- #color

// Syntaxe JavaScript
var monObjetCouleur = color(124, 22, 233);
put(monObjetCouleur.ilk());
// #color
```

- (signe moins)

Utilisation

```
-- Syntaxe Lingo
(Négation) : -expression
(Soustraction) : expression1 - expression2

// Syntaxe JavaScript
(Négation) : -expression
(Soustraction) : expression1 - expression2
```

Description

Opérateur mathématique ; lorsqu'il est utilisé pour la négation, - (minus) inverse le signe de la valeur de *expression*; lorsqu'il est utilisé pour la soustration, - (minus) effectue une soustraction arithmétique sur deux expressions numériques, soustrayant *expression2* de *expression1*.

Lorsqu'il est utilisé pour la négation, - (minus) est un opérateur arithmétique ayant un niveau de priorité de 5.

Lorsqu'il est utilisé pour la soustraction, les deux expressions sont des nombres entiers, la différence étant également un nombre entier. Si l'une ou les deux expressions sont des nombres à virgule flottante, la différence est un nombre à virgule-flottante. L'opérateur - (minus) est un opérateur arithmétique ayant un niveau de priorité de 3.

Exemple

(Négation) : L'instruction suivante inverse le signe de l'expression 2 + 3 :

```
-- Syntaxe Lingo
put(-(2 + 3))

// Syntaxe JavaScript
put(-(2 + 3));
```

Le résultat est -5.

(Soustraction) : L'instruction suivante soustrait le nombre entier 2 de 5 et affiche le résultat dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(5 - 2)

// Syntaxe JavaScript
put(5 -2);
```

Le résultat est 3, qui est un nombre entier.

(Soustraction) : Cette instruction soustrait le nombre à virgule flottante 1,5 de 3,25 et affiche le résultat dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(3.25 -1.5)

// Syntaxe JavaScript
put(3.25 -1.5);
```

Le résultat est 1,75, qui est un nombre à virgule flottante.

-- (séparateur de commentaires)

Utilisation

```
-- Syntaxe Lingo
-- commentaire

// Syntaxe JavaScript
// commentaire
```

Description

Séparateur de commentaires ; indique le début d'un commentaire dans un script. Dans n'importe quelle ligne de code, ce qui se trouve entre le symbole de commentaire (double trait d'union) et le caractère de retour chariot est interprété comme un commentaire, et non comme une instruction Lingo.

Exemple

Le gestionnaire suivant utilise un double trait d'union pour transformer les deuxième, quatrième et sixième lignes en commentaire :

```
-- Syntaxe Lingo
on razCouleurs
  -- Ce gestionnaire réinitialise les couleurs de l'image-objet.
  sprite(1).forecolor = 35
  -- rouge vif
  sprite(1).backcolor = 36
  -- bleu clair
end

// Syntaxe JavaScript
function razCouleurs() {
  // Ce gestionnaire réinitialise les couleurs de l'image-objet.
  sprite(1).forecolor = 35;
  // rouge vif
  sprite(1).backcolor = 36;
  // bleu clair
}
```

&, + (opérateur de concaténation)

Utilisation

```
-- Syntaxe Lingo
expression1 & expression2

// Syntaxe JavaScript
expression1 + expression2
```

Description

Opérateur de chaîne ; concatène les chaînes de deux expressions. Si *expression1* ou *expression2* est un nombre, elle est d'abord convertie en chaîne. L'expression qui en résulte est une chaîne.

Cet opérateur de chaîne a un niveau de priorité de 2.

Lingo vous permet d'utiliser certaines commandes et fonctions ne prenant qu'un seul argument sans que ce dernier soit encadré de parenthèses. Lorsque l'argument comprend un opérateur, Lingo n'interprète que le premier argument comme partie de la fonction, ce qui risque de provoquer une confusion.

Evitez ce problème en encadrant de parenthèses l'expression comprenant un opérateur. Les parenthèses suppriment toute confusion en modifiant la priorité avec laquelle Lingo traite l'opérateur, l'entraînant à traiter les deux parties de l'argument comme un argument complet.

Exemple

L'instruction suivante concatène les chaînes « abra » et « cadabra » et affiche la chaîne résultante dans la fenêtre Messages :

```
-- Syntaxe Lingo
put("abra" & "cadabra")

// Syntaxe JavaScript
put("abra " + "cadabra");
```

Le résultat est la chaîne « abracadabra ».

L'instruction suivante concatène le contenu de la variable `prix` et la chaîne « euros ». La chaîne concaténée est alors affectée à l'acteur champ Prix :

```
-- Syntaxe Lingo
member("Prix").text = prix & "euros"

// Syntaxe JavaScript
member("Prix").text = prix + "euros";
```

&&, + (opérateur de concaténation)

Utilisation

```
-- Syntaxe Lingo
expression1 && expression2

// Syntaxe JavaScript
expression1 + expression2
```

Description

Opérateur de chaîne ; concatène deux expressions et insère une espace entre les expressions chaînes d'origine. Si *expression1* ou *expression2* est un nombre, elle est d'abord convertie en chaîne. L'expression qui en résulte est une chaîne.

Cet opérateur de chaîne a un niveau de priorité de 2.

Exemple

L'instruction suivante concatène les chaînes « abra » et « cadabra » et insère une espace entre les deux chaînes :

```
-- Syntaxe Lingo
put("abra" && "cadabra")

// Syntaxe JavaScript
put("abra " + "cadabra");
```

Le résultat est la chaîne « abra cadabra ».

L'instruction suivante concatène les chaînes « Date : » et la date d'aujourd'hui au format long et insère une espace entre les deux chaînes :

```
-- Syntaxe Lingo
put("Date :" && date())

// Syntaxe JavaScript
put("Date : " + Date());
```

() (parenthèses)

Utilisation

```
-- Syntaxe Lingo
(expression)

// Syntaxe JavaScript
(expression)
```

Description

Opérateur de regroupement ; effectue une opération de regroupement pour contrôler l'ordre d'exécution des opérateurs d'une expression. Cet opérateur permet de contrôler l'ordre dans lequel les opérateurs sont exécutés dans cette expression et supplante l'ordre de priorité automatique, de sorte que l'expression entre parenthèses soit évaluée en premier. Lorsque des parenthèses sont imbriquées, le contenu des parenthèses intérieures est évalué avant celui des parenthèses extérieures.

Cet opérateur de regroupement a un niveau de priorité de 5.

Sachez que Lingo vous permet d'utiliser certaines commandes et fonctions ne prenant qu'un seul argument sans que ce dernier soit encadré de parenthèses. Lorsque l'argument comprend un opérateur, Lingo n'interprète que le premier argument comme partie de la fonction, ce qui risque de provoquer une confusion.

Par exemple, la commande `open window` permet à un argument de spécifier une fenêtre à ouvrir. Si vous utilisez l'opérateur `&` pour définir un nom de fichier et un chemin, Director n'interprète que la chaîne précédant l'opérateur `&` comme nom de fichier. Par exemple, Lingo interprète l'instruction `open window the applicationPath & "Animation"` comme `(open window the applicationPath) & ("Animation")`. Evitez ce problème en encadrant de parenthèses l'expression comprenant un opérateur, comme suit :

```
-- Syntaxe Lingo
open window (the applicationPath & "Animation")

// Syntaxe JavaScript
window(the applicationPath + "Animation").open();
```

Exemple

Les instructions suivantes utilisent l'opérateur de regroupement pour modifier l'ordre dans lequel les différentes opérations se produisent (le résultat apparaît en dessous de chaque instruction) :

```
-- Syntaxe Lingo
put((2 + 3) * (4 + 5))
-- 45
put(2 + (3 * (4 + 5)))
-- 29
put(2 + 3 * 4 + 5)
-- 19

// Syntaxe JavaScript
put((2 + 3) * (4 + 5));
// 45
put(2 + (3 * (4 + 5)));
// 29
put(2 + 3 * 4 + 5);
// 19
```

* (multiplication)

Utilisation

```
-- Syntaxe Lingo
expression1 * expression2

// Syntaxe JavaScript
expression1 * expression2
```

Description

Opérateur mathématique ; effectue une multiplication arithmétique de deux expressions numériques. Si les deux expressions sont des nombres entiers, le produit est également un nombre entier. Si l'une ou les deux expressions sont des nombres à virgule flottante, le produit est un nombre à-virgule flottante.

Cet opérateur arithmétique a un niveau de priorité de 4.

Exemple

L'instruction suivante multiplie le nombre entier 2 par 3 et affiche le résultat dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(2 * 3)

// Syntaxe JavaScript
put(2 * 3);
```

Le résultat est 6, qui est un nombre entier.

L'instruction suivante multiplie le nombre à virgule flottante 2,0 par 3,1414 et affiche le résultat dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(2.0 * 3.1416)

// Syntaxe JavaScript
put(2.0 * 3.1416);
```

Le résultat est 6.2832, qui est un nombre à virgule flottante.

+ (addition)

Utilisation

```
-- Syntaxe Lingo
expression1 + expression2

// Syntaxe JavaScript
expression1 + expression2
```

Description

Opérateur mathématique ; effectue une somme arithmétique de deux expressions numériques. Si ces deux expressions sont des nombres entiers, la somme est également un nombre entier. Si l'une ou les deux expressions sont des nombres à virgule flottante, la somme est un nombre à virgule flottante.

Cet opérateur arithmétique a un niveau de priorité de 4.

Exemple

L'instruction suivante additionne les nombres entiers 2 et 3, puis affiche le résultat (le nombre entier 5) dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(2 +3)

// Syntaxe JavaScript
put(2 +3);
```

L'instruction suivante additionne les nombres à virgule flottante 2,5 et 3,25, puis affiche le résultat (le nombre à virgule flottante 5,7500) dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(2.5 +3.25)

// Syntaxe JavaScript
put(2.5 + 3.25);
```

+ (addition) (3D)

Utilisation

```
-- Syntaxe Lingo
vecteur1 + vecteur2
vecteur + scalaire

// Syntaxe JavaScript
vecteur1 + vecteur2
vecteur + scalaire
```

Description

Opérateur 3D de vecteur ; ajoute les composants de deux vecteurs, ou la valeur scalaire à chaque composant du vecteur, et renvoie un nouveau vecteur.

vecteur1 + *vecteur2* ajoute les composants de *vecteur1* aux composants correspondants de *vecteur2* et renvoie un nouveau vecteur.

vecteur + *scalaire* ajoute la valeur scalaire à chaque composant du vecteur et renvoie un nouveau vecteur.

- (soustraction)

Utilisation

```
-- Syntaxe Lingo
vecteur1 - vecteur2
vecteur - scalaire

// Syntaxe JavaScript
vecteur1 - vecteur2
vecteur - scalaire
```

Description

Opérateur 3D de vecteur ; soustrait les composants de *vecteur2* des composants correspondants de *vecteur1* ou soustrait la valeur scalaire de chacun des composants et renvoie un nouveau vecteur.

vecteur1 - *vecteur2* soustrait les valeurs de *vecteur2* des composants correspondants de *vecteur1* et renvoie un nouveau vecteur.

vecteur - *scalaire* soustrait la valeur scalaire de chaque composant du vecteur et renvoie un nouveau vecteur.

* (multiplication)

Utilisation

```
-- Syntaxe Lingo
vecteur1 * vecteur2
vecteur * scalaire
transformation * vecteur

// Syntaxe JavaScript
vecteur1 * vecteur2
vecteur * scalaire
transformation * vecteur
```

Description

Opérateur 3D de vecteur ; multiplie les composants de *vecteur1* par les composants correspondants de *vecteur2* et renvoie le produit ou multiplie chaque composant du vecteur par la valeur scalaire et renvoie un nouveau vecteur.

vecteur1 * *vecteur2* renvoie le produit de deux vecteurs, qui n'est pas un nouveau vecteur. Cette opération est équivalente à *vecteur1*.dotproduct.*vecteur2*.

vecteur * *scalaire* multiplie chaque composant du vecteur par la valeur scalaire et renvoie un nouveau vecteur.

transformation * *vecteur* multiplie la *transformation* par le *vecteur* et renvoie un nouveau vecteur. Le nouveau vecteur est le résultat de l'application des modifications de position et de rotation définies par *transformation* à *vecteur*. Vous remarquerez que *vecteur* * *transformation* n'est pas pris en charge.

Voir aussi

dotProduct()

/ (division)

Utilisation

```
-- Syntaxe Lingo
expression1 / expression2

// Syntaxe JavaScript
expression1 / expression2
```

Description

Opérateur mathématique ; effectue une division arithmétique de deux expressions numériques, divisant *expression1* par *expression2*. Si ces deux expressions sont des nombres entiers, le quotient est également un nombre entier. Si l'une ou les deux expressions sont des nombres à virgule flottante, le quotient est un nombre à virgule flottante.

Cet opérateur arithmétique a un niveau de priorité de 4.

Exemple

L'instruction suivante divise le nombre entier 22 par 7, puis affiche le résultat dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(22 / 7)

// Syntaxe JavaScript
put(22 / 7);
```

Le résultat est 3. Puisque les deux nombres de la division sont des nombres entiers, Lingo arrondit le résultat au nombre entier le plus proche.

L'instruction suivante divise le nombre à virgule flottante 22,0 par 7,0, puis affiche le résultat dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(22.0 / 7.0)

// Syntaxe JavaScript
put(22.0 / 7.0);
```

Le résultat est 3.1429, qui est un nombre à virgule flottante.

/ (division) (3D)

Utilisation

```
-- Syntaxe Lingo
vecteur / scalaire

// Syntaxe JavaScript
vecteur / scalaire
```

Description

Opérateur 3D de vecteur ; divise chaque composant du vecteur par la valeur scalaire et renvoie un nouveau vecteur.

< (inférieur à)

Utilisation

```
-- Syntaxe Lingo
expression1 < expression2

// Syntaxe JavaScript
expression1 < expression2
```

Description

Opérateur de comparaison ; compare deux expressions et détermine si *expression1* est inférieure à *expression2* (TRUE) ou si *expression1* est supérieure ou égale à *expression2* (FALSE).

Cet opérateur peut comparer des chaînes, des nombres entiers, des nombres à virgule flottante, des rects et des points. Sachez que les comparaisons effectuées sur les rects ou les points sont traitées comme s'il s'agissait de listes, avec chaque élément de la première liste comparé à l'élément correspondant de la seconde liste.

Il s'agit d'un opérateur de comparaison avec niveau de priorité de 1.

<= (inférieur ou égal à)

Utilisation

```
-- Syntaxe Lingo
expression1 <= expression2

// Syntaxe JavaScript
expression1 <= expression2
```

Description

Opérateur de comparaison ; compare deux expressions et détermine si *expression1* est inférieure ou égale à *expression2* (TRUE)) ou si *expression1* est supérieure à *expression2* (FALSE).

Cet opérateur peut comparer des chaînes, des nombres entiers, des nombres à virgule flottante, des rects et des points. Sachez que les comparaisons effectuées sur les rects ou les points sont traitées comme s'il s'agissait de listes, avec chaque élément de la première liste comparé à l'élément correspondant de la seconde liste.

Il s'agit d'un opérateur de comparaison avec niveau de priorité de 1.

<> (différent de)

Utilisation

```
-- Syntaxe Lingo
expression1 <> expression2

// Syntaxe JavaScript
expression1 != expression2
```

Description

Opérateur de comparaison ; compare deux expressions, symboles ou opérateurs et détermine si *expression1* n'est pas égale à *expression2* (TRUE) ou si *expression1* est égale à *expression2* (FALSE).

Cet opérateur peut comparer des chaînes, des nombres entiers, des nombres à virgule flottante, des rects et des points. Sachez que les comparaisons effectuées sur les rects ou les points sont traitées comme s'il s'agissait de listes, avec chaque élément de la première liste comparé à l'élément correspondant de la seconde liste.

Il s'agit d'un opérateur de comparaison avec niveau de priorité de 1.

= (signal égal à)

Utilisation

```
-- Syntaxe Lingo
expression1 = expression2

// Syntaxe JavaScript
expression1 = expression2
```

Description

Opérateur de comparaison ; compare deux expressions, symboles ou objets et détermine si *expression1* est égale à *expression2* (TRUE) ou si *expression1* n'est pas égale à *expression2* (FALSE).

Cet opérateur peut comparer des chaînes, des nombres entiers, des nombres à virgule flottante, des rects, des listes et des points.

Les listes sont comparées sur la base du nombre d'éléments qu'elles contiennent. La liste contenant le plus d'éléments est considérée comme plus longue que la liste contenant moins d'éléments.

Il s'agit d'un opérateur de comparaison avec niveau de priorité de 1.

> (supérieur à)

Utilisation

```
-- Syntaxe Lingo
expression1 > expression2

// Syntaxe JavaScript
expression1 > expression2
```

Description

Opérateur de comparaison ; compare deux expressions et détermine si *expression1* est supérieure à *expression2* (TRUE) ou si *expression1* est inférieure ou égale à *expression2* (FALSE).

Cet opérateur peut comparer des chaînes, des nombres entiers, des nombres à virgule flottante, des rects et des points. Sachez que les comparaisons effectuées sur les rects ou les points sont traitées comme s'il s'agissait de listes, avec chaque élément de la première liste comparé à l'élément correspondant de la seconde liste.

Il s'agit d'un opérateur de comparaison avec niveau de priorité de 1.

> = (supérieur ou égal à)

Utilisation

```
-- Syntaxe Lingo
expression1 >= expression2

// Syntaxe JavaScript
expression1 >= expression2
```

Description

Opérateur de comparaison ; compare deux expressions et détermine si *expression1* est supérieure ou égale à *expression2* (TRUE) ou si *expression1* est inférieure à *expression2* (FALSE).

Cet opérateur peut comparer des chaînes, des nombres entiers, des nombres à virgule flottante, des rects et des points. Sachez que les comparaisons effectuées sur les rectangles ou les points sont traitées comme s'il s'agissait de listes, avec chaque élément de la première liste comparé à l'élément correspondant de la seconde liste.

Il s'agit d'un opérateur de comparaison avec niveau de priorité de 1.

[ ] (crochets d'accès)

Utilisation

```
-- Syntaxe Lingo
expressionTexte[numéroDeSousChaîneAdressée]
expressionTexte[premièreSousChaîne..dernièreSousChaîne]
```

Description

Opérateur ; permet de désigner une expression de sous-chaîne par un nombre. Cette fonction est utile pour trouver la *n*ième sous-chaîne de l'expression. Cette expression peut être un mot, une ligne, un caractère, un paragraphe ou une autre sous-chaîne d'un acteur texte.

Exemple

L'instruction suivante renvoie le premier mot de la troisième ligne de l'acteur texte Prénoms :

```
-- Syntaxe Lingo
put(member("Prénoms").text.line[3].word[1])

// Syntaxe JavaScript
put(member("Prénoms").getPropRef("ligne", 1).getProp("mot", 1));
```

[ ] (liste)

Utilisation

```
[entrée1, entrée2, entrée3, ...]
```

Description

Opérateur de liste ; spécifie que les entrées contenues dans ces crochets appartiennent à l'un des quatre types de listes suivants :

- Listes linéaires non triées
- Listes linéaires triées
- Listes de propriétés non triées
- Listes de propriétés triées

Chaque entrée d'une liste linéaire est une valeur unique à laquelle aucune autre propriété n'est associée. Chaque entrée d'une liste de propriétés est constituée d'une propriété et d'une valeur. La propriété précède la valeur et est séparée de celle-ci par le signe deux points. Vous ne pouvez pas stocker une propriété dans une liste linéaire. Lors de l'utilisation de chaînes comme entrées d'une liste, il convient de placer les chaînes entre guillemets.

Par exemple, [6, 3, 8] est une liste linéaire. Les nombres qu'elle contient ne sont associés à aucune propriété. Par contre, [#engrenages:6, #billes:3, #rampes:8] est une liste de propriétés. Chaque nombre de cette liste est associé à une propriété, une pièce de machine dans cet exemple. Cette liste de propriétés peut servir à contrôler le nombre de pièces de chaque sorte se trouvant sur la scène dans une simulation mécanique. Les propriétés peuvent apparaître plusieurs fois dans une liste de propriétés.

Les listes peuvent être triées en ordre alphanumérique. Une liste linéaire triée est classée selon les valeurs qu'elle contient. Une liste de propriétés triée est classée selon les propriétés qu'elle contient. Le tri d'une liste linéaire ou de propriétés s'opère avec la commande de tri appropriée.

- Dans les listes linéaires, les symboles et les chaînes sont sensibles à l'emploi des minuscules ou des majuscules.
- Dans les listes de propriétés, les symboles, contrairement aux chaînes, ne sont pas sensibles à la casse.

Une liste linéaire ou une liste de propriétés peut ne contenir aucune valeur. Une liste vide consiste en deux crochets ([ ]). Pour créer ou supprimer une liste linéaire, affectez-lui la valeur `[ ]`. Pour créer ou supprimer une liste de propriétés, affectez-lui la valeur `[:]`.

Vous pouvez modifier, tester ou lire les éléments contenus dans les listes.

Lingo traite toute instance de liste comme une référence à cette liste. Autrement dit, chaque instance représente les mêmes éléments de données et sa modification modifie l'original. Utilisez la commande `duplicate` pour créer des copies de listes.

Les listes sont automatiquement effacées lorsque aucune variable n'y fait plus référence. Si une liste est référencée dans une variable globale, elle existe d'animation en animation.

Vous pouvez initialiser une liste dans le gestionnaire `on prepareMovie` ou rédiger la liste comme acteur champ, l'affecter à une variable, puis la contrôler par la variable.

Tous les claviers ne sont pas forcément équipés de touches de crochets. Le cas échéant, utilisez la fonction `list` pour créer une liste linéaire.

Pour une liste de propriétés, créez les éléments de liste sous forme de chaîne avant de les convertir en une liste utile.

```
set maChaîneListe = numToChar(91) & ":" & numToChar(93)
put maChaîneListe
-- "[:]"
maListe = maChaîneListe.value
put maListe
-- [:]
put maListe.listP
-- 1
maListe[#nom] = "Pierre"
put maListe
-- [#name: "Pierre"]
```

Exemple

L'instruction suivante définit une liste en rendant la variable `machine` égale à la liste :

```
-- Syntaxe Lingo
machine = [#engrenages:6, #billes:3, #rampes:8]

// Syntaxe JavaScript
var machine = propList("engrenages",6, "billes",3, "rampes",8);
```

Le gestionnaire suivant trie la liste uneListe, puis affiche le résultat dans la fenêtre Messages :

```
-- Syntaxe Lingo
on trierListe uneListe
  uneListe.sort()
  put(uneListe)
end trierListe
```

```
// Syntaxe JavaScript
fonction trierListe(uneListe) {
  uneListe.sort();
  put(uneListe);
}
```

Si l'animation émet l'instruction `trierListe machine`, où `machine` est la liste de l'exemple précédent, le résultat est [#billes:3, #engrenages:6, #rampes:8].

Les instructions suivantes créent une liste linaire vide :

```
-- Syntaxe Lingo
x = [ ]
x = list()

// Syntaxe JavaScript
var x = list();
```

Les instructions suivantes créent une liste de propriétés vide :

```
-- Syntaxe Lingo
x = [:]
x = propList()

// Syntaxe JavaScript
var x = propList();
```

Voir aussi

`add`, `addVertex()`, `append`, `count()`, `deleteAt`, `duplicate() (fonction de liste)`, `findPos`, `findPosNear`, `getProp()`, `getAt`, `getLast()`, `getPos()`, `ilk()`, `list()`, `max()`, `min`, `setAt`, `setaProp`, `sort`

@ (chemin d'accès)

Utilisation

`@référenceDeChemin`

Description

Opérateur de chemin d'accès ; définit le chemin d'accès du dossier de l'animation en cours et offre l'avantage d'être compris par Windows comme par Macintosh.

Identifiez le dossier de l'animation en cours à l'aide du symbole `@` suivi de l'un des séparateurs de chemin suivants :

- `/` (barre oblique)
- `\` (barre oblique inverse)
- `:` (deux-points)

Lorsqu'une animation est interrogée pour en déterminer l'emplacement, la chaîne renvoyée inclut le symbole @.

Veillez à n'utiliser que le symbole @ lorsque vous passez d'une animation Director à une autre ou que vous modifiez la source d'un acteur média lié. Le symbole @ n'a aucun effet lorsque l'Xtra FileIO ou des fonctions autres que celles qui sont disponibles dans Director sont utilisés.

Vous pouvez vous reposer sur ce chemin pour spécifier des dossiers placés en amont ou en aval du dossier de l'animation en cours. N'oubliez pas que la partie @ représente la position de l'animation en cours et pas nécessairement celle de la projection.

- Ajoutez un séparateur de chemin immédiatement à la suite du symbole `@` pour spécifier un dossier en amont d'un niveau dans la hiérarchie.
- Ajoutez les noms de dossiers et fichiers (séparés par `/`, `\` ou `:`) à la suite du nom du dossier courant pour spécifier des dossiers et des fichiers placés en aval dans les dossiers.

Vous pouvez utiliser des chemins relatifs dans Lingo pour indiquer l'emplacement d'un fichier lié dans un dossier différent de celui contenant l'animation.

Exemple

Ces trois expressions sont équivalentes et spécifient le sous-dossier grosDossier qui se trouve dans le dossier de l'animation courante :

```
@/grosDossier
@:grosDossier
@\grosDossier
```

Les expressions suivantes sont équivalentes et spécifient le fichier fichierLié, dans le sous-dossier grosDossier, qui se trouve dans le dossier de l'animation courante :

```
@:grosDossier:fichierLié
@\grosDossier:fichierLié
@/grosDossier:fichierLié
```

L'expression suivante spécifie le fichier fichierLié, qui se trouve un niveau en amont du dossier contenant l'animation courante :

```
@//fichierLié
```

L'expression suivante spécifie le fichier fichierLié, qui se trouve deux niveaux en amont du dossier contenant l'animation courante :

```
@:::fichierLié
```

Les expressions suivantes sont équivalentes et servent à spécifier le fichier fichierLié, qui se trouve dans le dossier autreDossier. Le dossier autreDossier se trouve à un niveau en amont du dossier contenant l'animation courante.

```
@::autreDossier:fichierLié
@\\autreDossier:fichierLié
@//autreDossier:fichierLié
```

Voir aussi

`searchPathList`, `fileName (distribution)`, `fileName (acteur)`, `fileName (fenêtre)`

and

Utilisation

```
-- Syntaxe Lingo
expressionLogique1 et expressionLogique2

// Syntaxe JavaScript
expressionLogique1 && expressionLogique2
```

Description

Opérateur logique ; détermine si *`expressionLogique1`* et *`expressionLogique2`* ont toutes les deux la valeur `TRUE` (1) ou si l'une ou les deux expressions ont la valeur `FALSE`.

Cet opérateur logique a un niveau de priorité de 4.

Exemple

Cette instruction détermine si les deux expressions logiques ont la valeur `TRUE` et affiche le résultat dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(1 < 2 and 2 < 3)

// Syntaxe JavaScript
put((1 < 2) && (2 < 3));
```

Le résultat est 1, équivalent numérique de `TRUE`.

La première expression logique de l'instruction suivante a la valeur `TRUE` ; la deuxième a la valeur `FALSE`. Puisque les deux expressions logiques n'ont pas toutes deux la valeur `TRUE`, l'opérateur logique renvoie le résultat 0, qui est l'équivalent numérique de la valeur `FALSE`.

```
-- Syntaxe Lingo
put(1 < 2 and 2 < 1)
-- 0

// Syntaxe JavaScript
put((1 < 2) && (2 < 1));
// 0
```

Voir aussi

`not`, `or`

contains

Utilisation

```
-- Syntaxe Lingo
expressionChaîne1 contains expressionChaîne2

// Syntaxe JavaScript
expressionChaîne1.indexOf(expressionChaîne2);
```

Description

Opérateur ; compare deux chaînes et détermine si *`expressionChaîne1`* contient *`expressionChaîne2`* (`TRUE`) ou non (`FALSE`).

L'opérateur de comparaison `contains` a un niveau de priorité de 1.

L'opérateur `contains` est utile pour vérifier si l'utilisateur tape un caractère ou une chaîne spécifique de caractères. Vous pouvez également utiliser l'opérateur `contains` pour rechercher des chaînes de caractères spécifiques dans un ou plusieurs champs.

Exemple

L'exemple suivant détermine si le caractère transmis est un chiffre :

```
-- Syntaxe Lingo
on estNombre uneLettre
  chiffres = "1234567890"
  if chiffres contains uneLettre then
    return TRUE
  else
    return FALSE
  end if
end
```

```
// Syntaxe JavaScript
fonction estNombre(uneLettre) {
  var chiffres = "1234567890"
  if (digits.indexOf(uneLettre) >= 0) {
    return TRUE;
  } else {
    return FALSE;
  }
}
```

Remarque : La comparaison de chaînes n'étant pas sensible à la casse, les lettres « a » et « Å » sont traitées de la même manière.

Voir aussi

offset() (fonction de chaîne), starts

mod

Utilisation

```
-- Syntaxe Lingo
expressionEntière1 mod expressionEntière2

// Syntaxe JavaScript
expressionEntière1 % expressionEntière2
```

Description

Opérateur mathématique ; calcule le reste de la division de deux entiers. Dans cette opération, *expressionEntière1* est divisée par *expressionEntière2*.

La valeur de l'expression entière obtenue correspond au reste de la division sous forme d'entier. Ce nombre est toujours accompagné du signe de *expressionEntière1*.

Cet opérateur arithmétique a un niveau de priorité de 4.

Exemple

L'instruction suivante divise 7 par 4, puis affiche le reste dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(7 mod 4)

// Syntaxe JavaScript
put(7 % 4);
```

Le résultat est 3.

Le gestionnaire suivant donne à toutes les images-objets impaires l'effet d'encre `copy` (dont le numéro est 0). Le gestionnaire commence par vérifier si l'image-objet de la variable `monImageObjet` est une image-objet impaire en divisant son numéro par 2, puis vérifie si le reste de la division est 1. Le cas échéant, le gestionnaire définit l'effet d'encre sur `copy`.

```
-- Syntaxe Lingo
on définirLencre
  repeat with monImageObjet = 1 to _movie.lastChannel
    if (monImageObjet mod 2) = 1 then
      sprite(monImageObjet).ink = 0
    else
      sprite(monImageObjet).ink = 8
    end if
  end repeat
end définirLencre

// Syntaxe JavaScript
fonction setInk() {
  for (monImageObjet=1; monImageObjet<=_movie.lastChannel; mySprite++) {
    if ((monImageObjet % 2) == 1) {
      sprite(monImageObjet).ink = 0;
    } else {
      sprite(monImageObjet).ink = 8;
    }
  }
}
```

Le gestionnaire suivant change régulièrement l'acteur d'une image-objet en choisissant un autre acteur parmi un certain nombre de bitmaps :

```
-- Syntaxe Lingo
on exitFrame
  global gCompteur
  -- exemples de valeurs des numéros d'acteurs bitmaps
  lesBitmaps = [2,3,4,5,6,7]
  -- indiquez la piste d'image-objet affectée
  laPiste = 1
  -- cycle dans la liste d'acteurs
  gCompteur = 1 + (gCompteur mod lesBitmaps.count)
  sprite(laPiste).memberNum = lesBitmaps[gCompteur]
  _movie.go(_movie.frame)
end

// Syntaxe JavaScript
function exitFrame() {
  // exemples de valeurs des numéros d'acteurs bitmaps
  lesBitmaps = nouveau tableau(2,3,4,5,6,7);
  // indiquez la piste d'image-objet qui est affectée
  laPiste = 1;
  // cycle dans la liste d'acteurs
  _global.gCompteur = 1 + (_global.gCompteur % lesBitmaps.length);
  sprite(laPiste).memberNum = lesBitmaps[_global.gCompteur];
  _movie.go(_movie.frame);
}
```

not

Utilisation

```
-- Syntaxe Lingo
notexpressionLogique

// Syntaxe JavaScript
! expressionLogique
```

Description

Opérateur ; effectue la négation logique d'une expression logique. Elle revient à donner à une valeur `TRUE` la valeur `FALSE` et à donner à une valeur `FALSE` la valeur `TRUE`. Elle est pratique pour vérifier si une certaine condition connue existe ou non.

Cet opérateur logique a un niveau de priorité de 5.

Exemple

L'instruction suivante détermine si 1 n'est pas inférieur à 2 :

```
-- Syntaxe Lingo
put(not (1 < 2))

// Syntaxe JavaScript
put(!(1 < 2));
```

Puisque 1 est inférieur à 2, le résultat est 0, ce qui indique que l'expression est `FALSE`.

L'instruction suivante détermine si 1 n'est pas supérieur à 2 :

```
-- Syntaxe Lingo
put(not (1 > 2))

// Syntaxe JavaScript
put(!(1 > 2));
```

Puisque 1 n'est pas supérieur à 2, le résultat est 1, ce qui indique que l'expression est `TRUE`.

Le gestionnaire suivant donne à la propriété `the checkMark` de l'article Gras du menu Style l'inverse de sa valeur courante :

```
-- Syntaxe Lingo
on razDeLélémentDeMenu
  menu("Style").menuItem("Gras").checkMark = \
  not (menu("Style").menuItem("Gras").checkMark)
end razDeLélémentDeMenu

// Syntaxe JavaScript
function razDeLélémentDeMenu() {
  menu("Style").menuItem("Gras").checkMark =
  !(menu("Style").menuItem("Gras").checkMark)
}
```

Voir aussi

and, or

or

Utilisation

```
-- Syntaxe Lingo
expressionLogique1 orexpressionLogique2

// Syntaxe JavaScript
expressionLogique1 || expressionLogique2
```

Description

Opérateur ; effectue une opération OR logique sur deux expressions logiques ou plus pour déterminer si une expression est TRUE.

Cet opérateur logique a un niveau de priorité de 4.

Exemple

Cette instruction indique dans la fenêtre Messages si au moins une des expressions 1 < 2 et 1 > 2 est TRUE :

```
-- Syntaxe Lingo
put((1 < 2) or (1 > 2))

// Syntaxe JavaScript
put((1 < 2) || (1 > 2));
```

Puisque la première expression est TRUE, le résultat est 1 (équivalent numérique de TRUE).

L'instruction suivante vérifie si le contenu de l'acteur champ appelé Département est Ariège ou Gironde et, le cas échéant, affiche un message d'alerte :

```
-- Syntaxe Lingo
if member("Département").text = "Ariège" or member("Etat").text = "Gironde"
  then
  _player.alert("Vous vous êtes perdu !")
end if

// Syntaxe JavaScript
if (member("Département").text == "Ariège" || member("Etat").text ==
  "Gironde") {
  _player.alert("Vous vous êtes perdu !");
}
```

Voir aussi

and, not

starts

Utilisation

```
-- Syntaxe Lingo
chaîne1 starts chaîne2

// Syntaxe JavaScript
chaîne1.indexOf(chaîne2) == 0;
```

Description

Opérateur ; permet de déterminer si la *chaîne1* commence par la *chaîne2* (TRUE ou 1) ou non (FALSE ou 0).

La comparaison des chaînes ne tient pas compte des majuscules ou des accents ; *a* et *Å* sont ainsi considérés comme identiques.

Il s'agit d'un opérateur de comparaison avec niveau de priorité de 1.

Exemple

Cette instruction affiche dans la fenêtre Messages si le mot *Macromedia* commence par la chaîne Macro :

```
-- Syntaxe Lingo
put("Macromedia" starts "Macro")

// Syntaxe JavaScript
var chaîne1 = "Macromedia";
put(string1.indexOf("Macro") == 0);
```

Le résultat est 1, équivalent numérique de TRUE.

Voir aussi

contains

CHAPITRE 14
Propriétés

Cette section propose une liste alphabétique de toutes les propriétés offertes par Macromedia Director MX 2004.

_global

Utilisation

```
-- Syntaxe Lingo
_global

// Syntaxe JavaScript
_global;
```

Description

Propriété de haut niveau ; fait référence à l'objet Global, qui enregistre l'ensemble des variables globales. En lecture seule.

Toutes les variables globales sont accessibles à la fois aux syntaxes Lingo et JavaScript.

Exemple

Cette instruction affecte la variable `objGlobal` à la propriété `_global` :

```
-- Syntaxe Lingo
objGlobal = _global

// Syntaxe JavaScript
var objGlobal = _global;
```

Cette instruction utilise la propriété `_global` directement pour supprimer toutes les variables :

```
-- Syntaxe Lingo
_global.clearGlobals()

// Syntaxe JavaScript
_global.clearGlobals();
```

Voir aussi

`Global`

_key

Utilisation

```
-- Syntaxe Lingo
_key

// Syntaxe JavaScript
_key;
```

Description

Propriété de haut niveau ; fait référence à l'objet Key, qui sert à surveiller l'activité du clavier d'un utilisateur. En lecture seule.

Exemple

L'instruction suivante affecte la variable `objKey` à la propriété `_key` :

```
-- Syntaxe Lingo
objKey = _key

// Syntaxe JavaScript
var objKey = _key;
```

Cette instruction utilise la propriété `_key` directement pour accéder à la valeur de la propriété `key` :

```
-- Syntaxe Lingo
theKey = _key.key

// Syntaxe JavaScript
var theKey = _key.key;
```

Voir aussi

Touche

_mouse

Utilisation

```
-- Syntaxe Lingo
_mouse

// Syntaxe JavaScript
_mouse;
```

Description

Propriété de haut niveau ; fait référence à l'objet Mouse, qui permet de contrôler l'activité de la souris d'un utilisateur, y compris ses mouvements et les clics qu'elle effectue. En lecture seule.

Exemple

L'instruction suivante affecte la variable `objMouse` à la propriété `_mouse` :

```
-- Syntaxe Lingo
objMouse = _mouse

// Syntaxe JavaScript
var objMouse = _mouse;
```

Cette instruction utilise la propriété `_mouse` directement pour accéder à la valeur de la propriété `mouseH` :

```
-- Syntaxe Lingo
theMouseH = _mouse.mouseH

// Syntaxe JavaScript
var theMouseH = _mouse.mouseH;
```

Voir aussi

Souris

_movie

Utilisation

```
-- Syntaxe Lingo
_movie

// Syntaxe JavaScript
_movie;
```

Description

Propriété de haut niveau ; fait référence à l'objet Animation, qui représente l'animation en cours de projection dans le lecteur Director et permet d'accéder aux propriétés et aux méthodes disponibles dans le cadre d'une animation. En lecture seule.

Exemple

L'instruction suivante affecte la variable `objMovie` à la propriété `_movie` :

```
-- Syntaxe Lingo
objMovie = _movie

// Syntaxe JavaScript
var objMovie = _movie;
```

Cette instruction utilise la propriété `_movie` directement pour accéder à la valeur de la propriété `displayTemplate` :

```
-- Syntaxe Lingo
theTemplate = _movie.displayTemplate

// Syntaxe JavaScript
var theTemplate = _movie.displayTemplate;
```

Voir aussi

Animation

_player

Utilisation

```
-- Syntaxe Lingo
_player

// Syntaxe JavaScript
_player;
```

Description

Propriété de haut niveau ; fait référence à l'objet Player, qui gère et exécute l'ensemble des animations, y compris les animations au sein d'une fenêtre (MIAWs). En lecture seule.

Exemple

L'instruction suivante affecte la variable `objPlayer` à la propriété `_player` :

```
-- Syntaxe Lingo
objPlayer = _player

// Syntaxe JavaScript
var objPlayer = _player;
```

Cette instruction utilise la propriété `_player` directement pour accéder à la valeur de la propriété `xtraList` :

```
-- Syntaxe Lingo
theXtras = _player.xtraList

// Syntaxe JavaScript
var theXtras = _player.xtraList;
```

Voir aussi

Lecteur

_sound

Utilisation

```
-- Syntaxe Lingo
_sound

// Syntaxe JavaScript
_sound;
```

Description

Propriété de haut niveau ; fait référence à l'objet Son, qui contrôle la lecture audio pour l'ensemble des huit canaux de son disponibles. En lecture seule.

Exemple

L'instruction suivante affecte la variable `objSound` à la propriété `_sound` :

```
-- Syntaxe Lingo
objSound = _sound

// Syntaxe JavaScript
var objSound = _sound;
```

Cette instruction utilise la propriété `_sound` directement pour accéder à la propriété `soundLevel` :

```
-- Syntaxe Lingo
theLevel = _sound.soundLevel

// Syntaxe JavaScript
var theLevel = _sound.soundLevel;
```

Voir aussi

Son

_system

Utilisation

```
-- Syntaxe Lingo
_system

// Syntaxe JavaScript
_system;
```

Description

Propriété de haut niveau ; fait référence à l'objet System, qui permet d'accéder aux informations en matière de système et d'environnement, y compris les méthodes système. En lecture seule.

Exemple

L'instruction suivante affecte la variable `objSystem` à la propriété `_système` :

```
-- Syntaxe Lingo
objSystem = _system

// Syntaxe JavaScript
var objSystem = _system;
```

Cette instruction utilise la propriété `_system` directement pour accéder à la propriété `freeBytes` :

```
-- Syntaxe Lingo
theBytes = _system.freeBytes

// Syntaxe JavaScript
var theBytes = _system.freeBytes;
```

Voir aussi

Système

aboutInfo

Utilisation

```
-- Syntaxe Lingo
_movie.aboutInfo

// Syntaxe JavaScript
_movie.aboutInfo;
```

Description

Propriété d'animation ; permet de saisir une chaîne dans la boîte de dialogue Propriétés de l'animation pendant la programmation. En lecture seule.

Exemple

Ces instructions affichent des informations au sujet de l'animation dans la fenêtre Messages.

```
-- Syntaxe Lingo
trace(_movie.aboutInfo)

// Syntaxe JavaScript
trace(_movie.aboutInfo);
```

Voir aussi

copyrightInfo (animation), Animation

actionsEnabled

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.actionsEnabled

// Syntaxe JavaScript
réfObjActeurOuImageObjet.actionsEnabled;
```

Description

Propriété d'acteur et propriété d'image-objet ; contrôle si les actions d'un contenu Macromedia Flash sont activées (TRUE, valeur par défaut) ou désactivées (FALSE).

Cette propriété peut être testée et définie.

Exemple

Le gestionnaire suivant accepte une référence d'image-objet comme paramètre, puis active ou désactive la propriété actionsEnabled de l'image-objet.

```
-- Syntaxe Lingo
on ToggleActions(quelleImageObjet)
  sprite(quelleImageObjet).actionsEnabled = \
  not(sprite(quelleImageObjet).actionsEnabled)
end
```

```
// Syntaxe JavaScript
function ToggleActions(quelleImageObjet) {
  switch(sprite(quelleImageObjet).actionsEnabled) {
    cas 0 :
      sprite(quelleImageObjet).actionsEnabled = 1;
      break();
    cas 1 :
      sprite(quelleImageObjet).actionsEnabled = 0;
      break();
  }
}
```

active3dRenderer

Utilisation

```
-- Syntaxe Lingo
_movie.active3dRenderer

// Syntaxe JavaScript
_movie.active3dRenderer;
```

Description

Propriété d'animation ; indique le moteur de rendu utilisé par l'animation pour le dessin des images-objets 3D. Cette propriété est l'équivalent de la propriété `getRendererServices().renderer`. En lecture seule.

Les valeurs possibles de la propriété `active3dRenderer` sont `#openGL`, `#directX7_0` et `#directX5_2`, ainsi que `#software`. Les valeurs `#openGL`, `#directX7_0` et `#directX5_2`, qui représentent des pilotes de carte vidéo, permettront d'obtenir de meilleures performances que `#software`, un moteur de rendu logiciel utilisé quand aucune des trois premières options n'est disponible.

Vous devrez utiliser `getRendererServices().renderer` pour définir cette propriété.

Exemple

Les exemples ci-dessous indiquent les deux façons de déterminer le moteur de rendu en cours d'utilisation.

```
-- Syntaxe Lingo
put(_movie.active3dRenderer)
put(getRendererServices().renderer)

// Syntaxe JavaScript
put(_movie.active3dRenderer);
put(getRendererServices().renderer);
```

Voir aussi

`Animation`, `renderer`

activeCastLib

Utilisation

```
-- Syntaxe Lingo
_player.activeCastLib

// Syntaxe JavaScript
_player.activeCastLib;
```

Description

Propriété de lecteur ; indique quelle distribution a été activée en dernier. En lecture seule.

La valeur de la propriété `activeCastLib` correspond au numéro de cette distribution.

La propriété `activeCastLib` est utile lors de l'utilisation de la propriété `selection` de l'objet Distribution. Utilisez-la pour déterminer la distribution à laquelle la sélection fait référence.

Exemple

Les instructions suivantes affectent les acteurs sélectionnés de la distribution la plus récemment utilisée à la variable `acteursSélectionnés` :

```
-- Syntaxe Lingo
castLibDintérêt = _player.activeCastLib
acteursSélectionnés = castLib(castLibDintérêt).selection

// Syntaxe JavaScript
var castLibDintérêt = _player.activeCastLib;
var acteursSélectionnés = castLib(castLibDintérêt).selection;
```

Voir aussi

`Lecteur`, `selection`

activeWindow

Utilisation

```
-- Syntaxe Lingo
_player.activeWindow

// Syntaxe JavaScript
_player.activeWindow;
```

Description

Propriété de lecteur ; indique la fenêtre d'animation qui est active. En lecture seule.

Dans le cas de l'animation principale, `activeWindow` correspond à la scène. Dans le cas d'une animation dans une fenêtre (MIAW), `activeWindow` est l'animation dans la fenêtre.

Exemple

L'exemple suivant place le mot Active dans la barre de titre de la fenêtre sur laquelle l'utilisateur a cliqué et place le mot Inactive dans la barre de titre de toutes les autres fenêtres ouvertes :

```
-- Syntaxe Lingo
on activateWindow
  fenêtreCliquée = _player.windowList.getPos(_player.activeWindow)
  nombreDeFenêtres = _player.windowList.count
   repeat with x = 1 to nombreDeFenêtres
    if (x = fenêtreCliquée) then
      _player.window[fenêtreCliquée].title = "Active"
    else
      _player.windowList[x].title = "Inactive"
    end if
  end repeat
end activateWindow

// Syntaxe JavaScript
function activateWindow() {
  var fenêtreCliquée = _player.windowList.getPos(_player.activeWindow);
  var nombreDeFenêtres = _player.windowList.count;
  for (var x = 1; x <= nombreDeFenêtres; x++) {
    if (x == fenêtreCliquée) {
      _player.window[fenêtreCliquée].title = "Active"
    }
    else {
      _player.windowList[x].title = "Inactive"
    }
  }
}
```

Voir aussi

Lecteur

actorList

Utilisation

```
-- Syntaxe Lingo
_movie.actorList

// Syntaxe JavaScript
_movie.actorList;
```

Description

Propriété d'animation ; liste d'objets enfants explicitement ajoutés à cette liste. Lecture/écriture.

Les objets d'`actorList` reçoivent un message `stepFrame` à chaque passage de la tête de lecture sur une image.

Pour ajouter un objet à `the actorList` utilisez `_movie.actorList.append (`*`réfObjNouveauScript`*`)`. Le gestionnaire `stepFrame` de l'objet situé dans son script parent ou ancêtre sera exécuté automatiquement à chaque avancée d'image.

Pour supprimer les objets de la liste `the actorList`, affectez-lui la valeur `[ ]`, qui correspond à une liste vide.

Director n'efface pas le contenu de `actorList` lorsqu'il passe à une autre animation, ce qui peut provoquer un comportement imprévisible chez elle. Pour empêcher le transfert des objets enfants de l'animation en cours à la nouvelle animation, insérez l'instruction `actorList = [ ]` dans le gestionnaire `prepareMovie` de la nouvelle animation.

Exemple

L'instruction suivante ajoute un objet enfant créé à partir du script parent Balle roulante. Les trois valeurs sont des paramètres dont le script a besoin.

L'instruction suivante affiche le contenu d'`actorList` dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(_movie.actorList)

// Syntaxe JavaScript
put(_movie.actorList);
```

L'instruction suivante efface les objets d'`actorList`.

```
-- Syntaxe Lingo
_movie.actorList = [] - avec des crochets
_movie.actorList = list() -- avec list()

// Syntaxe JavaScript
_movie.actorList = list();
```

Voir aussi

`Animation`, `on prepareMovie`, `on stepFrame`

alertHook

Utilisation

```
-- Syntaxe Lingo
_player.alertHook

// Syntaxe JavaScript
_player.alertHook;
```

Description

Propriété de lecteur ; spécifie un script parent contenant le gestionnaire `alertHook`. Lecture/écriture.

Utilisez `alertHook` pour contrôler l'affichage de messages d'alerte concernant des erreurs de fichiers ou des erreurs de script. Si une erreur survient alors qu'un script parent est affecté à `alertHook`, Director exécute le gestionnaire `alertHook` dans le script parent.

Bien qu'il soit possible de placer des gestionnaires `alertHook` dans des scripts d'animation, il est vivement recommandé de placer un gestionnaire `alertHook` dans un script parent ou de comportement pour éviter d'appeler accidentellement le gestionnaire à partir de toutes sortes d'emplacements, ce qui risquerait de créer une certaine confusion quant à l'emplacement de l'erreur.

Le gestionnaire `alertHook` étant exécuté en cas d'erreur, évitez de l'utiliser avec un script autre qu'un script de traitement d'erreur. Par exemple, le gestionnaire `alertHook` n'est pas l'emplacement approprié pour y inclure une instruction `go()`.

Le gestionnaire `alertHook` reçoit un argument d'instance, deux arguments de chaîne décrivant l'erreur et un argument facultatif spécifiant un événement supplémentaire invoquant le gestionnaire.

Le quatrième argument peut avoir l'une des quatre valeurs suivantes :

- `#alert`—provoque le déclenchement du gestionnaire, à l'aide de la méthode `alert()`.
- `#movie`—provoque le déclenchement du gestionnaire, à cause d'une erreur de fichier introuvable lors de l'exécution de la commande `go()`.
- `#script`—provoque le déclenchement du gestionnaire, à cause d'une erreur de script.
- `#safeplayer` —provoque le déclenchement du gestionnaire, par la vérification de la propriété `safePlayer`.

Suivant le script qu'il contient, le gestionnaire `alertHook` peut ignorer l'erreur ou la signaler d'une autre façon.

Exemple

L'instruction suivante précise que le script parent Alerte est le script déterminant si une alerte doit être affichée lorsqu'une erreur se produit. Si une erreur se produit, le script affecte les chaînes d'erreur et de message à l'acteur champ Sortie et renvoie la valeur 1.

```
-- Syntaxe Lingo
on prepareMovie
  _player.alertHook = script("Alerte")
end

-- script "Alerte"
on alertHook me, err, msg
  member("Sortie").text = err && msg
  return 1
end

// Syntaxe JavaScript
function prepareMovie() {
  _player.alertHook = alert("Type d'erreur", "Message d'erreur");
}

// gestionnaire d'alertes
function alert(err, msg) {
  member("Sortie").text = err + " " + msg;
  return 1;
}
```

Voir aussi

`alertHook`, `Lecteur`, `safePlayer`

alignment

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.alignment

// Syntaxe JavaScript
réfObjActeur.alignment;
```

Description

Propriété d'acteur ; détermine l'alignement utilisé pour afficher des caractères dans l'acteur spécifié. Cette propriété s'applique uniquement aux acteurs champ et texte qui contiennent des caractères (une espace suffit).

Pour les acteurs champ, la valeur de la propriété est une chaîne contenant l'un des termes suivants : `left`, `center` ou `right`.

Pour les acteurs texte, la valeur de la propriété est un symbole contenant l'un des éléments suivants : `#left`, `#center`, `#right` ou `#full`.

Le paramètre *`réfObjActeur`* peut être le nom ou le numéro d'un acteur.

Cette propriété peut être testée et définie. Pour les acteurs texte, la propriété peut être définie paragraphe par paragraphe.

Exemple

L'instruction suivante donne à la variable intitulée `alignementDeCaractère` la valeur courante de `alignment` pour l'acteur champ Pierre :

```
-- Syntaxe Lingo
alignementDeCaractère = member("Pierre").alignment

// Syntaxe JavaScript
var alignementDeCaractère = member("Pierre").alignment;
```

Voir aussi

`text`, `font`, `lineHeight`, `fontSize`, `fontStyle`, `&`, `+ (opérateur de concaténation)`, `&&`, `+ (opérateur de concaténation)`

allowCustomCaching

Utilisation

```
-- Syntaxe Lingo
_movie.allowCustomCaching

// Syntaxe JavaScript
_movie.allowCustomCaching;
```

Description

Propriété d'animation ; contiendra les informations concernant un cache privé dans les futures versions de Director. Lecture/écriture.

La valeur par défaut de cette propriété est TRUE.

Voir aussi

allowGraphicMenu, allowSaveLocal, allowTransportControl, allowVolumeControl, allowZooming, Animation

allowGraphicMenu

Utilisation

```
-- Syntaxe Lingo
_movie.allowGraphicMenu

// Syntaxe JavaScript
_movie.allowGraphicMenu;
```

Description

Propriété d'animation ; contrôle la disponibilité de contrôles graphiques dans le menu contextuel lors de la lecture de l'animation dans un environnement Macromedia Shockwave. Lecture/écriture.

Affectez à cette propriété la valeur FALSE si vous préférez afficher un menu textuel plutôt que le menu contextuel graphique.

La valeur par défaut de cette propriété est TRUE.

Voir aussi

allowCustomCaching, allowSaveLocal, allowTransportControl, allowVolumeControl, allowZooming, Animation

allowSaveLocal

Utilisation

```
-- Syntaxe Lingo
_movie.allowSaveLocal

// Syntaxe JavaScript
_movie.allowSaveLocal;
```

Description

Propriété d'animation ; contrôle la disponibilité d'un contrôle d'enregistrement dans le menu contextuel lors de la lecture de l'animation dans un environnement Shockwave. Lecture/écriture.

Cette propriété permettra des améliorations dans de futures versions de Shockwave Player.

La valeur par défaut de cette propriété est TRUE.

Voir aussi

allowCustomCaching, allowGraphicMenu, allowTransportControl, allowVolumeControl, allowZooming, Animation

allowTransportControl

Utilisation

```
-- Syntaxe Lingo
_movie.allowTransportControl

// Syntaxe JavaScript
_movie.allowTransportControl;
```

Description

Propriété d'animation ; cette propriété permettra des améliorations dans de futures versions de Shockwave Player. Lecture/écriture.

La valeur par défaut de cette propriété est `TRUE`.

Voir aussi

`allowCustomCaching`, `allowGraphicMenu`, `allowSaveLocal`, `allowVolumeControl`, `allowZooming`, `Animation`

allowVolumeControl

Utilisation

```
-- Syntaxe Lingo
_movie.allowVolumeControl

// Syntaxe JavaScript
_movie.allowVolumeControl;
```

Description

Propriété d'animation ; contrôle la disponibilité d'un contrôle de volume dans le menu contextuel lors de la lecture de l'animation dans un environnement Shockwave Player. Lecture/écriture.

L'un des contrôles de volume est activé ou désactivé selon que vous affectez respectivement `TRUE` ou `FALSE` à cette propriété.

La valeur par défaut de cette propriété est `TRUE`.

Voir aussi

`allowCustomCaching`, `allowGraphicMenu`, `allowSaveLocal`, `allowTransportControl`, `allowZooming`, `Animation`

allowZooming

Utilisation

```
-- Syntaxe Lingo
_movie.allowZooming

// Syntaxe JavaScript
_movie.allowZooming;
```

Description

Propriété d'animation ; détermine si l'utilisateur peut agrandir ou zoomer sur l'animation lors de la lecture dans Shockwave Player et ShockMachine. Lecture/écriture.

Affectez la valeur `FALSE` à cette propriété pour empêcher le redimensionnement de l'animation dans un navigateur ou dans ShockMachine.

La valeur par défaut de cette propriété est `TRUE`.

Voir aussi

`allowCustomCaching`, `allowGraphicMenu`, `allowSaveLocal`, `allowTransportControl`, `allowVolumeControl`, `Animation`

alphaThreshold

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.alphaThreshold

// Syntaxe JavaScript
réfObjActeur.alphaThreshold;
```

Description

Propriété d'acteur bitmap ; régit l'incidence de la couche alpha du bitmap sur la détection des clics de souris. Cette propriété peut avoir une valeur de 0 à 255, en exacte correspondance avec les valeurs alpha de la couche alpha pour un bitmap 32 bits.

Pour un paramètre `alphaThreshold` donné, Director détecte un clic de souris si la valeur du pixel correspondant à ce point dans la couche alpha est égale ou supérieure au seuil. Si vous affectez la valeur 0 à `alphaThreshold`, vous rendez tous les pixels opaques à la détection des clics de souris, quel que soit le contenu de la couche alpha.

Voir aussi

`useAlpha`

ambient

Utilisation

```
member(quelActeur).shader(quelMatériau).ambient
member(quelActeur).model(quelModèle).shader.ambient
member(quelActeur).model(quelModèle).shaderList[[index]].\
  ambient
```

Description

Propriété 3D de matériau `#standard` ; indique la quantité de chaque composante de couleur de la lumière ambiante de l'acteur reflétée par le matériau.

Par exemple, si la couleur de la lumière ambiante est `rgb(255, 255, 255)` et la valeur de la propriété `ambient` du matériau est `rgb(255, 0, 0)`, le matériau reflètera toute la composante rouge de la lumière que les couleurs du matériau peuvent refléter. Toutefois, quelles que soient les couleurs du matériau, il ne reflètera pas les composantes bleue et verte de la lumière. Dans ce cas, s'il n'y a pas d'autres lumières dans la scène, le bleu et le vert du matériau ne refléteront aucune couleur et apparaîtront en noir.

La valeur par défaut de cette propriété est `rgb(63,63,63)`.

Exemple

L'instruction suivante donne à la propriété `ambient` du modèle Fauteuil la valeur `rgb(255, 255, 0)`. Le modèle Fauteuil reflètera entièrement dans la scène les composantes rouge et verte de la lumière ambiante et ignorera la composante bleue.

```
member("Pièce").model("Fauteuil").shader.ambient = rgb(255, 0, 0)
```

Voir aussi

`ambientColor`, `newLight`, `type (lumière)`, `diffuse`, `specular (matériau)`

ambientColor

Utilisation

```
member(quelActeur).ambientColor
```

Description

Propriété 3D d'acteur ; indique la couleur RVB de la lumière ambiante par défaut de l'acteur.

La valeur par défaut de cette propriété est `rgb(0, 0, 0)`. Cela n'ajoute aucune lumière à la scène.

Exemple

L'instruction suivante donne à la propriété `ambientColor` de l'acteur Pièce la valeur `rgb(255, 0, 0)`. La lumière ambiante par défaut de l'acteur sera rouge. Cette propriété peut également être définie dans l'inspecteur des propriétés.

```
member("Pièce").ambientColor = rgb(255, 0, 0)
```

Voir aussi

`directionalColor`, `directionalPreset`, `ambient`

ancestor

Utilisation

```
property {propriétésOption} ancestor
```

Description

Propriété d'objet ; permet aux objets enfants et aux comportements d'utiliser des gestionnaires qui ne sont pas contenus dans le script parent ou le comportement.

La propriété `ancestor` est généralement utilisée avec deux scripts parents ou plus. Vous pouvez utiliser cette propriété pour permettre aux objets enfants et aux comportements de partager certains comportements hérités d'un ancêtre, tout en ayant des comportements différents hérités de leurs parents.

Dans le cas des objets enfants, la propriété `ancestor` est en général affectée dans le gestionnaire `on new` du script parent. Lorsque vous envoyez un message à un objet enfant et que celui-ci ne dispose pas d'un gestionnaire, ce message est envoyé directement au script défini par la propriété `ancestor`.

Si un comportement a un ancêtre, cet ancêtre reçoit des événements de souris tels que `mouseDown` et `mouseWithin`.

La propriété `ancestor` vous permet de modifier les comportements et propriétés d'un grand nombre d'objets avec une seule commande.

Le script ancêtre peut contenir des variables de propriétés indépendantes accessibles par les objets enfants. Pour faire référence aux variables de propriétés du script ancêtre, respectez la syntaxe suivante :

```
me.variableDePropriété = valeur
```

Par exemple, l'instruction suivante fait passer la valeur de la variable de propriété `nombreDePattes` d'un script ancêtre à 4 :

```
me.nombreDePattes = 4
```

Utilisez la syntaxe `the nomDeVariable of` nomDeScript pour accéder aux variables de propriétés qui ne sont pas contenues dans l'objet courant. L'instruction suivante permet à la variable `monNombreDePattes` de l'objet enfant d'accéder à la variable de propriété `nombreDePattes` du script ancêtre :

```
set monNombreDePattes to the nombreDePattes of me
```

Exemple

Chacun des scripts suivants est un acteur. Le script ancêtre Animal et les scripts parents Chien et Homme interagissent pour définir des objets.

Le premier script, Chien, paramètre la variable de propriété `race` sur Bâtard, affecte à `the ancestor of Chien` le script Animal et donne à la variable `nombreDePattes` du script ancêtre la valeur 4 :

```
property race, ancestor
on new me
  set race = "Bâtard"
  set the ancestor of me to new(script "Animal")
  set the nombreDePattes of me to 4
  return me
end
```

Le second script, Homme, paramètre la variable de propriété `ethnie` sur Caucasien, affecte à `the ancestor of Homme` le script Animal et donne à la variable `nombreDePattes` du script ancêtre la valeur 2 :

```
property ethnie, ancestor
on new me
  set ethnie to "Caucasien"
  set the ancestor of me to new(script "Animal")
  set the nombreDePattes of me to 2
  return me
end
```

Voir aussi

`new()`, `me`, `property`

angle (3D)

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  emitter.angle
```

Description

Propriété 3D d'émission ; décrit la région dans laquelle les particules d'un système de particules sont émises. Un système de particules est une ressource de modèle de type `#particle`.

La principale direction de l'émission des particules est le vecteur défini par la propriété `direction` de l'émetteur. Toutefois, la direction de l'émission d'une particule donnée déviera de ce vecteur selon un angle aléatoire compris entre 0 et la valeur de la propriété `angle` de l'émetteur.

La plage de cette propriété s'étend de 0.0 à 180.0. La valeur par défaut est 180.0.

Exemple

L'instruction suivante donne à l'angle d'émission de la ressource de modèle mrFont la valeur 1, ce qui entraîne l'émission des particules en une fine ligne.

```
member("fontaine").modelResource("mrFont").emitter.angle = 1
```

Voir aussi

`emitter`, `direction`

angle (DVD)

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.angle

// Syntaxe JavaScript
réfObjDvd.angle;
```

Description

Propriété DVD ; renvoie la valeur numérique de l'angle courant de la caméra. En lecture seule.

La valeur renvoyée est un nombre entier.

Exemple

Cette instruction renvoie le nombre d'angles de caméras actuels :

```
-- Syntaxe Lingo
put(sprite(12).angle)

// Syntaxe JavaScript
put(sprite(12).angle);
```

Voir aussi

`DVD`

angleCount

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.angleCount

// Syntaxe JavaScript
réfObjDvd.angleCount;
```

Description

Propriété DVD ; renvoie le nombre d'angles de caméra disponibles dans le titre courant. En lecture seule.

La valeur renvoyée est un nombre entier compris entre 1 et 9.

Exemple

Cette instruction renvoie le nombre d'angles de caméras disponibles :

```
-- Syntaxe Lingo
put(member(12).angleCount)

// Syntaxe JavaScript
put(member(12).angleCount);
```

Voir aussi

DVD

animationEnabled

Utilisation

```
member(quelActeur).animationEnabled
```

Description

Propriété 3D d'acteur ; indique si les mouvements seront exécutés (TRUE) ou ignorés (FALSE). Cette propriété peut également être définie dans l'inspecteur des propriétés.

La valeur par défaut de cette propriété est TRUE.

Exemple

L'instruction suivante désactive l'animation pour l'acteur Scène.

```
member("Séquence").animationEnabled = FALSE
```

antiAlias

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.antiAlias

// Syntaxe JavaScript
réfObjActeurOuImageObjet.antiAlias;
```

Description

Propriété d'acteur ; contrôle si le rendu d'un acteur texte, forme vectorielle ou Flash repose sur l'anti-aliasing pour assurer une qualité élevée de rendu au détriment de la vitesse de lecture de l'animation. La propriété antiAlias a la valeur TRUE par défaut.

Dans le cas des formes vectorielles, la valeur TRUE correspond au paramètre de qualité #high (qualité supérieure) d'un élément Flash, alors que la valeur FALSE correspond au paramètre #low (basse qualité).

La propriété antiAlias peut également être utilisée comme propriété d'image-objet et ce, uniquement pour les images-objets forme vectorielle.

Cette propriété peut être testée et définie.

Exemple

Le comportement suivant vérifie le codage des couleurs du moniteur sur lequel l'animation est en cours de lecture. Si ce codage est réglé sur 8 bits ou moins (256 couleurs), le script affecte la valeur `FALSE` à la propriété `antiAlias` de l'image-objet.

```
-- Syntaxe Lingo
property spriteNum

on beginsprite me
  if _system.colorDepth <= 8 then
    sprite(spriteNum).antiAlias = FALSE
  end if
end

// Syntaxe JavaScript
function beginsprite() {
  var cd = _system.colorDepth;
  if (cd <= 8 ) {
    sprite(this.spriteNum).antiAlias = 0;
  }
}
```

Voir aussi

`antiAliasThreshold`, `quality`

antiAliasingEnabled

Utilisation

`sprite(`*`quelleImageObjet`*`).antiAliasingEnabled`

Description

Propriété 3D d'image-objet ; indique si l'univers 3D de l'image-objet *`quelleImageObjet`* est anti-aliasé. Elle peut être testée et définie. La valeur par défaut est `FALSE`, ce qui indique que l'anti-aliasing est désactivé. Si la propriété `antiAliasingEnabled` a pour valeur `TRUE` et que le moteur de rendu 3D est remplacé par un moteur ne supportant pas l'anti-aliasing, la propriété prend la valeur `FALSE`. La valeur de cette propriété n'est pas enregistrée avec l'animation.

Les images-objets anti-aliasées imposent une charge supplémentaire au processeur et ont besoin d'une plus grande quantité de mémoire. La désactivation temporaire de l'anti-aliasing peut améliorer les performances des effets d'animation et l'interaction avec l'utilisateur.

Exemple

L'instruction Lingo suivante vérifie si le moteur de rendu 3D actuel de l'image-objet 2 prend en charge l'anti-aliasing à l'aide de la propriété `antiAliasingSupported`. Si l'anti-aliasing est pris en charge, la seconde instruction active l'anti-aliasing pour l'image-objet avec la propriété `antiAliasingEnabled`.

```
if sprite(2).antiAliasingSupported = TRUE then
  sprite(2).antiAliasingEnabled = TRUE
end if
```

Voir aussi

`antiAliasingSupported`, `renderer`, `rendererDeviceList`

antiAliasingSupported

Utilisation

```
sprite(quelleImageObjet).antiAliasingSupported
```

Description

Propriété 3D d'image-objet ; indique si l'anti-aliasing est pris en charge par le moteur de rendu 3D actuel. Cette propriété peut être testée, mais pas définie. Cette propriété renvoie `TRUE` ou `FALSE`.

Exemple

L'instruction Lingo suivante vérifie si le moteur de rendu 3D actuel de l'image-objet 3 supporte l'anti-aliasing. Si l'anti-aliasing est pris en charge, la seconde instruction active l'anti-aliasing pour l'image-objet avec la propriété `antiAliasingEnabled`.

```
if sprite(3).antiAliasingSupported = TRUE then
  sprite(3).antiAliasingEnabled = TRUE
end if
```

Voir aussi

`antiAliasingEnabled`, `renderer`, `rendererDeviceList`

antiAliasThreshold

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.antiAliasThreshold

// Syntaxe JavaScript
réfObjActeur.antiAliasThreshold;
```

Description

Propriété d'acteur texte ; contrôle la taille de point pour laquelle l'anti-aliasing est automatiquement appliqué à un acteur texte. Elle n'a d'effet que lorsque la propriété `antiAlias` de l'acteur texte a la valeur `TRUE`.

Le paramètre est un nombre entier indiquant la taille (en points) pour laquelle l'anti-aliasing est exécuté.

Cette propriété a une valeur par défaut de 14 points.

Voir aussi

`antiAlias`

appearanceOptions

Utilisation

```
-- Syntaxe Lingo
réfObjFenêtre.appearanceOptions

// Syntaxe JavaScript
réfObjFenêtre.appearanceOptions;
```

Description

Propriété de fenêtre; indique une liste de propriétés qui répertorie les options d'aspect d'une fenêtre. Lecture/écriture.

La liste de propriétés contient les propriétés suivantes.

Propriété	Description
`#mask`	Indique l'acteur 1-bit à utiliser en tant que masque de la fenêtre.
`#border`	Indique le type de bordure de la fenêtre. Cette propriété peut avoir l'une des trois valeurs suivantes : • `#none`. Indique l'absence de bordure autour de la fenêtre. • `#line`. Indique une bordure noire de 1 pixel autour de la fenêtre. Les propriétés #none et #line ne sont efficaces que si la propriété `titlebarOptions.visible` a pour valeur `FALSE`.
`#metal`	(Macintosh uniquement) Indique si la fenêtre doit avoir un aspect métallique (`TRUE`) ou pas. Si `FALSE`, la fenêtre aura un aspect glacé.
`#dragRegionMask`	Indique l'acteur 1-bit à utiliser en tant que masque d'une région de la fenêtre.
`#shadow`	(Macintosh uniquement) Indique si la fenêtre doit avoir une ombre ou pas. En général, les fenêtres Macintosh en ont une.
`#liveresize`	(Macintosh uniquement) Indique si la fenêtre doit comporter un redimensionnement dynamique. Si `TRUE`, le redimensionnement dynamique est activé. Si `FALSE`, le redimensionnement dynamique est désactivé.

Il est également possible d'accéder à ces propriétés à l'aide de la propriété `displayTemplate` de l'objet d'animation.

Exemple

Cette instruction affiche toutes les options rélatives à l'aspect de la fenêtre intitulée Tableau de commande, dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(window("Tableau de commande").appearanceOptions)

// Syntaxe JavaScript
put(window("Tableau de commande").appearanceOptions);
```

Cette instruction affecte à la propriété border une bordure d'1 pixel autour de la fenêtre intitulée Tableau de commande :

```
-- Syntaxe Lingo
window("Tableau de commande").appearanceOptions.border = #line

// Syntaxe JavaScript
window("Tableau de commande").appearanceOptions.border = symbol("line");
```

Voir aussi

displayTemplate, titlebarOptions, visible, Fenêtre

applicationName

Utilisation

```
-- Syntaxe Lingo
_player.applicationName

// Syntaxe JavaScript
_player.applicationName;
```

Description

Propriété de lecteur ; indique le nom de la copie en cours d'exécution de l'application Director pendant la programmation, ou le nom d'un fichier de projection au moment de l'exécution. En lecture seule.

La valeur de la propriété est une chaîne.

Shockwave Player ne prend pas en charge cette propriété.

Exemple

Cette instruction affiche le nom de l'application Director, Director.exe.

```
-- Syntaxe Lingo
put(_player.applicationName)

// Syntaxe JavaScript
put(_player.applicationName);
```

Voir aussi

applicationPath, Lecteur

applicationPath

Utilisation

```
-- Syntaxe Lingo
_player.applicationPath

// Syntaxe JavaScript
_player.applicationPath;
```

Description

Propriété de lecteur ; détermine le chemin d'accès ou l'emplacement du dossier contenant l'exemplaire de l'application Director en cours d'exécution pendant la programmation ou le dossier contenant la projection pendant l'exécution. En lecture seule.

La valeur de la propriété est une chaîne.

Si vous utilisez `applicationPath` suivi de `&` et d'un chemin de dossier en aval, mettez l'expression entière entre parenthèses de façon à ce que le script l'analyse comme un tout.

Shockwave Player ne prend pas en charge cette propriété.

Exemple

L'instruction suivante affiche le chemin du dossier contenant l'application Director.

```
-- Syntaxe Lingo
put(_player.applicationPath)

// Syntaxe JavaScript
put(_player.applicationPath);
```

L'instruction suivante ouvre l'animation Sunset Boulevard dans une fenêtre (sur un ordinateur Windows) :

```
-- Syntaxe Lingo
window(_player.applicationPath & "\Film Noir\Sunset Boulevard").open()

// Syntaxe JavaScript
window(_player.applicationPath + "\Film Noir\Sunset Boulevard").open();
```

Voir aussi

`applicationName`, `Lecteur`

aspectRatio

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.aspectRatio

// Syntaxe JavaScript
réfObjDvd.aspectRatio;
```

Description

Propriété DVD. Renvoie une liste de propriétés indiquant la largeur et la hauteur de l'acteur DVD. En lecture seule.

Aussi bien la largeur que la hauteur figurent sous forme de nombres entiers.

Exemple

L'instruction suivante renvoie la valeur `aspectRatio` de l'acteur 1 :

```
-- Syntaxe Lingo
trace(member(1).aspectRatio) -- [#width: 16, #height:9]

// Syntaxe JavaScript
trace(member(1).aspectRatio); // ["width": 16, "height":9];
```

Voir aussi

`DVD`

attenuation

Utilisation

```
member(quelActeur).light(quelleLumière).attenuation
```

Description

Propriété 3D de lumière ; indique les facteurs d'atténuation constante, linéaire et quadratique pour les projecteurs et les points lumineux.

La valeur par défaut de cette propriété est vector(1.0, 0.0, 0.0).

Exemple

L'instruction suivante donne à la propriété `attenuation` de la lumière Lampe la valeur `vector (.5, 0, 0)`, ce qui l'assombrit légèrement.

```
member("Univers 3D").light("Lampe").attenuation = \
  vector(.5, 0, 0)
```

Voir aussi

color (lumière)

attributeName

Utilisation

```
nœudXML.attributeName[ numéroDattribut ]
```

Description

Propriété XML ; renvoie le nom du nœud enfant spécifié d'un document XML analysé.

Exemple

Avec le code XML suivant :

```
<?xml version="1.0"?>
  <e1>
    <nomDeBalise attr1="val1" attr2="val2"/>
    <e2>élém 2</e2>
    <e3>élém 3</e3>
    Exemple de texte
  </e1>
```

- L'instruction Lingo suivante renvoie le nom du premier attribut de la balise appelée `nomDeBalise` :

```
put gObjetDanalyse.child[1].child[1].attributeName[1]
-- "attr1"
```

Voir aussi

attributeValue

attributeValue

Utilisation

```
nœudXML.attributeValue[ nomOuNuméroDattribut ]
```

Description

Propriété XML ; renvoie la valeur du nœud enfant spécifié d'un document XML analysé.

Exemple

Avec le code XML suivant :

```
<?xml version="1.0"?>
  <e1>
    <nomDeBalise attr1="val1" attr2="val2"/>
    <e2>élém 2</e2>
    <e3>élém 3</e3>
    Exemple de texte
  </e1>
```

L'instruction Lingo suivante renvoie la valeur du premier attribut de la balise appelée `nomDeBalise` :

```
put gObjetDanalyse.child[1].child[1].attributeValue[1]
-- "val1"
```

Voir aussi

attributeName

audio (DVD)

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.audio

// Syntaxe JavaScript
réfObjDvd.audio;
```

Description

Propriété DVD. Détermine si l'audio est activé (`TRUE`, par défaut) ou pas (`FALSE`). Lecture/écriture.

Exemple

Cette instruction désactive le son :

```
-- Syntaxe Lingo
member(14).audio = 0

// Syntaxe JavaScript
member(14).audio = 0;
```

Voir aussi

DVD

audio (RealMedia)

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.audio

// Syntaxe JavaScript
réfObjActeurOuImageObjet.audio;
```

Description

Propriété d'acteur ou image-objet RealMedia ; permet de lire (`TRUE`) ou de mettre en sourdine (`FALSE`) l'audio du flux RealMedia. Le paramètre par défaut de cette propriété est `TRUE`. Les valeurs entières autres que 1 ou 0 sont traitées comme `TRUE`. La définition de cette propriété n'a aucun effet si la méthode `realPlayerNativeAudio()` a pour valeur `TRUE`.

Si la propriété `audio` a pour valeur `FALSE` lorsque démarre la lecture d'un acteur RealMedia, la piste audio reste affectée, ce qui vous permet d'activer ou de désactiver le son lors de la lecture.

Une certaine latence peut se produire lors de la définition de cette propriété, ce qui signifie qu'un léger délai peut être observé avant l'activation ou la désactivation du son.

Exemple

Les exemples suivants donnent à la propriété `audio` de l'image-objet 2 et de l'acteur Real la valeur `TRUE`, ce qui signifie que la portion audio du flux RealMedia sera lue.

```
-- Syntaxe Lingo
put(sprite(2).audio) -- 1
put(member("Real").audio) -- 1

// Syntaxe JavaScript
put(sprite(2).audio); // 1
put(member("Real").audio); // 1
```

Le code suivant donne à la propriété `audio` de l'image-objet 2 et de l'acteur Real la valeur `FALSE`, ce qui signifie que la portion audio du flux RealMedia ne sera pas lue en même temps que l'animation.

```
-- Syntaxe Lingo
sprite(2).audio = FALSE
member("Real").audio = FALSE

// Syntaxe JavaScript
sprite(2).audio = 0;
member("Real").audio = 0;
```

Voir aussi

soundChannel (RealMedia), video (RealMedia, Windows Media), sound (lecteur)

audio (Windows Media)

Utilisation

```
-- Syntaxe Lingo
réfObjWindowsMedia.audio

// Syntaxe JavaScript
réfObjWindowsMedia.audio;
```

Description

Propriété Windows Media. Détermine si l'audio est activé (TRUE, par défaut) ou pas (FALSE) lors de la lecture. Lecture/écriture.

Exemple

Cette instruction affiche, dans la fenêtre Messages, si le son est activé ou non pour l'acteur 5 :

```
-- Syntaxe Lingo
trace(member(5).audio)

// Syntaxe JavaScript
trace(member(5).audio);
```

Voir aussi

Windows Media

audioChannelCount

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.audioChannelCount

// Syntaxe JavaScript
réfObjDvd.audioChannelCount;
```

Description

Propriété DVD ; renvoie le nombre de pistes audio. En lecture seule.

Voir aussi

DVD

audioExtension

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.audioExtension

// Syntaxe JavaScript
réfObjDvd.audioExtension;
```

Description

Propriété DVD. Renvoie un symbole qui indique, le cas échéant, les extensions de langage audio d'un flux audio. En lecture seule.

Les valeurs renvoyées possibles sont les suivantes :

Symbole	Description
#caption	Le flux audio contient des sous-titres.
#lowvision	Le flux audio comporte du contenu pour les personnes dont la vision est déficiente.
#directorcomments1	Le flux audio contient des « commentaires director 1. »
#directorcomments2	Le flux audio contient des « commentaires director 2. »
#none	Le DVD n'indique aucune extension de langage audio pour ce flux audio, ou elle n'a pas pu être définie.

Voir aussi

DVD

audioFormat

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.audioFormat

// Syntaxe JavaScript
réfObjDvd.audioFormat;
```

Description

Propriété DVD. Renvoie un symbole qui indique le format (mode d'encodage) d'un flux audio. En lecture seule.

Les valeurs renvoyées possibles sont les suivantes :

Symbole	Description
#AC3	Le format audio est Dolby AC-3.
#MPEG1	Le format audio est MPEG-1.
#MPEG1DRC	Le format audio est MPEG-1 avec contrôle de plage dynamique.
#MPEG2	Le format audio est MPEG-2.
#MPEG2DRC	Le format audio est MPEG-2 avec contrôle de plage dynamique.
#LPCM	Le format audio est LPCM (Linear Pulse Code Modulated).
#DTS	Le format audio est DTS (Digital Theater Systems).
#SDDS	Le format audio est SDDS (Sony Dynamic Digital Sound).

Voir aussi

DVD

audioSampleRate

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.audioSampleRate

// Syntaxe JavaScript
réfObjDvd.audioSampleRate;
```

Description

Propriété DVD ; renvoie la fréquence, en hertz, d'un flux audio. En lecture seule.

Voir aussi

DVD

audioStream

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.audioStream

// Syntaxe JavaScript
réfObjDvd.audioStream;
```

Description

Propriété DVD. Renvoie le flux audio actuellement utilisé. Lecture/écriture.

Les valeurs correctes sont comprises entre 1 et 8.

Voir aussi

DVD

audioStreamCount

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.audioStreamCount

// Syntaxe JavaScript
réfObjDvd.audioStreamCount;
```

Description

Propriété DVD ; renvoie le nombre de flux audio disponibles dans le titre courant. En lecture seule.

Le nombre de flux audio disponibles est compris entre 1 et 8.

Voir aussi

DVD

auto

Utilisation

```
member(quelActeur).model(quelModèle).lod.auto
```

Description

Propriété de modificateur 3D `lod` ; permet au modificateur de gérer la réduction des détails dans le modèle au fur et à mesure que la distance entre le modèle et la caméra change.

La définition de la propriété `bias` du modificateur détermine dans quelle mesure le modificateur peut réduire les détails du modèle lorsque la propriété `auto` a pour valeur `TRUE`.

Le modificateur met sa propriété `level` à jour en même temps qu'il ajuste le niveau de détails du modèle. La définition de la propriété `level` n'a aucun effet, sauf lorsque la propriété `auto` a pour valeur `FALSE`.

Le modificateur `#lod` ne peut être ajouté qu'aux modèles créés dans un programme de modélisation 3D autre que Director. La valeur de la propriété `type` des ressources de modèle utilisées par ces modèles est `#fromFile`. Le modificateur ne peut pas être ajouté aux primitives créées dans Director.

Exemple

L'instruction suivante donne à la propriété `auto` du modificateur `lod` du modèle vaisseauSpatial la valeur `TRUE`. Le modificateur définira automatiquement le niveau de détails du modèle.

```
member("Univers 3D").model("vaisseauSpatial").lod.auto = TRUE
```

Voir aussi

`lod (modificateur)`, `bias`, `level`

autoblend

Utilisation

```
member(quelActeur).model(quelModèle).\
   keyframePlayer.autoblend
member(quelActeur).model(quelModèle).bonesPlayer.autoblend
```

Description

Propriété de modificateur 3D `keyframePlayer` et `bonesPlayer` ; indique si le modificateur crée une transition linéaire entre le mouvement en cours de lecture et le mouvement précédent `(TRUE)` ou pas `(FALSE)`. Si `autoBlend` est `TRUE`, la durée de la transition est définie par la propriété `blendTime` du modificateur. Si `autoBlend` est `FALSE`, la transition est contrôlée par la propriété `blendFactor` du modificateur et `blendTime` est ignoré.

La fusion des mouvements est totalement désactivée lorsque `blendTime` a pour valeur 0 et `autoBlend` pour valeur `TRUE`.

La valeur par défaut de cette propriété est `TRUE`.

Exemple

L'instruction suivante désactive `autoblend` pour le modèle Martien3. Le paramètre `blendFactor` pour le modèle sera utilisé pour fusionner plusieurs mouvements successifs de la liste de lecture.

```
member("nouveauxMartiens").model("Martien3").keyframePlayer.\
  autoblend = FALSE
```

Voir aussi

`blendFactor`, `blendTime`

autoCameraPosition

Utilisation

```
member(quelActeurTexte).autoCameraPosition
```

Description

Propriété 3D de caméra ; indique si la caméra de l'acteur texte 3D est automatiquement positionnée pour afficher tout le texte (`TRUE`) ou pas (`FALSE`). Cela est utile lors de la modification du texte, de la police ou de sa taille, et d'autres propriétés de l'acteur.

Cette propriété n'est pas valide avec d'autres types d'acteurs 3D.

Exemple

L'instruction suivante donne à la propriété `autoCameraPosition` de l'acteur Titres la valeur `FALSE`. Lorsque l'acteur est affiché en mode 3D, la caméra n'est pas positionnée automatiquement.

```
member("Titres").autoCameraPosition = FALSE
```

Voir aussi

`displayMode`

autoMask

Utilisation

```
member(quelActeurCurseur).autoMask
the autoMask of member quelActeur
```

Description

Propriété d'acteur; spécifie si les pixels blancs de l'acteur curseur couleur animé *quelActeurCurseur* sont transparents, pour permettre l'affichage de l'arrière-plan (`TRUE`, valeur par défaut), ou opaque (`FALSE`).

Exemple

Dans le script suivant, lorsque le curseur animé personnalisé stocké dans l'acteur 5 entre dans l'image-objet, la fonction de masque automatique est activée de sorte que l'arrière-plan de l'image-objet s'affiche au travers des pixels blancs. La fonction de masque automatique est désactivée dès que le curseur quitte l'image-objet.

```
-- Syntaxe Lingo
on mouseEnter
  member(5).autoMask = TRUE
end

on mouseLeave
  member(5).autoMask = FALSE
end
```

Syntaxe verbose :

```
on mouseEnter
  set the autoMask of member 5 = TRUE
end

on mouseLeave
  set the autoMask of member 5 = FALSE
end
```

autoTab

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.autoTab

// Syntaxe JavaScript
réfObjActeur.autoTab;
```

Description

Propriété d'acteur ; détermine l'effet qu'une pression sur la touche Tab a sur le champ modifiable ou l'acteur texte spécifié par *quelActeur*. Cette propriété peut être rendue active (TRUE) ou inactive (FALSE). L'ordre de tabulation dépend du numéro des images-objets et non de leur position sur la scène.

Exemple

L'instruction suivante entraîne l'acteur Commentaires à avancer automatiquement le point d'insertion au champ modifiable ou à l'image-objet texte suivant(e) lorsque l'utilisateur appuie sur Tab.

```
-- Syntaxe Lingo
member("Commentaires").autotab = TRUE

// Syntaxe JavaScript
member("Commentaires").autotab = true;
```

axisAngle

Utilisation

```
member(quelActeur).model(quelModèle).transform.axisAngle
member(quelActeur).camera(quelleCaméra).transform.axisAngle
member(quelActeur).light(quelleLumière).transform.axisAngle
member(quelActeur).group(quelGroupe).transform.axisAngle
référenceDeTransformation.axisAngle
```

Description

Propriété 3D de transformation ; décrit la rotation de la transformation comme une paire axe/angle.

La propriété `axisAngle` est une liste linéaire contenant un vecteur (l'axe) et une valeur à virgule flottante (l'angle). Le vecteur est l'axe de pivotement de la transformation. La valeur à virgule flottante est l'importance, en degrés, de la rotation.

La valeur par défaut de cette propriété est `[vector( 1.0000, 0.0000, 0.0000 ), 0.0000]`.

Exemple

L'instruction suivante indique la rotation du modèle boîteAuxLettres. Le modèle pivote de 145,5 degrés dans le sens opposé aux aiguilles d'une montre autour de l'axe des y.

```
put member("Terrain").model("boîteAuxLettres").transform.axisAngle
-- [vector( 0.0000, 1.0000, 0.0000 ), -145.5000]
```

Voir aussi

rotation (transformation)

back

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).back
```

Description

Propriété de ressource de modèle 3D #box ; indique si le côté de la boîte coupé par son axe des z positif est fermé (`TRUE`) ou ouvert (`FALSE`).

La valeur par défaut de cette propriété est `TRUE`.

Exemple

L'instruction suivante donne à la propriété `back` de la ressource de modèle Caisse la valeur `FALSE`, ce qui signifie que l'arrière de la caisse sera ouvert.

```
member("Univers 3D").modelResource("caisse").back = FALSE
```

Voir aussi

bottom (3D), front, top (3D), left (3D), right (3D)

backColor

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.backColor

// Syntaxe JavaScript
réfObjImageObjet.backColor;
```

Description

Propriété d'image-objet ; définit la couleur d'arrière-plan de l'image-objet spécifiée en fonction de la valeur de couleur affectée. Lecture/écriture.

La définition de `backColor` pour une image-objet revient à choisir la couleur d'arrière-plan dans la palette des outils lorsque l'image-objet est sélectionnée sur la scène. Pour que la valeur définie par un script dure au-delà de l'image-objet courante, un script doit être affecté à l'image-objet. La couleur d'arrière-plan s'applique à tous les acteurs bitmap, en plus des acteurs champ, bouton, case et radio.

La valeur de `backColor` va de 0 à 255 pour un codage sur 8 bits et de 0 à 15 pour un codage sur 4 bits. Ces numéros correspondent au numéro d'index de la couleur d'arrière-plan dans la palette en cours. Ce numéro apparaît dans l'angle inférieur gauche de la palette des couleurs lorsque vous cliquez sur la couleur.

Si cette propriété est définie sur des acteurs bitmap supérieurs à 1 bit, `backColor` ne s'affichera peut-être pas si l'arrière-plan du bitmap n'est pas visible.

Si l'opacité d'une image-objet est inférieure à 100, mais supérieure à 0, `backColor` se mélangera avec les couleurs transparentes.

Remarque : Il est recommandé d'utiliser la nouvelle propriété `bgColor` à la place de la propriété `backColor`.

Exemple

L'instruction suivante donne à la variable `ancienneCouleur` la couleur d'arrière-plan de l'image-objet 5 :

```
-- Syntaxe Lingo
ancienneCouleur = sprite(5).backColor

// Syntaxe JavaScript
var ancienneCouleur = sprite(5).backColor;
```

L'instruction suivante modifie de façon aléatoire la couleur d'arrière-plan d'une image-objet choisie au hasard entre les image-objets 11 et 13 et lui affecte la couleur 36 :

```
-- Syntaxe Lingo
sprite(10 + random(3)).backColor = 36

// Syntaxe JavaScript
sprite(10 + random(3)).backColor = 36;
```

Voir aussi

Image-objet

backdrop

Utilisation

```
sprite(quelleImageObjet).camera{(index)}.backdrop[index].loc
member(quelActeur).camera(quelleCaméra).backdrop[index].loc
sprite(quelleImageObjet).camera{(index)}.backdrop[index].source
member(quelActeur).camera(quelleCaméra).backdrop[index].source
sprite(quelleImageObjet).camera{(index)}.backdrop[index].scale
member(quelActeur).camera(quelleCaméra).backdrop[index].scale
sprite(quelleImageObjet).camera{(index)}.backdrop[index].rotation
member(quelActeur).camera(quelleCaméra).\
  backdrop[index].rotation
sprite(quelleImageObjet).camera{(index)}.backdrop[index].regPoint
member(quelActeur).camera(quelleCaméra).\
  backdrop[index].regPoint
sprite(quelleImageObjet).camera{(index)}.backdrop[index].blend
member(quelActeur).camera(quelleCaméra).backdrop[index].blend
sprite(quelleImageObjet).camera{(index)}.backdrop.count
member(quelActeur).camera(quelleCaméra).backdrop.count
```

Description

Propriété 3D de caméra ; une image en 2D rendue sur le plan de projection de la caméra. Tous les modèles présents dans la vue de la caméra apparaissent devant le fond.

Les fonds ont les propriétés suivantes :

Remarque : Ces propriétés peuvent aussi être utilisées pour obtenir, définir et manipuler les recouvrements. Pour des informations détaillées, consultez les entrées des différentes propriétés.

`loc (fond et recouvrement)` indique l'emplacement 2D du fond, mesuré à partir du coin supérieur gauche de l'image-objet.

`source` indique la texture utilisée pour le fond.

`scale (fond et recouvrement)` est le nombre par lequel la hauteur et la largeur de la texture sont multipliées pour déterminer les dimensions du fond.

`rotation (fond et recouvrement)` est le nombre qui détermine la rotation du fond par rapport à son point d'alignement.

`regPoint (3D)` indique le point d'alignement du fond.

`blend (3D)` indique l'opacité du fond.

`count (3D)` indique le nombre d'éléments dans la liste de fonds de la caméra.

Utilisez les commandes suivantes pour créer et retirer des fonds :

`addBackdrop` crée un fond à partir d'une texture et l'ajoute à la fin de la liste des fonds de la caméra.

`insertBackdrop` crée un fond à partir d'une texture et l'ajoute à la liste des fonds de la caméra à une position d'index spécifique.

`removeBackdrop` supprime le fond.

Voir aussi

`overlay`

backgroundColor

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.backgroundColor

// Syntaxe JavaScript
réfObjActeur.backgroundColor;
```

Description

Propriété d'acteur forme vectorielle ; affecte à la couleur d'arrière-plan de l'acteur ou de l'image-objet spécifié la valeur de couleur RVB indiquée.

Cette propriété peut être testée et définie.

Exemple

```
-- Syntaxe Lingo
member("Archie").backgroundColor= color(255,255,255)

// Syntaxe JavaScript
member("Archie").backgroundColor= color(255,255,255);
```

Voir aussi

bgColor (fenêtre)

beepOn

Utilisation

```
-- Syntaxe Lingo
_movie.beepOn

// Syntaxe JavaScript
_movie.beepOn;
```

Description

Propriété d'animation ; détermine si l'ordinateur émet automatiquement un bip sonore lorsque l'utilisateur clique en dehors de toute image-objet active (TRUE) ou non (FALSE, valeur par défaut). Lecture/écriture.

Les scripts définissant la propriété beepOn devraient être placés dans les scripts d'images ou d'animations.

Exemple

L'instruction suivante donne à la propriété beepOn la valeur TRUE :

```
-- Syntaxe Lingo
_movie.beepOn = TRUE

// Syntaxe JavaScript
_movie.beepOn = true;
```

L'instruction suivante inverse la valeur de la propriété de `beepOn` :

```
-- Syntaxe Lingo
_movie.beepOn = not(_movie.beepOn)

// Syntaxe JavaScript
_movie.beepOn = !(_movie.beepOn);
```

Voir aussi

`Animation`

bevelDepth

Utilisation

```
member(quelActeurTexte).bevelDepth
member(quelActeur3D).modelResource(quelleRessourceDeModèle).\
  bevelDepth
```

Description

Propriété 3D de texte ; indique la taille du biseau du texte 3D.

Pour un acteur texte, cette propriété n'a aucun effet, sauf si sa propriété `displayMode` de l'acteur a pour valeur `#mode3D` et si sa propriété `bevelType` a pour valeur `#miter` ou `#round`.

Dans le cas d'un texte extrudé d'un acteur 3D, cette propriété n'a aucun effet, sauf si la propriété `bevelType` de la ressource de modèle a pour valeur `#miter` ou `#round`.

La plage de cette propriété s'étend de 0.0 à 10.0, la valeur par défaut étant 10.0.

Exemple

Dans l'exemple suivant, l'acteur Logo est un acteur texte. L'instruction suivante donne à la propriété `bevelDepth` de Logo la valeur 5.5. Lorsque l'acteur Logo est affiché en mode 3D, si sa propriété `bevelType` a pour valeur `#miter` ou `#round`, le bord de ses lettres sera fortement biseauté.

```
member("Logo").bevelDepth = 5.5
```

Dans l'exemple suivant, la ressource du modèle Slogan est du texte extrudé. L'instruction suivante donne à la propriété `bevelDepth` de la ressource de modèle Slogan la valeur 5. Si la propriété `bevelType` de Slogan a pour valeur `#miter` ou `#round`, les bords de ses lettres seront fortement biseautés.

```
member("Séquence").model("Slogan").resource.bevelDepth = 5
```

Voir aussi

`bevelType`, `extrude3D`, `displayMode`

bevelType

Utilisation

```
member(quelActeurTexte).bevelType
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  bevelType
```

Description

Propriété 3D de texte ; indique le style de biseau appliqué au texte 3D.

Pour les acteurs texte, il s'agit d'une propriété d'acteur. Pour le texte extrudé d'un acteur 3D, il s'agit d'une propriété de ressource de modèle.

La propriété `bevelType` a les valeurs possibles suivantes :

- `#none`
- `#miter` (chanfrein, la valeur par défaut)
- `#round`

Exemple

Dans l'exemple suivant, l'acteur Logo est un acteur texte. L'instruction suivante donne à la propriété `bevelType` de Logo la valeur `#round`.

```
member("Logo").beveltype = #round
```

Dans l'exemple suivant, la ressource du modèle Slogan est du texte extrudé. L'instruction suivante donne à la propriété `bevelType` de la ressource de modèle Slogan la valeur `#miter`.

```
member("Séquence").model("Slogan").resource.bevelType = #miter
```

Voir aussi

`bevelDepth`, `extrude3D`, `displayMode`

bgColor (fenêtre)

Utilisation

```
-- Syntaxe Lingo
réfObjFenêtre.bgColor

// Syntaxe JavaScript
réfObjFenêtre.bgColor;
```

Description

Propriété de fenêtre ; détermine la couleur d'arrière-plan d'une fenêtre. Lecture/écriture.

La sélection de la propriété `bgColor` revient au même que de définir la couleur dans la boîte de dialogue Propriétés de l'animation.

Exemple

L'exemple suivant donne à la couleur de la fenêtre intitulée Animaux une valeur RVB.

```
-- Syntaxe Lingo
window("Animaux").bgColor = color(255, 153, 0)

// Syntaxe JavaScript
window("Animaux").bgColor = color(255, 153, 0);
```

Voir aussi

Fenêtre

bgColor (image-objet, acteur 3D)

Utilisation

```
sprite(quelNuméroDimage).bgColor
the bgColor of sprite quelNuméroDimage
the bgColor of the stage
(the stage).bgColor
member(quelActeur3D).bgcolor
```

Description

Propriété d'image-objet, propriété système et propriété d'acteur 3D ; détermine la couleur d'arrière-plan de l'image-objet spécifiée par *quelNuméroDimage*, la couleur de la scène ou la couleur de fond de l'acteur 3D. La définition de la propriété d'image-objet `bgColor` revient au même que de choisir la couleur d'arrière-plan dans la fenêtre Outils, lorsque l'image-objet est sélectionnée sur la scène. La sélection de la propriété `bgColor` pour la scène revient au même que de définir la couleur dans la boîte de dialogue Propriétés de l'animation.

La propriété d'image-objet a une fonction équivalente à celle de `backColor`, mais la valeur de couleur renvoyée est un objet couleur du type défini pour cette image-objet.

Cette propriété peut être testée et définie.

Exemple

L'exemple suivant donne à la couleur de la scène une valeur RVB.

Syntaxe à points :

```
(the stage).bgColor = rgb(255, 153, 0)
```

Syntaxe verbose :

```
set the bgColor of the stage = rgb(255, 153, 0)
```

Voir aussi

color(), backColor, backgroundColor

bias

Utilisation

```
member(quelActeur).model(quelModèle).lod.bias
```

Description

Propriété de modificateur 3D `lod` ; indique dans quelle mesure le modificateur supprime les détails du modèle lorsque sa propriété `auto` a pour valeur `TRUE`. Cette propriété n'a aucun effet lorsque la propriété `auto` du modificateur a pour valeur `FALSE`.

La plage de cette propriété s'étend de 0.0 (qui supprime tous les polygones) à +100,0 (qui ne supprime aucun polygone). Le paramètre par défaut est 100.0.

Le modificateur `#lod` ne peut être ajouté qu'aux modèles créés dans un programme de modélisation 3D autre que Director. La valeur de la propriété `type` des ressources de modèle utilisées par ces modèles est `#fromFile`. Le modificateur ne peut pas être ajouté aux primitives créées dans Director.

Exemple

L'instruction suivante donne à la propriété `bias` du modificateur `lod` du modèle vaisseauSpatial la valeur 10. Si la propriété `auto` du modificateur `lod` a pour valeur `TRUE`, il réduira considérablement le niveau de détail de vaisseauSpatial au fur et à mesure que celui-ci s'éloignera de la caméra.

```
member("Univers 3D").model("vaisseauSpatial").lod.bias = 10
```

Voir aussi

`lod (modificateur)`, `auto`, `level`

bitmapSizes

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.bitmapSizes

// Syntaxe JavaScript
réfObjActeur.bitmapSizes;
```

Description

Propriété d'acteur police ; renvoie une liste des tailles de bitmap, en points, incluses lorsque l'acteur police a été créé.

Exemple

L'instruction suivante affiche les tailles de bitmap, en points, incluses lorsque l'acteur 11 a été créé :

```
-- Syntaxe Lingo
put(member(11).bitmapSizes)

// Syntaxe JavaScript
put(member(11).bitmapSizes);
```

Voir aussi

`recordFont`, `characterSet`, `originalFont`

bitRate

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.bitRate

// Syntaxe JavaScript
réfObjActeur.bitRate;
```

Description

Propriété d'acteur Shockwave Audio (SWA) ; renvoie le débit de téléchargement (en Kbps) de l'acteur SWA spécifié préchargé depuis le serveur.

La propriété d'acteur `bitRate` renvoie 0 tant que le transfert en flux continu n'a pas commencé.

Exemple

Le comportement suivant affiche le débit de téléchargement d'un acteur SWA lors de la première apparition de l'image-objet.

```
-- Syntaxe Lingo
property spriteNum

on beginSprite (me)
  acteurCourant = sprite(spriteNum).member.name
  put("Le débit de l'acteur"&&acteurCourant&&"est
  de"&&member(acteurCourant).bitRate)
end

// Syntaxe JavaScript
function beginSprite() {
  var acteurCourant = sprite(spriteNum).member.name;
  put("Le débit de l'acteur " + acteurCourant +" est de " +
  member(acteurCourant).bitRate);
}
```

bitsPerSample

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.bitsPerSample

// Syntaxe JavaScript
réfObjActeur.bitsPerSample;
```

Description

Propriété d'acteur Shockwave Audio (SWA) ; indique le codage du fichier d'origine qui a été encodé pour Shockwave Audio (SWA). Cette propriété n'est disponible qu'après le début de la lecture du son SWA ou après le préchargement du fichier avec la commande `preLoadBuffer`.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante affecte le codage d'origine du fichier utilisé pour l'acteur SWA en flux continu Paul Robin à l'acteur champ Codage :

```
-- Syntaxe Lingo
member("Codage").text = member("Paul Robin").bitsPerSample

// Syntaxe JavaScript
member("Codage").text = member("Paul Robin").bitsPerSample;
```

blend (3D)

Utilisation

```
sprite(quelleImageObjet).camera{(index)}.backdrop[index].blend
member(quelActeur).camera(quelleCaméra).backdrop[index].blend
sprite(quelleImageObjet).camera{(index)}.overlay[index].blend
member(quelActeur).camera(quelleCaméra).overlay[index].blend
member(quelActeur).shader(quelMatériau).blend
member(quelActeur).model(quelModèle).shader.blend
member(quelActeur).model(quelModèle).shaderList{[index]}.blend
```

Description

Propriété 3D de fond, de recouvrement et de matériau `#standard` ; indique l'opacité du fond, du recouvrement ou du matériau.

La définition de la propriété `blend` d'un matériau n'a aucun effet, sauf si sa propriété `transparent` a pour valeur `TRUE`.

La plage de cette propriété s'étend de 0 à 100, la valeur par défaut étant 100.

Exemple

L'instruction suivante donne à la propriété `blend` du matériau du modèle Fenêtre la valeur 80. Si la propriété `transparent` du matériau de Fenêtre a pour valeur `TRUE`, le modèle sera légèrement transparent.

```
member("Maison").model("Fenêtre").shader.blend = 80
```

Voir aussi

`bevelDepth`, `overlay`, `shadowPercentage`, `transparent`

blend (image-objet)

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.blend

// Syntaxe JavaScript
réfObjImageObjet.blend;
```

Description

Propriété d'image-objet ; renvoie ou définit la valeur d'opacité d'une image-objet, de 0 à 100, correspondant aux valeurs de la boîte de dialogue Propriétés de l'image-objet. Lecture/écriture.

Les couleurs possibles dépendent des couleurs disponibles dans la palette, quel que soit le codage de couleurs du moniteur.

Pour assurer des résultats optimaux, utilisez l'encre Opacité avec des images possédant un codage de couleurs supérieur à 8 bits.

Exemple

L'instruction suivante affecte une valeur d'opacité de 40% à l'image-objet 3.

```
-- Syntaxe Lingo
sprite(3).blend = 40

// Syntaxe JavaScript
sprite(3).blend = 40;
```

L'instruction suivante affiche la valeur d'opacité de l'image-objet 3 dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(sprite(3).blend)

// Syntaxe JavaScript
put(sprite(3).blend);
```

Voir aussi

blendLevel, Image-objet

blendConstant

Utilisation

```
member(quelActeur).shader(quelMatériau).blendConstant
member(quelActeur).model(quelModèle).shader.blendConstant
member(quelActeur).model(quelModèle).shaderList[[index]].\
  blendConstant
```

Description

Propriété 3D de matériau #standard ; indique le taux d'opacité utilisé pour la première couche de texture du matériau.

Si la propriété useDiffuseWithTexture du matériau a pour valeur TRUE, la texture se mélange à la couleur définie par la propriété diffuse du matériau. Si useDiffuseWithTexture a pour valeur FALSE, le blanc est utilisé pour l'opacité.

Chacune des autres couches de texture se mélange à la couche inférieure. Utilisez la propriété blendConstantList pour contrôler l'opacité dans ces couches de texture.

La propriété blendConstant ne fonctionne que lorsque la propriété blendSource du matériau a pour valeur #constant. Pour plus d'informations, consultez blendSource et blendSourceList.

La plage de cette propriété va de 0 à 100, la valeur par défaut étant 50.

Exemple

Dans l'exemple suivant, la liste des matériaux du modèle Mystère contient six matériaux. L'instruction suivante donne à la propriété `blendConstant` du second matériau la valeur 20. Cette propriété est affectée par les paramètres des propriétés `blendFunction`, `blendFunctionList`, `blendSource` et `blendSourceList`.

```
member("Niveau").model("Mystère").shaderList[2].\
  blendConstant = 20
```

Voir aussi

`blendConstantList`, `blendFunction`, `blendFunctionList`, `blendSource`, `blendSourceList`, `useDiffuseWithTexture`, `diffuse`, `diffuseColor`

blendConstantList

Utilisation

```
member(quelActeur).shader(quelMatériau).blendConstantList
member(quelActeur).model(quelModèle).shader.blendConstant\
  List[[index]]
member(quelActeur).model(quelModèle).shaderList[[index]].\
  blendConstantList[[index]]
```

Description

Propriété 3D de matériau `#standard` ; indique le taux utilisé pour fusionner une couche de texture du matériau avec la couche de texture inférieure.

La liste des textures et la liste des constantes de fusion du matériau doivent comporter huit positions d'index chacune. Chaque position d'index de la liste des constantes de fusion contrôle la fusion de la texture à la position d'index correspondante de la liste des textures. Vous pouvez donner la même valeur à toutes les positions d'index de la liste en ne spécifiant pas le paramètre *index* facultatif. Utilisez le paramètre *index* pour définir les positions d'index une par une.

La propriété `blendConstantList` ne fonctionne que lorsque la propriété `blendSource` de la couche de texture correspondante a pour valeur `#constant`.

La plage de cette propriété va de 0 à 100, la valeur par défaut étant 50.

Exemple

Dans l'exemple suivant, la liste des matériaux du modèle Mystère contient six matériaux. L'instruction suivante indique la propriété `blendConstant` de chacune des textures utilisées par le second matériau. Cette propriété est affectée par les paramètres des propriétés `blendFunction`, `blendFunctionList`, `blendSource` et `blendSourceList`.

```
put member("Niveau2").model("Mystère").shaderList[2].\
  blendConstantList
-- [20.0000, 50.0000, 50.0000, 50.0000, 20.0000, 50.0000, \
  50.0000, 50.0000]
```

Voir aussi

`blendConstant`, `blendFunction`, `blendFunctionList`, `blendSource`, `blendSourceList`, `useDiffuseWithTexture`, `diffuse`, `diffuseColor`

blendFactor

Utilisation

```
member(quelActeur).model(quelModèle).keyframePlayer.\
  blendFactor
member(quelActeur).model(quelModèle).bonesPlayer.blendFactor
```

Description

Propriété 3D de modificateur `keyframePlayer` et `bonesPlayer` ; indique le degré de combinaison d'un mouvement au mouvement qui le précède.

La plage de cette propriété s'étend de 0 à 100, la valeur par défaut étant 0.

`BlendFactor` n'est utilisé que lorsque la propriété `autoblend` du modificateur a pour valeur `FALSE`. Si la valeur de `blendFactor` est 100, le mouvement courant ne possédera aucune des caractéristiques du mouvement qui le précède. Si la valeur de `blendFactor` est 0, le mouvement courant possédera toutes les caractéristiques du mouvement qui le précède sans aucune valeur qui lui sont propres. Si la valeur de `blendFactor` est 50, le mouvement courant sera une synthèse également composée de ses propres caractéristiques et de celles du mouvement qui le précède. La valeur de `blendFactor` peut être modifiée pour créer des transitions, à la différence de la transition linéaire créée lorsque la propriété `autoblend` du modificateur a pour valeur `TRUE`.

Exemple

L'instruction suivante donne à la propriété `blendFactor` du modèle Martien3 la valeur 50. Si la propriété `autoblend` du modificateur est `FALSE`, chaque mouvement de la liste de lecture du `keyframePlayer` pour Martien3 sera une fusion, à part égale, du mouvement en cours et de celui qui le précède.

```
member("nouveauxMartiens").model("Martien3").keyframePlayer.blendFactor = 50
```

Voir aussi

`autoblend`, `keyframePlayer (modificateur)`

blendFunction

Utilisation

```
member(quelActeur).shader(quelMatériau).blendFunction
member(quelActeur).model(quelModèle).shader.blendFunction
member(quelActeur).model(quelModèle).shaderList{[index]}.\
  blendFunction
```

Description

Propriété 3D de matériau `#standard` ; indique le type de fusion utilisé pour la première couche de texture du matériau.

Si la propriété `useDiffuseWithTexture` du matériau a pour valeur `TRUE`, la texture se mélange à la couleur définie par la propriété `diffuse` du matériau. Si `useDiffuseWithTexture` a pour valeur `FALSE`, le blanc est utilisé pour l'opacité.

Chacune des autres couches de texture se mélange à la couche inférieure. Utilisez la propriété `blendFunctionList` pour contrôler l'opacité dans ces couches de texture.

La propriété `blendFunction` peut avoir les valeurs suivantes :

#multiply multiplie les valeurs RVB de la couche de texture par la couleur utilisée pour l'opacité (voir précédemment).

#add ajoute les valeurs RVB de la couche de texture à la couleur utilisée pour l'opacité, puis se fixe à 255.

#replace empêche la fusion de la texture avec la couleur définie par la propriété diffuse du matériau.

#blend combine les couleurs de la couche de texture avec la couleur utilisée pour la fusion selon le taux défini par la propriété blendConstant.

La valeur par défaut de cette propriété est #multiply.

Exemple

Dans l'exemple suivant, la liste des matériaux du modèle Mystère contient six matériaux. L'instruction suivante donne à la propriété blendFunction du second matériau la valeur #blend. Cela a pour effet d'activer les propriétés blendSource, blendSourceList, blendConstant et blendConstantList.

```
member("Niveau2").model("Mystère").shaderList[2].\
  blendFunction = #blend
```

Voir aussi

blendConstant, blendConstantList, blendFunctionList, blendSource, blendSourceList, useDiffuseWithTexture, diffuse, diffuseColor

blendFunctionList

Utilisation

```
member(quelActeur).shader(quelMatériau).\
  blendFunctionList{[index]}
member(quelActeur).model(quelModèle).shader.\
  blendFunctionList{[index]}
member(quelActeur).model(quelModèle).shaderList{[index]}.\
  blendFunctionList{[index]}
```

Description

Propriété 3D de matériau #standard ; liste linéaire indiquant la façon selon laquelle chaque couche de texture est fusionnée à la couche inférieure.

La liste des textures et la liste des fonctions de fusion du matériau doivent comporter chacune huit positions d'index. Chaque position d'index de la liste des fonctions de fusion contrôle la fusion de la texture à la position d'index correspondante de la liste de textures. Vous pouvez donner la même valeur à toutes les positions d'index de la liste en ne spécifiant pas le paramètre *index* facultatif. Utilisez le paramètre *index* pour définir les positions d'index une par une.

Chaque position d'index de la liste des fonctions de fusion peut avoir une des valeurs suivantes :

#multiply multiplie les valeurs RVB de la couche de texture par les valeurs RVB de la couche de texture inférieure.

#add ajoute les valeurs RVB de la couche de texture aux valeurs RVB de la couche de texture inférieure, puis se fixe à 255.

#replace a pour résultat que la texture recouvre la couche de texture inférieure. Aucune fusion n'a lieu.

`#blend` a pour résultat que la fusion est contrôlée par la valeur de la propriété `blendSource`, ce qui permet une fusion alpha.

La valeur par défaut de cette propriété est `#multiply`.

Exemple

Dans l'exemple suivant, la propriété `shaderList` du modèle Mystère contient six matériaux. L'instruction suivante indique que la valeur de la quatrième position d'index de la propriété `blendFunctionList` du second matériau a pour valeur `#blend`. La fusion de la quatrième couche de texture du deuxième matériau du modèle sera contrôlée par les paramètres des propriétés `blendSource`, `blendSourceList`, `blendConstant`, `blendConstantList`, `diffuse`, `diffuseColor` et `useDiffuseWithTexture`.

```
put member("Niveau2").model("Mystère").shaderList[2].\
  blendFunctionList[4]
-- #blend
```

Voir aussi

`blendConstant`, `blendConstantList`, `blendFunction`, `blendSource`, `blendSourceList`, `diffuse`, `diffuseColor`, `useDiffuseWithTexture`

blendLevel

Utilisation

```
sprite(quelNuméroDimage).blendLevel
the blendLevel of sprite quelNuméroDimage
```

Description

Propriété d'image-objet ; permet de définir la valeur d'opacité courante d'une image-objet ou d'y accéder. Les valeurs possibles sont comprises entre 0 et 255. Cette plage est différente de celle de l'inspecteur d'image-objet, qui est comprise entre 0 et 100. Cependant, elles ont toutes deux le même résultat, la seule différence concernant l'échelle des valeurs.

Cette propriété équivaut à la propriété d'image-objet `blend`.

Exemple

```
sprite(3).blendlevel = 99
```

Voir aussi

`blend (image-objet)`

blendRange

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle)\
   .blendRange.start
référenceDobjetDeRessourceDeModèle.blendRange.end
member(quelActeur).modelResource(quelleRessourceDeModèle)\
   .blendRange.start
référenceDobjetDeRessourceDeModèle.blendRange.end
```

Description

Propriété 3D ; utilisée avec une ressource de modèle de type `#particle`, cette propriété permet d'obtenir ou de définir le début et la fin de la plage d'opacité de la ressource.

L'opacité des particules du système est interpolée de façon linéaire entre `blendRange.start` et `blendRange.end` pendant toute la durée de vie de chaque particule.

La valeur de cette propriété doit toujours être supérieure ou égale à 0.0 et inférieure ou égale à 100.0. La valeur par défaut de cette propriété est 100,0.

Exemple

L'instruction suivante définit les propriétés `blendRange` de la ressource de modèle systèmeThermique, qui est du type `#particle`.

La première ligne donne la valeur de départ 100 et la seconde donne la valeur de fin 0. Cette instruction rend les particules de systèmeThermique complètement opaques lorsqu'elles apparaissent pour la première fois, puis elles s'estompent petit à petit jusqu'à devenir transparentes.

```
member("Chauffage").modelResource("systèmeThermique").blendRange.\
  start = 100.0
member("Chauffage").modelResource("systèmeThermique").blendRange.\
  end = 0.0
```

blendSource

Utilisation

```
member(quelActeur).shader(quelMatériau).blendSource
member(quelActeur).model(quelModèle).shader.blendSource
member(quelActeur).model(quelModèle).shaderList{[index]}.\
  blendSource
```

Description

Propriété 3D de matériau `#standard` ; indique si la fusion de la première couche de texture de la liste des textures du matériau est basée sur l'information alpha de la texture ou sur un taux constant.

Si la propriété `useDiffuseWithTexture` du matériau a pour valeur `TRUE`, la texture se mélange à la couleur définie par la propriété `diffuse` du matériau. Si `useDiffuseWithTexture` a pour valeur `FALSE`, le blanc est utilisé pour l'opacité.

Chacune des autres couches de texture se mélange à la couche inférieure. Utilisez la propriété `blendSourceList` pour contrôler l'opacité dans ces couches de texture.

La propriété `blendSource` ne fonctionne que lorsque la propriété `blendFunction` du matériau a pour valeur `#blend`.

Les valeurs possibles de cette propriété sont les suivantes :

`#alpha` a pour résultat que c'est l'information alpha de la texture qui détermine le taux d'opacité de chaque pixel de la texture avec la couleur utilisée (voir ci-dessus).

`#constant` a pour résultat que la valeur de la propriété `blendConstant` du matériau est utilisée comme taux d'opacité pour tous les pixels de la texture.

La valeur par défaut de cette propriété est `#constant`.

Exemple

Dans l'exemple suivant, la liste des matériaux du modèle Mystère contient six matériaux. L'instruction suivante donne à la propriété `blendSource` de la première texture utilisée par le second matériau la valeur `#constant`. Cela active les paramètres des propriétés `blendConstant` et `blendConstantList`.

```
member("Niveau2").model("Mystère").shaderList[2].\
  blendSource = #constant
```

Voir aussi

`blendSourceList`, `blendFunction`, `blendFunctionList`, `blendConstant`, `blendConstantList`, `useDiffuseWithTexture`, `diffuse`, `diffuseColor`

blendSourceList

Utilisation

```
member(quelActeur).shader(quelMatériau).\
  blendSourceList[index]
member(quelActeur).model(quelModèle).shader.\
  blendSourceList{[index]}
member(quelActeur).model(quelModèle).\
  shaderList{[index]}.blendSourceList{[index]}
```

Description

Propriété 3D de matériau `#standard` ; indique si la fusion d'une texture avec la texture inférieure est basée sur l'information alpha de la texture ou sur un taux constant.

La liste des textures et la liste des sources de fusion du matériau doivent comporter huit positions d'index chacune. Chaque position d'index de la liste des sources de fusion contrôle la fusion de la texture à la position d'index correspondante de la liste de textures. Vous pouvez donner la même valeur à toutes les positions d'index de la liste en ne spécifiant pas le paramètre *index* facultatif. Utilisez le paramètre *index* pour définir les positions d'index une par une.

La propriété `blendSourceList` ne fonctionne que lorsque la propriété `blendFunction` de la couche de texture correspondante a pour valeur `#blend`. Pour plus d'informations, consultez `blendFunction` et `blendFunctionList`.

Les valeurs possibles de cette propriété sont les suivantes :

`#alpha` fait que c'est l'information alpha de la texture qui détermine le taux d'opacité de chaque pixel de la couche de texture avec la couche inférieure.

`#constant` fait que la valeur de la propriété `blendConstant` de la couche de texture correspondante est utilisée comme taux de fusion pour tous les pixels de la couche de texture. Pour plus d'informations, consultez `blendConstant` et `blendConstantList`.

La valeur par défaut de cette propriété est `#constant`.

Exemple

Dans l'exemple suivant, la liste des matériaux du modèle Mystère contient six matériaux. Chaque matériau possède une liste de textures pouvant contenir jusqu'à huit textures. L'instruction suivante indique que la propriété `blendSource` de la quatrième texture utilisée par le second matériau a pour valeur `#constant`. Cela a pour effet d'activer les propriétés `blendConstant`, `blendConstantList` et `useDiffuseWithTexture`.

```
member("Niveau2").model("Mystère").shaderList[2].\
  blendSourceList[4] = #constant
```

Voir aussi

`blendSource`, `blendFunction`, `blendFunctionList`, `blendConstant`, `blendConstantList`, `useDiffuseWithTexture`, `diffuse`, `diffuseColor`

blendTime

Utilisation

```
member(quelActeur).model(quelModèle).keyframePlayer.\
  blendTime
member(quelActeur).model(quelModèle).bonesPlayer.blendTime
```

Description

Propriété 3D de modificateur `keyframePlayer` et `bonesPlayer` ; détermine la durée, en millisecondes, de la transition entre les mouvements de la liste de lecture du modificateur du modèle.

La propriété `blendTime` fonctionne en conjonction avec la propriété `autoBlend` du modificateur. Lorsque la propriété `autoBlend` a pour valeur `TRUE`, le modificateur crée une transition linéaire entre le mouvement courant du modèle et le mouvement précédent. La valeur de la propriété `blendTime` est la durée de cette transition. La propriété `blendTime` est ignorée lorsque `autoBlend` a pour valeur `FALSE`.

Le paramètre par défaut de cette propriété est 500.

Exemple

L'instruction suivante définit la durée de la transition entre les mouvements de la liste de lecture du modificateur du modèle Martien5 à 1200 millisecondes.

```
member("nouveauxMartiens").model("Martien5").keyframePlayer.\
  blendTime = 1200
```

Voir aussi

`autoblend`, `blendFactor`

bone

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  bone.count
member(quelActeur).model(quelModèle).bonesPlayer.\
  bone[index].transform
member(quelActeur).model(quelModèle).bonesPlayer.\
  bone[index].worldTransform
```

Description

Elément 3D ; un segment est un élément structurel d'une ressource de modèle créée dans un programme de modélisation 3D. Les segments ne peuvent pas être créés, supprimés ou réarrangés dans Director.

Les mouvements #bones, qui doivent aussi être programmés dans un programme de modélisation 3D, agissent sur la structure des segments d'une ressource de modèle et sont gérés dans Director par le modificateur bonesPlayer.

Voir aussi

count (3D), bonesPlayer (modificateur), transform (propriété), worldTransform

bonesPlayer (modificateur)

Utilisation

```
member(quelActeur).model(quelModèle).
  bonesPlayer.quellePropriétéDeModifBonesPlayer
```

Description

Modificateur 3D ; gère l'utilisation des mouvements par les modèles. Les mouvements gérés par le modificateur bonesPlayer animent les segments du modèle.

Les mouvements et les modèles qui les utilisent doivent être créés dans un programme de modélisation 3D, exportés au format *.w3d, puis importés dans une animation. Les mouvements ne peuvent pas être appliqués aux primitives de modèle créées dans Director.

L'ajout du modificateur bonesPlayer à un modèle à l'aide de la commande addModifier permet d'accéder aux propriétés suivantes du modificateur bonesPlayer :

playing (3D) indique le mouvement d'un modèle.

playlist est une liste linéaire de listes de propriétés contenant les paramètres de lecture des mouvements d'un modèle en file d'attente.

currentTime (3D) indique la position, en millisecondes, du mouvement en cours de lecture ou en pause.

playRate (3D) est un nombre multiplié par le paramètre *échelle* de la commande play() ou queue() pour déterminer la cadence de lecture du mouvement.

playlist.count (3D) renvoie le nombre de mouvements en file d'attente dans la liste de lecture.

rootLock indique si le composant de translation du mouvement est utilisé ou ignoré.

currentLoopState indique si le mouvement est lu une seule fois ou répété de façon continue.

`blendTime` indique la durée de la transition créée par le modificateur entre les mouvements lorsque la propriété `autoblend` a pour valeur `TRUE`.

`autoblend` indique si le modificateur crée une transition linéaire entre le mouvement en cours de lecture et le mouvement précédent.

`blendFactor` indique le degré de fusion entre les mouvements lorsque la propriété `autoBlend` a pour valeur `FALSE`.

`bone[IdDeSegment].transform` indique la transformation du segment par rapport au segment parent. Vous pouvez trouver la valeur de `IdDeSegment` en testant la propriété `getBoneID` de la ressource de modèle. Lorsque vous définissez la transformation du segment, il n'est plus contrôlé par le mouvement en cours. Le contrôle manuel s'arrête avec le mouvement en cours.

`bone[IdDeSegment].getWorldTransform` renvoie la transformation du segment par rapport à l'univers.

`lockTranslation` indique si le modèle peut être déplacé à partir des plans spécifiés.

`positionReset` indique si le modèle retourne à sa position de départ à la fin du mouvement ou de chaque itération d'une boucle.

`rotationReset` indique l'élément de rotation d'une transition d'un mouvement à un autre ou de la boucle d'un seul mouvement.

Remarque : Pour plus d'informations, consultez les entrées des différentes propriétés.

Le modificateur `bonesPlayer` utilise les commandes suivantes :

`pause() (3D)` stoppe le mouvement du modèle en cours d'exécution.

`play() (3D)` entraîne ou reprend l'exécution d'un mouvement.

`playNext() (3D)` entraîne la lecture du mouvement suivant de la liste de lecture.

`queue() (3D)` ajoute un mouvement à la fin de la liste de lecture.

Le modificateur `bonesPlayer` génère les événements suivants, qui sont utilisés par les gestionnaires déclarés dans les commandes `registerForEvent()` et `registerScript()`. L'appel au gestionnaire déclaré contient trois arguments : le type d'événement (`#animationStarted` ou `#animationEnded`), le nom du mouvement, ainsi que sa position. Pour plus d'informations sur les événements de notification, consultez l'entrée de `registerForEvent()`.

`#animationStarted` est envoyé au début de la lecture d'un mouvement. Si la fusion est utilisée entre des mouvements, l'événement est envoyé au début de la transition.

`#animationEnded` est envoyé à la fin d'un mouvement. Si la fusion est utilisée entre des mouvements, l'événement est envoyé à la fin de la transition.

Voir aussi

`keyframePlayer (modificateur)`, `addModifier`, `modifiers`, `modifier`

border

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.border

// Syntaxe JavaScript
réfObjActeur.border;
```

Description

Propriété d'acteur champ ; indique l'épaisseur, en pixels, du cadre entourant l'acteur champ spécifié.

Exemple

L'instruction suivante donne au cadre de l'acteur champ Titre une épaisseur de 10 pixels :

```
-- Syntaxe Lingo
member("Titre").border = 10

// Syntaxe JavaScript
member("Titre").border = 10;
```

bottom

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.bottom

// Syntaxe JavaScript
réfObjImageObjet.bottom;
```

Description

Propriété d'image-objet ; spécifie la coordonnée verticale du bord inférieur du rectangle de délimitation d'une image-objet. Lecture/écriture.

Exemple

L'instruction suivante affecte la coordonnée verticale de la partie inférieure de l'image-objet numérotée (i + 1) à la variable `plusbas`.

```
-- Syntaxe Lingo
plusbas = sprite(i + 1).bottom

// Syntaxe JavaScript
var plusbas = sprite(i + 1).bottom;
```

Voir aussi

Image-objet

bottom (3D)

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).bottom
```

Description

Propriété 3D de la ressource de modèle #box ; indique si le côté de la boîte coupé par son axe des y négatif est fermé (TRUE) ou ouvert (FALSE).

La valeur par défaut de cette propriété est TRUE.

Exemple

L'instruction suivante donne à la propriété bottom de la ressource de modèle boîteAcadeau la valeur TRUE, ce qui signifie que le fond de cette boîte sera fermé.

```
member("Univers 3D").modelResource("boîteAcadeau").bottom = TRUE
```

Voir aussi

back, front, top (3D), left (3D), right (3D), bottomCap

bottomCap

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  bottomCap
```

Description

Propriété 3D de ressource de modèle #cylinder ; indique si le fond du cylindre coupé par son axe des y négatif est fermé (TRUE) ou ouvert (FALSE).

La valeur par défaut de cette propriété est TRUE.

Exemple

L'instruction suivante donne à la propriété bottomCap de la ressource de modèle Tube la valeur FALSE, ce qui signifie que le fond de ce cylindre sera ouvert.

```
member("Univers 3D").modelResource("tube").bottomCap = FALSE
```

Voir aussi

topCap, bottomRadius, bottom (3D)

bottomRadius

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  bottomRadius
```

Description

Propriété 3D de ressource de modèle #cylinder ; indique le rayon, en unités de l'univers, de l'extrémité du cylindre à l'intersection de son axe des y négatif.

La valeur par défaut de cette propriété est 25.

Exemple

L'instruction suivante donne à la propriété `bottomRadius` de la ressource de modèle Tube la valeur 38.5.

```
member("Univers 3D").modelResource("tube").bottomRadius = 38.5
```

Voir aussi

`topRadius`, `bottomCap`

bottomSpacing

Utilisation

```
-- Syntaxe Lingo
expressionDeSousChaîne.bottomSpacing

// Syntaxe JavaScript
expressionDeSousChaîne.bottomSpacing;
```

Description

Propriété d'acteur texte ; permet de spécifier tout espacement supplémentaire applicable au bas de chaque paragraphe dans la partie *expressionDeSousChaîne* de l'acteur texte.

La valeur même est un entier, qui indique un espacement moindre entre les paragraphes s'il est inférieur à 0 et un espacement plus important s'il est supérieur à 0.

La valeur par défaut est 0 ; elle correspond à l'espacement par défaut entre les paragraphes.

Remarque : Cette propriété, comme toutes les propriétés d'acteur texte, ne supporte que la syntaxe à points.

Exemple

Dans l'exemple suivant, une espace est ajoutée après le premier paragraphe de l'acteur texte Nouvelles du jour.

```
-- Syntaxe Lingo
member("Nouvelles du jour").paragraph[1].bottomSpacing=20

// Syntaxe JavaScript
member("Nouvelles du jour").getPropRef("paragraph", 1).bottomSpacing=20;
```

Voir aussi

`top (3D)`

boundary

Utilisation

```
member(quelActeur).model(quelModèle).inker.boundary
member(quelActeur).model(quelModèle).toon.boundary
```

Description

Propriété 3D de modificateur `inker` et `toon` ; permet de définir si une ligne est tracée aux bords d'un modèle.

Le paramètre par défaut de cette propriété est `TRUE`.

Exemple

L'instruction suivante donne à la propriété `boundary` du modificateur `inker` appliqué au modèle Boîte la valeur `TRUE`. Les lignes seront tracées aux bords de la surface du modèle.

```
member("formes").model("Boîte").inker.boundary = TRUE
```

Voir aussi

`lineColor`, `lineOffset`, `silhouettes`, `creases`

boundingSphere

Utilisation

```
member(quelActeur).model(quelModèle).boundingSphere
member(quelActeur).group(quelGroupe).boundingSphere
member(quelActeur).light(quelleLumière).boundingSphere
member(quelActeur).camera(quelleCaméra).boundingSphere
```

Description

Propriété 3D de modèle, de groupe, de lumière et de caméra ; décrit une sphère contenant le modèle, le groupe, la lumière ou la caméra et ses enfants.

La valeur de cette propriété est une liste contenant la position vectorielle du centre de la sphère et la longueur, exprimée en valeur à virgule flottante, du rayon de la sphère.

Cette propriété peut être testée, mais pas définie.

Exemple

L'exemple suivant affiche la sphère de délimitation d'une lumière dans la fenêtre Messages.

```
put member("nouveauMartien").light[5].boundingSphere
-- [vector( 166,8667, -549,6362, 699,5773 ), 1111,0039]
```

Voir aussi

`debug`

boxDropShadow

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.boxDropShadow

// Syntaxe JavaScript
réfObjActeur.boxDropShadow;
```

Description

Propriété d'acteur ; détermine la taille, en pixels, de l'ombre portée du cadre de l'acteur champ spécifié par *quelActeur*.

Exemple

L'instruction suivante donne à l'ombre portée de l'acteur champ Titre une largeur de 10 pixels :

```
-- Syntaxe Lingo
member("Titre").boxDropShadow = 10

// Syntaxe JavaScript
member("Titre").boxDropShadow = 10;
```

boxType

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.boxType

// Syntaxe JavaScript
réfObjActeur.boxType;
```

Description

Propriété d'acteur ; détermine le type de zone de texte utilisé pour l'acteur spécifié. Les valeurs possibles sont `#adjust` (ajustable), `#scroll` (défilant), `#fixed` (fixe) et `#limit` (limité).

Exemple

L'instruction suivante fait du cadre de l'acteur champ Editorial un champ défilant.

```
-- Syntaxe Lingo
member("Editorial").boxType = #scroll

// Syntaxe JavaScript
member("Editorial").boxType = symbol("scroll");
```

brightness

Utilisation

```
member(quelActeur).shader(quelMatériau).brightness
member(quelActeur).model(quelModèle).shader.brightness
member(quelActeur).model(quelModèle).shaderList[[index]].\
  brightness
```

Description

Propriété 3D de matériau `#newsprint` et `#engraver` ; indique la quantité de blanc fusionnée au matériau.

La plage de cette propriété va de 1 à 100, la valeur par défaut étant 0.

Exemple

L'instruction suivante définit la luminosité du matériau utilisé par le modèle gbCyl2 à la moitié de sa valeur maximum.

```
member("séquence").model("gbCyl2").shader.brightness = 50
```

Voir aussi

`newShader`

broadcastProps

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.broadcastProps

// Syntaxe JavaScript
réfObjActeur.broadcastProps;
```

Description

Propriété d'acteur ; contrôle si les modifications apportées à un acteur Flash ou forme vectorielle sont immédiatement transmises à toutes ses images-objets présentes sur la scène (`TRUE`) ou non (`FALSE`).

Lorsque cette propriété a pour valeur `FALSE`, les modifications apportées à l'acteur sont utilisées comme valeurs par défaut pour les nouvelles images-objets et n'affectent pas les images-objets sur la scène.

La valeur par défaut de cette propriété est `TRUE` et elle peut être testée et définie.

Exemple

Le script d'image suivant suppose que l'acteur Animation de navigation d'une animation Flash utilise une propriété `broadcastProps` possédant la valeur `FALSE`. Il permet provisoirement de modifier un acteur animation Flash à diffuser aux images-objets placées sur la scène. Il définit ensuite la propriété `viewScale` de l'acteur animation Flash et cette modification est transmise à son image-objet. Le script interdit alors à l'animation Flash de diffuser les modifications ultérieures à ses images-objets.

```
-- Syntaxe Lingo
on enterFrame
  member("Animation de navigation").broadcastProps = TRUE
  member("Animation de navigation").viewScale = 200
  member("Animation de navigation").broadcastProps = FALSE
end

// Syntaxe JavaScript
function enterFrame() {
  member("Animation de navigation").broadcastProps = 1;
  member("Animation de navigation").viewScale = 200;
  member("Animation de navigation").broadcastProps = 0;
}
```

bufferSize

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.bufferSize

// Syntaxe JavaScript
réfObjActeur.bufferSize;
```

Description

Propriété d'acteur Flash ; contrôle le nombre d'octets d'une animation Flash liée qui sont passés en mémoire en une seule fois. La propriété `bufferSize` peut uniquement avoir comme valeur un nombre entier. Cette propriété produit uniquement un effet lorsque la propriété `preload` de l'acteur a pour valeur `FALSE`.

Cette propriété peut être testée et définie. La valeur par défaut est 32 768 octets.

Exemple

Le gestionnaire `startMovie` suivant définit un acteur animation Flash pour une lecture en flux continu puis définit sa propriété `bufferSize`.

```
-- Syntaxe Lingo
on startMovie
  member.("Démo Flash").preload = FALSE
  member.("Démo Flash").bufferSize = 65536
end

// Syntaxe JavaScript
function startMovie() {
  member.("Démo Flash").preload = 0;
  member.("Démo Flash").bufferSize = 65536;
}
```

Voir aussi

`bytesStreamed`, `preLoadRAM`, `stream()`, `streamMode`

buttonCount

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.buttonCount

// Syntaxe JavaScript
réfObjDvd.buttonCount;
```

Description

Propriété DVD ; renvoie le nombre de boutons disponibles dans le menu DVD courant. En lecture seule.

Voir aussi

`DVD`

buttonsEnabled

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.buttonsEnabled

// Syntaxe JavaScript
réfObjActeurOuImageObjet.buttonsEnabled;
```

Description

Propriété d'acteur Flash et propriété d'image-objet ; contrôle si les boutons d'une animation Flash sont actifs (`TRUE`, valeur par défaut) ou inactifs (`FALSE`). Les actions de bouton sont uniquement déclenchées lorsque la propriété `actionsEnabled` reçoit la valeur `TRUE`.

Cette propriété peut être testée et définie.

Exemple

Le gestionnaire suivant accepte une référence d'image-objet et permet d'activer ou de désactiver la propriété `buttonsEnabled` de l'image-objet.

```
-- Syntaxe Lingo
on ToggleButtons(quelleImageObjet)
  sprite(quelleImageObjet).buttonsEnabled = \
  not(sprite(quelleImageObjet).buttonsEnabled)
end

// Syntaxe JavaScript
function ToggleActions(quelleImageObjet) {
  sprite(quelleImageObjet).buttonsEnabled =
  !(sprite(quelleImageObjet).buttonsEnabled);
}
```

Voir aussi

`actionsEnabled`

buttonStyle

Utilisation

```
-- Syntaxe Lingo
_movie.buttonStyle

// Syntaxe JavaScript
_movie.buttonStyle;
```

Description

Propriété d'animation ; détermine la réponse visuelle des boutons alors que le bouton de la souris est enfoncé. Lecture/écriture.

Elle s'applique uniquement aux boutons créés avec l'outil Bouton de la palette des outils.

La propriété `buttonStyle` peut avoir les valeurs suivantes :

- 0 (style de liste : valeur par défaut) – Les boutons suivants sont mis en surbrillance lorsque le pointeur les survole. Si l'utilisateur relâche le bouton de la souris, le script associé à ce dernier est activé.

- `1` (style de boîte de dialogue)—Seul le premier bouton sur lequel l'utilisateur clique est mis en surbrillance. Les boutons suivants ne le sont pas. Si l'utilisateur relâche le bouton de la souris alors que le pointeur survole un bouton autre que celui sur lequel il a cliqué initialement, le script associé à ce bouton n'est pas activé.

Exemple

L'instruction suivante donne à la propriété `buttonStyle` la valeur 1 :

```
-- Syntaxe Lingo
_movie.buttonStyle = 1

// Syntaxe JavaScript
_movie.buttonStyle = 1;
```

L'instruction suivante permet de mémoriser le paramètre courant de la propriété `buttonStyle` en le stockant dans la variable `valeurDeStyleDeBouton` :

```
-- Syntaxe Lingo
valeurDeStyleDeBouton = _movie.buttonStyle

// Syntaxe JavaScript
var valeurDeStyleDeBouton = _movie.buttonStyle;
```

Voir aussi

`Animation`

buttonType

Utilisation

```
member(quelActeur).buttonType
the buttonType of member quelActeur
```

Description

Propriété d'acteur bouton ; indique le type de l'acteur bouton spécifié. Les valeurs possibles sont `#pushButton` (bouton-poussoir), `#checkBox` (case à cocher) et `#radioButton` (bouton radio). Cette propriété s'applique uniquement aux boutons créés avec l'outil Bouton de la palette des outils.

Exemple

L'instruction suivante transforme l'acteur bouton Editorial en case à cocher :

Syntaxe à points :

```
member("Editorial").buttonType = #checkBox
```

Syntaxe verbose :

```
set the buttonType of member "Editorial" to #checkBox
```

bytesStreamed

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.bytesStreamed

// Syntaxe JavaScript
réfObjActeur.bytesStreamed;
```

Description

Propriété d'acteur Flash et Shockwave Audio ; indique le nombre d'octets de l'acteur spécifié qui ont été chargés en mémoire. La propriété `bytesStreamed` renvoie une valeur uniquement pendant la lecture de l'animation Director. Elle renvoie un nombre entier.

Cette propriété peut être testée, mais pas définie.

Exemple

Le gestionnaire suivant accepte une référence d'acteur en tant que paramètre, puis utilise la commande `stream` pour charger l'acteur en mémoire. A chaque fois qu'il transfère une partie de l'acteur en mémoire, il utilise la propriété `bytesStreamed` pour indiquer dans la fenêtre Messages le nombre d'octets transmis.

```
-- Syntaxe Lingo
on téléchargerLanimation(quelleAnimationFlash)
  repeat while member(quelleAnimationFlash).percentStreamed < 100
    stream(member(quelleAnimationFlash))
    put("Nombre d'octets transmis :" &&
  member(quelleAnimationFlash).bytesStreamed)
  end repeat
end

// Syntaxe JavaScript
function téléchargerLanimation(quelleAnimationFlash)
  var i = member(quelleAnimationFlash).percentStreamed;
  while (i < 100) {
    stream(member(quelleAnimationFlash));
    trace("Nombre d'octets transmis : " +
  member(quelleAnimationFlash).bytesStreamed);
  }
}
```

Voir aussi

`bufferSize`, `percentStreamed (acteur)`, `stream()`

bytesStreamed (3D)

Utilisation

```
member(quelActeur).bytesStreamed
```

Description

Propriété 3D d'acteur ; indique la quantité qui a été chargée du fichier initial ou du dernier fichier demandé.

Exemple

L'instruction suivante indique que 325 300 octets de l'acteur Séquence ont été chargés.

```
put member("Séquence").bytesStreamed
-- 325300
```

Voir aussi

streamSize (3D), state (3D)

camera

Utilisation

```
member(quelActeur).camera(quelleCaméra)
member(quelActeur).camera[index]
member(quelActeur).camera(quelleCaméra).quellePropriétéDeCaméra
member(quelActeur).camera[index].quellePropriétéDeCaméra
sprite(quelleImageObjet).camera{(index)}
sprite(quelleImageObjet).camera{(index)}.quellePropriétéDeCaméra
```

Description

Elément 3D ; objet à une position de vecteur à partir de laquelle l'univers 3D est observé.

Chaque image-objet possède une liste de caméras. Les vues des différentes caméras de la liste sont affichées au-dessus de celles des caméras en position *index* inférieures. Vous pouvez définir la propriété rect (caméra) de chaque caméra afin d'afficher plusieurs vues au sein de l'image-objet.

Les caméras sont enregistrées dans la palette des caméras de l'acteur. Utilisez les commandes newCamera et deleteCamera pour créer et supprimer les caméras d'un acteur 3D.

La propriété camera d'une image-objet est la première caméra de la liste des caméras de l'image-objet. La caméra référencée par sprite(*quelleImageObjet*).camera est la même que sprite(*quelleImageObjet*).camera(1). Utilisez les commandes addCamera et deleteCamera pour construire la liste des caméras d'une image-objet 3D.

Vous trouverez une liste complète des propriétés et commandes de caméra dans la rubrique Utilisation de Director du panneau d'aide de Director.

Exemple

L'instruction suivante affecte à l'image-objet 1 la caméra camArbre de l'acteur Picnic.

```
sprite(1).camera = member("Picnic").camera("camArbre")
```

L'instruction suivante affecte à l'image-objet 1 la caméra 2 de l'acteur Picnic.

```
sprite(1).camera = member("Picnic").camera[2]
```

Voir aussi

bevelDepth, overlay, modelUnderLoc, spriteSpaceToWorldSpace, fog, clearAtRender

cameraPosition

Utilisation

```
member(quelActeur).cameraPosition
sprite(quelleImageObjet).cameraPosition
```

Description

Propriété 3D d'acteur et d'image-objet ; indique la position de la caméra par défaut.

La valeur par défaut de cette propriété est `vector(0, 0, 250)`. Il s'agit de la position de la caméra par défaut dans le nouvel acteur 3D.

Exemple

L'instruction suivante indique que la position de la caméra par défaut de l'acteur LileAuxEnfants est `vector(-117.5992, -78.9491, 129.0254)`.

```
member("LileAuxEnfants").cameraPosition = vector(-117.5992, \
  -78.9491, 129.0254)
```

Voir aussi

`cameraRotation`, `autoCameraPosition`

cameraRotation

Utilisation

```
member(quelActeur).cameraRotation
sprite(quelleImageObjet).cameraRotation
```

Description

Propriété 3D d'acteur et d'image-objet ; indique la position de la caméra par défaut.

La valeur par défaut de cette propriété est `vector(0, 0, 0)`. Il s'agit de la rotation de la caméra par défaut dans le nouvel acteur 3D.

Exemple

L'instruction suivante indique que la rotation de la caméra par défaut de l'acteur lileAuxEnfants est `vector(82.6010, -38.8530, -2.4029)`.

```
member("lileAuxEnfants").cameraRotation = vector(82.6010, \
  -38.8530, -2.4029)
```

Voir aussi

`cameraPosition`, `autoCameraPosition`

castLib

Utilisation

```
-- Syntaxe Lingo
_movie.castLib[nomOuNumDeDistribution]

// Syntaxe JavaScript
_movie.castLib[nomOuNumDeDistribution];
```

Description

Propriété d'animation ; fournit un accès nommé ou indexé aux bibliothèques de distributions d'une animation, que celle-ci soit active ou non. En lecture seule.

L'argument *nomOuNumDeDistribution* peut être une chaîne spécifiant le nom de l'animation à laquelle accéder ou un entier spécifiant le numéro de l'animation à laquelle accéder.

Cette propriété fournit des fonctions similaires à la méthode `castLib()` de niveau supérieur, cette dernière ne s'appliquant toutefois qu'à l'animation en cours de projection.

Exemple

L'instruction suivante affiche le numéro de la distribution Boutons dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(_movie.castLib["Boutons"].number)

// Syntaxe JavaScript
put(_movie.castLib["Boutons"].number);
```

Voir aussi

`castLib()`, `Animation`

castLibNum

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.castLibNum

// Syntaxe JavaScript
réfObjActeur.castLibNum;
```

Description

Propriété d'acteur ; détermine le numéro de la bibliothèque de distribution à laquelle appartient un acteur. En lecture seule.

Exemple

L'instruction suivante détermine le numéro de la distribution à laquelle l'acteur Jazz est affecté :

```
-- Syntaxe Lingo
put(member("Jazz").castLibNum)

// Syntaxe JavaScript
put(member("Jazz").castLibNum);
```

L'instruction suivante modifie l'acteur affecté à l'image-objet 5 en remplaçant sa distribution par la distribution Mercredi :

```
-- Syntaxe Lingo
sprite(5).castLibNum = castLib("Mercredi").number

// Syntaxe JavaScript
sprite(5).castLibNum = castLib("Mercredi").number;
```

Voir aussi

Bibliothèque de distribution, Acteur

castMemberList

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.castMemberList

// Syntaxe JavaScript
réfObjActeur.castMemberList;
```

Description

Propriété d'acteur curseur ; spécifie la liste des acteurs composant les images d'un curseur. Remplacez *quelActeurCurseur* par le nom (entre guillemets) ou le numéro d'un acteur. Vous pouvez spécifier des acteurs appartenant à différentes distributions.

Le premier acteur de la liste est la première image du curseur, le deuxième acteur est la deuxième image, et ainsi de suite.

Si vous spécifiez des acteurs impossibles à utiliser dans un curseur, ils sont ignorés et les acteurs restants sont utilisés.

Cette propriété peut être testée et définie.

Exemple

La commande suivante crée une série de quatre acteurs pour l'acteur curseur couleur animé monCurseur.

```
-- Syntaxe Lingo
member("monCurseur").castmemberList = \
  [member(1), member(2), member(1, 2), member(2, 2)]

// Syntaxe JavaScript
member("monCurseur").castmemberList =
  list(member(1), member(2), member(1, 2), member(2, 2));
```

center

Utilisation

```
member(quelActeur).center
the center of member quelActeur
```

Description

Propriété d'acteur ; interagit avec la propriété d'acteur `crop`.

- Lorsque la propriété `the crop` est `FALSE`, la propriété `center` n'a aucun effet.

- Lorsque `crop` est `TRUE` et `center` est `TRUE`, un découpage se produit autour du centre de l'acteur vidéo numérique.
- Lorsque `crop` est `TRUE` et `center` est `FALSE`, un découpage se produit sur les côtés droit et inférieur de la vidéo numérique.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante provoque l'affichage, dans le coin supérieur gauche de l'image-objet, de l'acteur vidéo numérique Interview :

Syntaxe à points :

```
member("Interview").center = FALSE
```

Syntaxe verbose :

```
set the center of member "Interview" to FALSE
```

Voir aussi

`crop`, `centerRegPoint`, `regPoint`, `scale (acteur)`

centerRegPoint

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.centerRegPoint

// Syntaxe JavaScript
réfObjActeur.centerRegPoint;
```

Description

Propriété d'acteur Flash, forme vectorielle et bitmap ; centre automatiquement le point d'alignement de l'acteur lorsque vous redimensionnez l'image-objet (`TRUE`, valeur par défaut) ou sert également à repositionner le point d'alignement à sa valeur courante lorsque vous redimensionnez l'acteur ou que vous définissez la propriété `defaultRect` ou `regPoint` (`FALSE`).

Cette propriété peut être testée et définie.

Exemple

Le script suivant vérifie si la propriété `centerRegPoint` d'une animation Flash a pour valeur `TRUE`. Le cas échéant, le script utilise la propriété `regPoint` pour repositionner le point d'alignement de l'image-objet dans son coin supérieur gauche. Lorsqu'il vérifie la propriété `centerRegPoint`, le script s'assure qu'il ne repositionne pas un point d'alignement précédemment défini au moyen de la propriété `regPoint`.

```
-- Syntaxe Lingo
property spriteNum

on beginSprite me
  if sprite(spriteNum).member.centerRegPoint = TRUE then
    sprite(spriteNum).member.regPoint = point(0,0)
  end if
end
```

```
// Syntaxe JavaScript
function beginSprite() {
  var ctrRg = sprite(this.spriteNum).member.centerRegPoint;
  if (ctrRg = 1) {
    sprite(this.spriteNum).member.regPoint = point(0,0);
  }
}
```

Voir aussi

regPoint

centerStage

Utilisation

```
-- Syntaxe Lingo
_movie.centerStage

// Syntaxe JavaScript
_movie.centerStage;
```

Description

Propriété d'animation ; détermine si la scène est au centre du moniteur lors du chargement de l'animation (TRUE, valeur par défaut) ou non (FALSE). Lecture/écriture.

Placez l'instruction incluant cette propriété dans l'animation précédant celle qui doit être affectée.

Cette propriété est utile pour vérifier l'emplacement de la scène avant de lire une projection.

Remarque : Veuillez noter que le comportement constaté pendant la lecture d'une projection peut varier suivant que vous utilisez un système Windows ou Macintosh. Les paramètres sélectionnés pendant la création de la projection peuvent remplacer cette propriété.

Exemple

L'instruction suivante déplace l'animation vers une image spécifique si la scène n'est pas centrée :

```
-- Syntaxe Lingo
if (_movie.centerStage = FALSE) then
  _movie.go("Décalé")
end if

// Syntaxe JavaScript
if (_movie.centerStage == false) {
  _movie.go("Décalé");
}
```

L'instruction suivante inverse la valeur courante de la propriété centerStage :

```
-- Syntaxe Lingo
_movie.centerStage = not(_movie.centerStage)

// Syntaxe JavaScript
_movie.centerStage = !(_movie.centerStage)
```

Voir aussi

fixStageSize, Animation

changeArea

Utilisation

```
member(quelActeur).changeArea
the changeArea of member quelActeur
```

Description

Propriété d'acteur transition ; détermine si une transition doit s'appliquer uniquement au secteur modifié de la scène (TRUE) ou à la scène entière (FALSE). Son effet est semblable à celui de l'option Zone modifiée seulement de la boîte de dialogue Propriétés de l'image : Transition.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante permet à l'acteur transition Vague de ne s'appliquer qu'au secteur modifié de la scène.

Syntaxe à points :

```
member("Vague").changeArea = TRUE
```

Syntaxe verbose :

```
set the changeArea of member "Vague" to TRUE
```

channelCount

Utilisation

```
-- Syntaxe Lingo
réfObjPisteAudio.channelCount

// Syntaxe JavaScript
réfObjPisteAudio.channelCount;
```

Description

Propriété de piste audio ; détermine le nombre de pistes du son en cours de lecture ou en pause dans une piste audio spécifiée. En lecture seule.

Cette propriété est utile pour savoir si un son est mono ou stéréo.

Exemple

L'instruction suivante détermine le nombre de pistes contenues dans l'acteur son Jazz.

```
-- Syntaxe Lingo
put(member("Jazz").channelCount)

// Syntaxe JavaScript
put(member("Jazz").channelCount);
```

L'instruction suivante détermine le nombre de pistes contenues dans l'acteur son Jazz lu dans la piste 2.

```
-- Syntaxe Lingo
put(sound(2).channelCount)

// Syntaxe JavaScript
put(sound(2).channelCount);
```

Voir aussi

Piste audio

chapter

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.chapter

// Syntaxe JavaScript
réfObjDvd.chapter;
```

Description

Propriété DVD ; renvoie le numéro du chapitre courant. Lecture/écriture.

Exemple

Cette instruction renvoie le chapitre courant.

```
-- Syntaxe Lingo
trace (member(1).chapter) -- 1

// Syntaxe JavaScript
trace (member(1).chapter); // 1
```

Voir aussi

DVD

chapterCount

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.chapterCount

// Syntaxe JavaScript
réfObjDvd.chapterCount;
```

Description

Propriété DVD ; renvoie le nombre de chapitres disponibles dans un titre. En lecture seule.

Voir aussi

chapterCount(), DVD

characterSet

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.characterSet

// Syntaxe JavaScript
réfObjActeur.characterSet;
```

Description

Propriété d'acteur police ; renvoie une chaîne contenant les caractères inclus pour l'importation lors de la création de l'acteur. Si tous les caractères de la police d'origine sont inclus, le résultat est une chaîne vide.

Exemple

L'instruction suivante affiche les caractères inclus lorsque l'acteur 11 a été créé. Les caractères inclus durant l'importation étaient des caractères numériques et romains.

```
-- Syntaxe Lingo
put(member(11).characterSet)

// Syntaxe JavaScript
put(member(11).characterSet);
```

Voir aussi

recordFont, bitmapSizes, originalFont

charSpacing

Utilisation

```
-- Syntaxe Lingo
expressionSousChaîne.charSpacing

// Syntaxe JavaScript
expressionSousChaîne.charSpacing;
```

Description

Propriété d'acteur texte ; permet de spécifier tout espacement supplémentaire à appliquer à chaque lettre de la partie *expressionSousChaîne* de l'acteur texte.

Une valeur inférieure à 0 indique un espacement plus réduit entre les lettres. Une valeur supérieure à 0 indique un plus grand espacement entre les lettres.

La valeur par défaut est 0, ce qui active l'espacement par défaut entre les lettres.

Exemple

Le gestionnaire suivant augmente l'espacement des caractères courant du troisième au cinquième mot de l'acteur texte monTitre par une valeur de 2 :

```
-- Syntaxe Lingo
on monEspaceDeCaractères
  maValeurDespace = member("monTitre").word[3..5].charSpacing
  member("monTitre").word[3..5].charSpacing = (maValeurDespace + 2)
end
```

```
// Syntaxe JavaScript
function monEspaceDeCaractères() {
  var i = 3;
  while (i < 6) {
    var maValeurDespace = member("monTitre").getPropRef("word",
  i).charSpacing;
    member("monTitre").getPropRef("word", i).charSpacing =
  (maValeurDespace + 2);
  }
}
```

checkMark

Utilisation

```
the checkMark of menuItem quelElément of menu quelMenu
```

Description

Propriété d'élément de menu ; détermine si l'élément de menu personnalisé spécifié est affiché avec une coche (`TRUE`) ou non (`FALSE`, valeur par défaut).

La valeur *`quelElément`* peut être un nom ou un numéro d'élément de menu. La valeur *`quelMenu`* peut être un nom ou un numéro de menu.

Cette propriété peut être testée et définie.

Remarque : Les menus ne sont pas disponibles dans Shockwave Player.

Exemple

Le gestionnaire suivant désactive tout élément activé du menu personnalisé spécifié par l'argument `leMenu`. Par exemple, `désélectionner ("Format")` désactive tous les éléments du menu Format.

```
on désélectionner leMenu
  set n = the number of menuItems of menu leMenu
  repeat with i = 1 to n
    set the checkMark of menuItem i of menu leMenu to FALSE
  end repeat
end désélectionner
```

Voir aussi

`installMenu`, `enabled`, `name (propriété d'élément de menu)`, `number (éléments de menu)`, `script`, `menu`

child (3D)

Utilisation

```
member(quelActeur).model(quelNoeudParent).\
  child(quelNoeudEnfant)
member(quelActeur).model(quelNœudParent).child[index]
```

Description

Propriété 3D de modèle, de groupe, de lumière et de caméra ; renvoie le nœud enfant nommé *`quelNœudEnfant`* ou à l'index spécifié dans la liste d'enfants du nœud parent. Un nœud est un modèle, un groupe, une caméra ou une lumière.

La transformation d'un nœud est relative au parent. Si vous modifiez la position du parent, ses enfants se déplacent avec lui et leur position relative au parent est conservée. De même, la modification des propriétés de rotation et d'échelle du parent est également reflétée dans ses enfants.

Utilisez la méthode `addChild` du nœud parent ou définissez la propriété `parent` du nœud enfant pour l'ajouter à la liste d'enfants du parent. Alors qu'un enfant ne peut avoir qu'un parent, un parent peut avoir un nombre illimité d'enfants. Un enfant peut lui-même avoir des enfants.

Exemple

L'instruction suivante indique que le second enfant du modèle Voiture est le modèle Pneu.

```
put member("3D").model("Voiture").child[2]
-- model("Pneu")
```

Voir aussi

`addChild`, `parent`

child (XML)

Utilisation

nœudXML.child[*numéroDenfant*]

Description

Propriété XML ; fait référence au nœud enfant spécifié de la structure imbriquée d'un document XML analysé.

Exemple

Avec le code XML suivant :

```
<?xml version="1.0"?>
  <e1>
    <nomDeBalise attr1="val1" attr2="val2"/>
    <e2>élém 2</e2>
    <e3>élém 3</e3>
    Exemple de texte
  </e1>
```

L'instruction Lingo suivante renvoie le nom du premier nœud enfant du code XML précédent :

```
put gObjetDanalyse.child[1].name
-- "e1"
```

chunkSize

Utilisation

```
member(quelActeur).chunkSize
the chunkSize of member quelActeur
```

Description

Propriété d'acteur transition ; détermine la taille des blocs de la transition (entre 1 et 128 pixels) et a le même effet que le curseur de fluidité dans la boîte de dialogue Propriétés de l'image : Transition. Plus la taille des blocs est petite, plus la transition est fluide.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante paramètre la taille des blocs de l'acteur transition Brouillard sur 4 pixels :

Syntaxe à points :

```
member("Brouillard").chunkSize = 4
```

Syntaxe verbose :

```
set the chunkSize of member "Brouillard" to 4
```

clearAtRender

Utilisation

```
member(quelActeur).camera(quelleCaméra).colorBuffer.\
  clearAtRender
sprite(quelleImageObjet).camera{(index)}.colorBuffer.clearAtRender
```

Description

Propriété 3D ; indique si le tampon des couleurs est vidé après chaque image. La valeur `FALSE`, qui signifie que le tampon n'est pas vidé, produit un effet similaire aux traces d'encre. La valeur par défaut de cette propriété est `TRUE`.

Exemple

L'instruction suivante empêche Director d'effacer les images précédentes de la vue de la caméra. Les modèles en mouvement laisseront une trace sur la scène.

```
sprite(1).camera.colorBuffer.clearAtRender = 0
```

Voir aussi

`clearValue`

clearValue

Utilisation

```
member(quelActeur).camera(quelleCaméra).colorBuffer\
  .clearValue
sprite(quelleImageObjet).camera{(index)}.colorBuffer.clearValue
```

Description

Propriété 3D ; spécifie la couleur utilisée pour vider le tampon des couleurs si `colorBuffer.clearAtRender` a pour valeur `TRUE`. Le paramètre par défaut de cette propriété est `rgb(0, 0, 0)`.

Exemple

L'instruction suivante donne à la propriété `clearValue` de la caméra la valeur `rgb(255, 0, 0)`. L'espace de l'univers 3D qui n'est pas occupé par les modèles apparaîtra en rouge.

```
sprite(1).camera.colorBuffer.clearValue= rgb(255, 0, 0)
```

Voir aussi

`clearAtRender`

clickLoc

Utilisation

```
-- Syntaxe Lingo
_mouse.clickLoc

// Syntaxe JavaScript
_mouse.clickLoc;
```

Description

Propriété de souris ; identifie le dernier endroit de l'écran où un clic de la souris a eu lieu. En lecture seule.

Exemple

Le gestionnaire `on mouseDown` suivant affiche l'emplacement du dernier clic de la souris :

```
-- Syntaxe Lingo
on mouseDown
  put(_mouse.clickLoc)
end mouseDown

// Syntaxe JavaScript
function mouseDown() {
  put(_mouse.clickLoc);
}
```

Si l'utilisateur a cliqué sur un emplacement de la scène situé à 50 pixels de son bord gauche et à 100 pixels de son bord supérieur, la fenêtre Messages affiche :

```
point(50, 100)
```

Voir aussi

`clickOn`, `Souris`

clickMode

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.clickMode

// Syntaxe JavaScript
réfObjActeurOuImageObjet.clickMode;
```

Description

Propriété d'image-objet et d'acteur Flash ; contrôle le moment auquel l'image-objet de l'animation Flash détecte des événements de type clic de souris (`mouseUp` et `mouseDown`), ainsi que le moment auquel il détecte des survols (`mouseEnter`, `mouseWithin` et `mouseLeave`). La propriété `clickMode` peut avoir l'une des valeurs suivantes :

- `#boundingBox` – Détecte les événements de type clic de souris n'importe où dans le rectangle de délimitation de l'image-objet et détecte les survols aux limites de l'image-objet.

- `#opaque` (valeur par défaut) – Détecte les événements de type clic de souris uniquement lorsque le pointeur se trouve sur une portion opaque de l'image-objet et détecte les survols sur les limites des portions opaques de l'image-objet si l'effet d'encre de cette dernière est réglé sur Fond transparent. Si l'effet d'encre de l'image-objet a une autre valeur, ce paramètre produit le même effet que `#boundingBox`.
- `#object` – Détecte les événements de type clic de souris lorsque le pointeur de la souris se trouve sur une zone remplie (pas d'arrière-plan) de l'image-objet et détecte les survols sur les limites de toute zone remplie. Ce paramètre fonctionne quel que soit l'effet d'encre de l'image-objet.

Cette propriété peut être testée et définie.

Exemple

Le script suivant vérifie si l'image-objet spécifiée avec l'effet d'encre Fond transparent est définie pour un affichage au premier plan sur la scène. Si tel n'est pas le cas, la propriété `clickMode` prend pour valeur `#opaque`. Autrement (puisque les effets d'encre sont ignorés pour les images-objets d'animation Flash affichées au premier plan sur la scène), la propriété `clickMode` de l'image-objet prend pour valeur `#boundingBox`.

```
-- Syntaxe Lingo
property spriteNum

on beginSprite me
  if sprite(spriteNum).directToStage = FALSE then
    sprite(spriteNum).clickMode = #opaque
  else
    sprite(spriteNum).clickMode = #boundingBox
  end if
end

// Syntaxe JavaScript
function beginSprite(me){
  var dts = sprite(this.spriteNum).directToStage;
  if (dts = 0) {
    sprite(this.spriteNum).clickMode = symbol("opaque");
  } else {
    sprite(this.spriteNum).clickMode = symbol("boundingBox");
  }
}
```

clickOn

Utilisation

```
-- Syntaxe Lingo
_mouse.clickOn

// Syntaxe JavaScript
_mouse.clickOn;
```

Description

Propriété de souris ; renvoie la dernière image-objet active sur laquelle l'utilisateur a cliqué. En lecture seule.

Une image-objet active est une image-objet à laquelle un script d'image-objet ou d'acteur est associé.

Lorsque l'utilisateur clique sur la scène, `clickOn` renvoie 0. Pour détecter si l'utilisateur a cliqué sur une image-objet à laquelle aucun script n'est associé, vous devez lui assigner un script d'événement de souris afin de permettre sa détection par `clickOn`. Par exemple :

```
-- Syntaxe Lingo
on mouseUp me
  ...
end
```

Les boutons, cases à cocher et boutons radio sont détectés par `clickOn` même si aucun script ne leur est associé.

La propriété `clickOn` peut être testée dans une boucle. Cependant, ni `clickOn` ni `clickLoc` ne change de valeur lors de l'exécution du gestionnaire. La valeur que vous obtenez est celle précédant le démarrage du gestionnaire.

Exemple

L'instruction suivante vérifie si l'image-objet 7 est la dernière image-objet active sur laquelle l'utilisateur a cliqué :

```
-- Syntaxe Lingo
if (_mouse.clickOn = 7) then
  _player.alert("Désolé, veuillez recommencer.")
end if

// Syntaxe JavaScript
if (_mouse.clickOn = 7) {
  _player.alert("Désolé, veuillez recommencer.");
}
```

L'instruction suivante affecte une couleur aléatoire à la propriété `foreColor` de la dernière image-objet active sur laquelle l'utilisateur a cliqué :

```
-- Syntaxe Lingo
sprite(_mouse.clickOn).foreColor = (random(255) - 1)

// Syntaxe JavaScript
sprite(_mouse.clickOn).foreColor = (random(255) - 1);
```

Voir aussi

`clickLoc`, `Souris`

closed

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.closed

// Syntaxe JavaScript
réfObjActeur.closed;
```

Description

Propriété d'acteur forme vectorielle ; indique si les extrémités du contour sont ouvertes ou fermées.

Les formes vectorielles doivent être fermées pour leur remplissage.

La valeur peut être :

- `TRUE` – les extrémités sont fermées.
- `FALSE` – les extrémités sont ouvertes.

closedCaptions

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.closedCaptions

// Syntaxe JavaScript
réfObjDvd.closedCaptions;
```

Description

Propriété DVD. Détermine si l'insertion de sous-titres codés est activée (`TRUE`) ou si elle ne l'est pas – ou ne peut pas l'être (`FALSE`). A l'heure actuelle, pas de prise en charge sur Macintosh. Lecture/écriture.

Exemple

Ces instructions essaient de donner à `closedCaptions` la valeur TRUE et affichent une alerte si elles ne parviennent pas à activer les sous-titres codés :

```
-- Syntaxe Lingo
member(3).closedCaptions = TRUE
if (member(3).closedCaptions = FALSE) then
  _player.alert("Impossible d'activer les sous-titres codés.")
end if

// Syntaxe JavaScript
member(3).closedCaptions = true
if (member(3).closedCaptions == false) {
  _player.alert("Impossible d'activer les sous-titres codés.");
}
```

Voir aussi

`DVD`

collision (modificateur)

Utilisation

```
member(quelActeur).model(quelModèle).\
  collision.propriétéDeModificateurDeCollision
```

Description

Modificateur 3D ; gère la détection et la résolution des collisions. L'ajout du modificateur `collision` à un modèle à l'aide de la commande `addModifier` permet d'accéder aux propriétés suivantes du modificateur `collision` :

`enabled (collision)` indique si des collisions avec le modèle sont détectées.

`resolve` indique si les collisions avec le modèle sont résolues.

`immovable` indique si un modèle peut être déplacé d'une image à l'autre.

`mode (collision)` indique la géométrie utilisée pour la détection de collision.

Remarque : Pour plus d'informations, consultez les entrées des différentes propriétés.

Le modificateur de collision génère les événements suivants. Pour plus d'informations sur les événements de collision, consultez l'entrée de `registerForEvent()`.

Un événement `#collideAny` est généré lorsqu'une collision survient entre des modèles auxquels le modificateur `collision` a été associé.

Un événement `#collideWith` est généré lorsqu'une collision survient avec un modèle spécifique auquel le modificateur `collision` a été associé.

L'objet `collisionData` est envoyé comme argument avec les événements `#collideAny` et `#collideWith`. Pour plus d'informations sur ces propriétés, consultez `collisionData`.

Voir aussi

`addModifier`, `removeModifier`, `modifiers`

collisionData

Utilisation

```
on nomDeMonGestionnaire me, collisionData
```

Description

Objet de données 3D ; envoyé comme argument avec les événements `#collideWith` et `#collideAny` au gestionnaire spécifié dans les commandes `registerForEvent`, `registerScript` et `setCollisionCallback`. L'objet `collisionData` a les trois propriétés suivantes :

`modelA` est un des modèles impliqués dans la collision.

`modelB` est l'autre modèle impliqué dans la collision.

`pointOfContact` est la position de la collision dans l'univers.

`collisionNormal` est la direction de la collision.

Exemple

L'exemple suivant est constitué de trois parties. La première partie est la première ligne de code, qui enregistre le gestionnaire `#placerDétails` de l'événement `#collideAny`. La deuxième partie est le gestionnaire `#placerDétails`. Lorsque deux modèles de l'acteur maSéquence entrent en collision, le gestionnaire `#placerDétails` est appelé, et l'argument `collisionData` lui est envoyé. Ce gestionnaire affiche les quatre propriétés de l'objet `collisionData` dans la fenêtre Messages. La troisième partie de l'exemple affiche les résultats de la fenêtre Messages. Les deux premières lignes indiquent que les modèles impliqués dans la collision étaient balleVerte, modèle A, et balleJaune, modèle B. La troisième ligne indique le point de contact des deux modèles. La dernière ligne indique la direction de la collision.

```
member("maSéquence").registerForEvent(#collideAny, #placerDétails, 0)

on placerDétails me, collisionData
  put collisionData.modelA
  put collisionData.modelB
  put collisionData.pointOfContact
  put collisionData.collisionNormal
end

-- model("balleVerte")
-- model("balleJaune")
-- vector( 24.800, 0, 0 )
-- vector( 0, -1 000, 0 )
```

Voir aussi

Propriétés collisionData : modelA, modelB, pointOfContact, collisionNormal

Méthodes collisionData : resolveA, resolveB, collision (modificateur)

collisionNormal

Utilisation

collisionData.collisionNormal

Description

Propriété 3D collisionData ; vecteur indiquant la direction de la collision.

L'objet collisionData est envoyé comme argument avec les événements #collideWith et #collideAny au gestionnaire spécifié dans les commandes registerForEvent, registerScript et setCollisionCallback.

Les événements #collideWith et #collideAny sont envoyés lors d'une collision entre deux modèles associés à des modificateur de collision. La propriété resolve des modificateurs des modèles doit être TRUE.

Cette propriété peut être testée, mais pas définie.

Exemple

L'exemple suivant est constitué de deux parties. La première partie est la première ligne de code, qui enregistre le gestionnaire #explose de l'événement #collideAny. La seconde partie est le gestionnaire #explose. Lorsque deux modèles de l'acteur maSéquence entrent en collision, le gestionnaire #explose est appelé, et l'argument collisionData lui est envoyé. Les dix premières lignes du gestionnaire #explose créent la ressource de modèle sourceDétincelles et en définissent les propriétés. Cette ressource de modèle est une simple explosion de particules. La dixième ligne donne à la direction de l'explosion la valeur collisionNormal, ce qui est la direction de la collision. La onzième ligne du gestionnaire crée un modèle modèleDétincelles à l'aide de la ressource de modèle sourceDétincelles. La dernière ligne du gestionnaire définit la position de modèleDétincelles à l'emplacement de la collision. L'effet obtenu est une collision qui entraîne une explosion d'étincelles qui volent dans la direction de la collision, à partir du point de contact.

```
member("maSéquence").registerForEvent(#collideAny, #explose, 0)
on explose me, collisionData
  nmr = member("maSéquence").newModelResource("sourceDétincelles", #particle)
  nmr.emitter.mode = #burst
  nmr.emitter.loop = 0
  nmr.emitter.minSpeed = 30
  nmr.emitter.maxSpeed = 50
  nmr.emitter.angle = 45
  nmr.colorRange.start = rgb(0, 0, 255)
  nmr.colorRange.end = rgb(255, 0, 0)
  nmr.lifetime = 5000
  nmr.emitter.direction = vector(0,0,-1)
  nm = member("maSéquence").newModel("modèleDétincelles", nmr)
  nm.transform.position = collisionData.pointOfContact
  nm.pointAt(collisionData.pointOfContact + collisionData.collisionNormal)
end
```

Voir aussi

pointOfContact, modelA, modelB, resolveA, resolveB, collision (modificateur)

color()

Utilisation

```
color(#rvb, valeurRouge, valeurVerte, valeurBleue)
color(#indexDePalette, numéroDindexDePalette)
rgb(chaîneHexRvb)
rgb(valeurRouge, valeurVerte, valeurBleue)
paletteIndex(numéroDindexDePalette)
```

Description

Fonction et type de données ; détermine la couleur d'un objet sous la forme de valeurs RVB ou de valeurs d'un index de palette de 8 bits. Ces valeurs sont les mêmes que celles utilisées dans la propriété d'acteur `color` et dans la propriété d'image-objet `color`, dans la propriété d'acteur `bgColor` et dans la propriété d'image-objet `bgColor`, ainsi que dans la propriété de scène `bgColor`.

La fonction `color` permet de manipuler les valeurs de couleur sur 24 bits ou 8 bits, ainsi que de les appliquer aux acteurs, aux images-objets et à la scène.

Dans le cas des valeurs RVB, chaque composant de couleur possède une gamme de valeurs comprises entre 0 et 255, toutes les autres valeurs étant tronquées. Dans le cas des types `paletteIndex`, un entier de 0 à 255 est utilisé pour indiquer le numéro d'index dans la palette courante, toutes les autres valeurs étant tronquées.

Exemple

L'instruction suivante effectue une opération mathématique :

```
objetCouleurPal = paletteIndex(20)
put objetCouleurPal
-- paletteIndex(20)
put objetCouleurPal / 2
-- paletteIndex(10)
```

L'instruction suivante convertit un type de couleur en un autre type :

```
nouvelObjetCouleur = color(#rgb, 155, 0, 75)
put nouvelObjetCouleur
-- rgb(155, 0, 75)
nouvelObjetCouleur.colorType = #paletteIndex
put nouvelObjetCouleur
-- paletteIndex(106)
```

L'instruction suivante obtient la représentation hexadécimale d'une couleur, quel que soit son type :

```
unObjetCouleur = color(#paletteIndex, 32)
put unObjetCouleur.hexString()
-- "#FF0099"
```

L'instruction suivante détermine les composants RVB individuels et la valeur `paletteIndex` d'une couleur, quel que soit son type :

```
nouvelObjetCouleur = color(#rgb, 155, 0, 75)
put nouvelObjetCouleur.green
-- 0
put nouvelObjetCouleur.paletteIndex
-- 106
nouvelObjetCouleur.green = 100
put nouvelObjetCouleur.paletteIndex
-- 94
put nouvelObjetCouleur
-- rgb(155, 100, 75)
nouvelObjetCouleur.paletteIndex = 45
put nouvelObjetCouleur
-- paletteIndex(45)
```

L'instruction suivante modifie la couleur des caractères 4 à 7 de l'acteur texte mesCitations :

```
member("mesCitations").char[4..7].color = rgb(200, 150, 75)
```

L'instruction Lingo suivante affiche la couleur de l'image-objet 6 dans la fenêtre Messages, puis définit la couleur de l'image-objet 6 avec une nouvelle valeur RVB :

```
put sprite(6).color
-- rgb(255, 204, 102)
sprite(6).color = rgb(122, 98, 210)
```

Remarque : La définition de la valeur `paletteIndex` d'un type de couleur RVB remplace `colorType` par `paletteIndex`. La définition d'une composante RVB d'une couleur `paletteIndex` définit sa valeur `colorType` sur RVB.

Voir aussi

`bgColor (fenêtre)`

color (brouillard)

Utilisation

```
member(quelActeur).camera(quelleCaméra).fog.color
sprite(quelleImageObjet).camera{(index)}.fog.color
```

Description

Propriété 3D ; indique la couleur introduite dans la scène par la caméra lorsque sa propriété `fog.enabled` a pour valeur `TRUE`.

Le paramètre par défaut de cette propriété est `rgb(0, 0, 0)`.

Exemple

L'instruction suivante donne à la couleur du brouillard de la caméra vueDeLaBaie la valeur `rgb(255, 0, 0)`. Si la propriété `fog.enabled` de la caméra a pour valeur `TRUE`, les modèles dans le brouillard auront une teinte rouge.

```
member("monTerrain").camera("vueDeLaBaie").fog.color = rgb(255, 0, 0)
```

Voir aussi

`fog`

color (lumière)

Utilisation

`member(`*`quelActeur`*`).light(`*`quelleLumière`*`).color`

Description

Propriété 3D de lumière ; indique la valeur RVB de la lumière.

La valeur par défaut de cette propriété est `rgb(191,191,191)`.

Exemple

L'instruction suivante donne à la couleur de la lumière lumièreDeLaPièce la valeur `rgb(255, 0, 255)`.

```
member("Pièce").light("lumièreDeLaPièce").color = rgb(255,0,255)
```

Voir aussi

`fog`

colorBufferDepth

Utilisation

`getRendererServices().colorBufferDepth`

Description

Propriété 3D `rendererServices` ; indique la précision des couleurs du tampon de sortie matériel du système de l'utilisateur. La valeur est 16 ou 32, en fonction du matériel de l'utilisateur.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante indique que la valeur `colorBufferDepth` de la carte vidéo de l'utilisateur est 32.

```
put getRendererServices().colorBufferDepth
-- 32
```

Voir aussi

`getRendererServices()`, `getHardwareInfo()`, `depthBufferDepth`

colorDepth

Utilisation

```
-- Syntaxe Lingo
_system.colorDepth

// Syntaxe JavaScript
_system.colorDepth;
```

Description

Propriété système ; paramètre le codage des couleurs du moniteur. Lecture/écriture.

- Sous Windows, elle vous permet de vérifier et définir le codage des couleurs du moniteur. Le réglage de la propriété `colorDepth` est parfois impossible pour certaines combinaisons de cartes vidéo et de pilotes. Vérifiez toujours que le codage des couleurs a bien été changé après votre essai.
- Sur le Macintosh, elle vous permet de vérifier le codage des couleurs des différents moniteurs et de le modifier si nécessaire.

Les valeurs possibles sont :

`1`	Noir et blanc
`2`	4 couleurs
`4`	16 couleurs
`8`	256 couleurs
`16`	32 768 ou 65 536 couleurs
`32`	16 777 216 couleurs

Si vous essayez de changer le codage des couleurs avec une valeur impossible pour le moniteur, le codage reste inchangé.

Dans le cas des ordinateurs disposant de plusieurs moniteurs, la propriété `colorDepth` se réfère au moniteur sur lequel la scène est affichée. Si la scène s'étend sur plus d'un moniteur, la propriété `colorDepth` indique le codage le plus élevé de ces moniteurs ; `colorDepth` essaie d'affecter à tous ces moniteurs le codage spécifié.

Exemple

L'instruction suivante ne permet l'ouverture de l'animation Couleurs complètes que si le codage des couleurs du moniteur est de 256 couleurs :

```
-- Syntaxe Lingo
if (_system.colorDepth = 8) then
  window("Couleurs complètes").open()
end if

// Syntaxe JavaScript
if (_system.colorDepth == 8) {
  window("Couleurs complètes").open()
}
```

Le gestionnaire suivant tente de changer le codage des couleurs et, s'il n'y arrive pas, un message d'alerte apparaît :

```
-- Syntaxe Lingo
on essaiDeRéglageDesCouleurs(codageSouhaité)
  _system.colorDepth = codageSouhaité
  if (_system.colorDepth = codageSouhaité) then
    return true
  else
    _player.alert("Veuillez changer le codage des couleurs de votre ordinateur
  sur" && codageSouhaité &&"et redémarrer.")
    return false
  end if
end

// Syntaxe JavaScript
function essaiDeRéglageDesCouleurs(codageSouhaité) {
  _system.colorDepth = codageSouhaité;
  if (_system.colorDepth == codageSouhaité) {
    return TRUE;
  }
  else {
    _player.alert("Veuillez changer le codage des couleurs de votre ordinateur
  sur " + codageSouhaité +
  " et redémarrer.");
    return FALSE;
  }
}
```

Voir aussi

Système

colorList

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  colorList
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  colorList[index]
member(quelActeur).model(quelModèle).meshdeform.mesh\
  [indexDeMaille].colorList
member(quelActeur).model(quelModèle).meshdeform.mesh\
  [indexDeMaille].colorList[index]
```

Description

Propriété 3D ; permet d'obtenir ou de définir chaque couleur utilisée dans une maille. Cette commande ne peut être utilisée que pour les ressources de modèle de type #mesh. Chaque couleur peut être partagée par plusieurs sommets (faces) de la maille. Vous avez également la possibilité de spécifier les coordonnées de texture des faces de la maille et d'appliquer un matériau à l'aide de cette ressource de modèle.

Cette commande doit être définie à l'aide d'une liste ayant le même nombre de valeurs de couleurs Lingo que dans l'appel newMesh.

Exemple

L'instruction suivante indique que la troisième couleur de la liste `colorList` de la ressource de modèle Maille2 est `rgb(255, 0, 0)`.

```
put member("formes").modelResource("maille2").colorlist[3]
-- rgb(255,0,0)
```

Voir aussi

`face[ ]`, `colors`

colorRange

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  colorRange.start
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  colorRange.end
```

Description

Propriété 3D de ressource de modèle `#particle` ; indique les couleurs de début et de fin des particules d'un système de particules.

La propriété `start` définit la couleur des particules lorsqu'elles sont créées. La propriété `end` définit la couleur des particules à la fin de leur vie. La couleur de chaque particule change progressivement de la valeur `start` à la valeur `end` au long de leur vie.

Par défaut, les propriétés `start` et `end` ont pour valeur `rgb(255, 255, 255)`.

Exemple

L'instruction suivante définit les propriétés `colorRange` de la ressource de modèle systèmeThermique. La première ligne donne à `start` la valeur `rgb(255, 0, 0)` et la seconde donne à `end` la valeur `rgb(0, 0, 255)`. Cette instruction fait progressivement évoluer les particules de systèmeThermique du rouge au bleu, entre le moment où elles apparaissent et la fin de leur vie.

```
member(8,2).modelResource("systèmeThermique").colorRange.start = \
  rgb(255,0,0)
member(8,2).modelResource("systèmeThermique").colorRange.end = \
  rgb(0,0,255)
```

Voir aussi

`emitter`, `blendRange`, `sizeRange`

colors

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  face[indexDeFaces].colors
```

Description

Propriété 3D de face ; liste linéaire de trois nombres entiers indiquant les positions d'index de la liste des couleurs de la ressource de modèle à utiliser pour les trois sommets de la face. La liste des couleurs est une liste linéaire de valeurs RVB.

La propriété `colors` n'est utilisée qu'avec les ressources de modèle de type `#mesh`.

Vous devez utiliser la commande `build()` de la ressource de modèle après avoir défini cette propriété ; sinon, les modifications ne prendront pas effet.

Exemple

L'exemple suivant crée une ressource de modèle de type `#mesh`, en spécifie les propriétés, puis l'utilise pour créer un nouveau modèle.

La ligne 1 utilise la commande `newMesh()` pour créer une ressource de modèle `#mesh` nommée Triangle, qui consiste en une face, trois sommets et un maximum de trois couleurs. Le nombre de normales et de coordonnées de textures n'est pas défini.

La ligne 2 définit la propriété `vertexList` en une liste de trois vecteurs.

La ligne 3 affecte les vecteurs de la propriété `vertexList` aux sommets de la première face de Triangle.

La ligne 4 donne à la liste des couleurs trois valeurs RVB.

La ligne 5 affecte les couleurs à la première face de Triangle. La troisième couleur de la liste est appliquée au premier sommet de Triangle, la deuxième couleur au deuxième sommet, et la première couleur au troisième sommet. Les couleurs seront étalées sur la première face de Triangle en dégradés.

La ligne 6 crée les normales de Triangle avec la commande `generateNormals()`.

La ligne 7 utilise la commande `build()` pour construire la maille.

La ligne 8 crée un nouveau modèle nommé triModèle, qui utilise la nouvelle maille.

```
nm = member("Formes").newMesh("Triangle",1,3,0,3,0)
nm.vertexList = [vector(0,0,0), vector(20,0,0), vector(20, 20, 0)]
nm.face[1].vertices = [ 1,20,3 ]
nm.colorList = [rgb(255,255,0), rgb(0, 255, 0), rgb(0,0,255)]
nm.face[1].colors = [3,20,1]
nm.generateNormals(#smooth)
nm.build()
nm = member("Formes").newModel("triModèle", nm)
```

Voir aussi

`face`, `vertices`, `vertices`, `flat`

colorSteps

Utilisation

```
member(quelActeur).model(quelModèle).toon.colorSteps
member(quelActeur).model(quelModèle).shader.colorSteps
member(quelActeur).shader(quelMatériau).colorSteps
```

Description

Propriété 3D de modificateur `toon` et `painter` ; nombre maximum de couleurs disponibles pour le modificateur `toon` ou `painter`. La valeur de cette propriété peut être 2, 4, 8 ou 16. Si vous donnez une valeur différente à `colorSteps`, elle sera arrondie à l'une de ces valeurs autorisées.

La valeur par défaut est 2.

Exemple

L'instruction suivante limite le nombre de couleurs disponibles pour le modificateur `toon` du modèle Théière à 8. Le rendu de la théière sera composé au maximum de huit couleurs.

```
member("formes").model("Théière").toon.colorSteps = 8
```

Voir aussi

`highlightPercentage`, `shadowPercentage`

commandDown

Utilisation

```
-- Syntaxe Lingo
_key.commandDown

// Syntaxe JavaScript
_key.commandDown;
```

Description

Propriété de touche ; détermine si l'utilisateur appuie sur la touche Cmd du Macintosh ou Ctrl sous Windows. En lecture seule.

Cette propriété renvoie `TRUE` si l'utilisateur appuie sur la touche Cmd ou Ctrl. Sinon, elle renvoie `FALSE`.

Vous pouvez utiliser la fonction `commandDown` avec l'élément `key` pour déterminer si l'utilisateur appuie sur la touche Cmd ou Ctrl en combinaison avec une autre touche. Cela vous permet de créer des gestionnaires exécutés lorsque l'utilisateur appuie sur les touches spécifiées en combinaison avec la touche Cmd ou Ctrl.

Les raccourcis clavier utilisant la touche Cmd ou Ctrl des menus auteur de Director ont la priorité durant la lecture de l'animation, sauf si vous avez installé des menus personnalisés Lingo ou avec JavaScript, ou pendant la lecture d'une projection.

Exemple

Ces instructions mettent une projection en pause lorsque la tête de lecture entre dans une image et l'utilisateur appuie sur Ctrl+A (Windows) ou Commande+A (Macintosh).

```
-- Syntaxe Lingo
on enterFrame
  if (_key.commandDown and _key.key = "a") then
    _movie.go(_movie.frame)
  end if
end

// Syntaxe JavaScript
function enterFrame() {
  if (_key.commandDown && _key.key == "a") {
    _movie.go(_movie.frame);
  }
}
```

Voir aussi

`Touche`, `key`

comments

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.comments

// Syntaxe JavaScript
réfObjActeur.comments;
```

Description

Propriété d'acteur ; fournit un emplacement de stockage des commentaires que vous souhaitez conserver à propos d'un acteur ou de toute autre chaîne que vous souhaitez associer à cet acteur. Lecture/écriture.

Cette propriété peut également être définie dans le volet Acteur de l'inspecteur des propriétés.

Exemple

L'instruction suivante définit les commentaires de l'acteur Fond comme « Obtenez la permission d'utiliser ce graphique ».

```
-- Syntaxe Lingo
member("Fond").comments = "Obtenez la permission d'utiliser ce graphique"

// Syntaxe JavaScript
member("Fond").comments = "Obtenez la permission d'utiliser ce graphique";
```

Voir aussi

Acteur

compressed

Utilisation

```
member(quelActeur).texture(quelleTexture).compressed
```

Description

Propriété 3D de texture ; indique si l'acteur source de la texture est compressé (TRUE) ou non (FALSE). La valeur de la propriété compressed passe automatiquement de TRUE à FALSE lorsque la texture est nécessaire pour le rendu. La valeur peut être FALSE pour décompresser la texture en avance. Elle peut recevoir TRUE pour annuler l'affectation de la représentation décompressée de la mémoire. Les acteurs utilisés pour les textures ne sont pas compressés si la valeur est TRUE (indépendamment de la compression standard utilisée pour les acteurs bitmap lors de l'enregistrement d'une animation Director). La valeur par défaut de cette propriété est TRUE.

Exemple

L'instruction suivante donne à la propriété compressed de la texture placagePluton la valeur TRUE.

```
member("Séquence").texture("placagePluton").compressed = TRUE
```

Voir aussi

texture

constraint

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.constraint

// Syntaxe JavaScript
réfObjImageObjet.constraint;
```

Description

Propriété d'image-objet ; détermine si le point d'alignement d'une image-objet est limité au rectangle de délimitation d'une autre image-objet (1 ou `TRUE`) ou non (0 ou `FALSE`, la valeur par défaut). Lecture/écriture.

La propriété `constraint` est utile pour limiter le déplacement d'une image-objet mobile au rectangle de délimitation d'une autre image-objet. Elle permet de simuler une piste pour un curseur ou de limiter l'endroit de l'écran où l'utilisateur peut faire glisser un objet dans un jeu.

La propriété `constraint` s'applique aux images-objets mobiles et aux propriétés `locH` et `locV`. Le point de contrôle d'une image-objet mobile ne peut pas être déplacé en dehors du rectangle de délimitation de l'image-objet associée. (Dans le cas d'une image-objet bitmap, l'origine est le point d'alignement ; dans le cas d'une image-objet forme, l'origine est l'angle supérieur gauche délimitant la forme.) Dans le cas d'une image-objet forme, le point de contrôle est l'angle supérieur gauche délimitant la forme.) Lorsqu'une contrainte est définie pour une image-objet, les limites définies ont priorité sur toute coordonnée établie à l'aide des propriétés `locH` et `locV`.

Exemple

L'instruction suivante supprime une propriété d'image-objet `constraint` :

```
-- Syntaxe Lingo
sprite(5).constraint = 0

// Syntaxe JavaScript
sprite(5).constraint = 0;
```

L'instruction suivante contraint l'image-objet (i + 1) à la limite de l'image-objet 14 :

```
-- Syntaxe Lingo
sprite(i + 1).constraint = 14

// Syntaxe JavaScript
sprite(i + 1).constraint = 14;
```

L'instruction suivante vérifie si le déplacement de l'image-objet 3 est limité et, le cas échéant, active le gestionnaire `showConstraint` :

```
-- Syntaxe Lingo
if (sprite(3).constraint <> 0) then
  showConstraint
end if

// Syntaxe JavaScript
if (sprite(3).constraint != 0) {
  showConstraint();
}
```

Voir aussi

`locH`, `locV`, `Image-objet`

controlDown

Utilisation

```
-- Syntaxe Lingo
_key.controlDown

// Syntaxe JavaScript
_key.controlDown;
```

Description

Propriété de touche ; détermine si l'utilisateur appuie sur la touche Ctrl. En lecture seule.

Cette propriété renvoie `TRUE` si l'utilisateur appuie sur la touche Ctrl. Sinon, elle renvoie `FALSE`.

Vous pouvez utiliser la fonction `controlDown` avec la propriété `key` pour déterminer si l'utilisateur appuie sur la touche Ctrl en combinaison avec une autre touche. Cela vous permet de créer des gestionnaires exécutés lorsque l'utilisateur appuie sur les touches spécifiées en combinaison avec la touche Ctrl.

Les raccourcis clavier utilisant la touche Ctrl des menus auteur de Director ont la priorité durant la lecture de l'animation, sauf si vous avez installé des menus personnalisés Lingo ou de syntaxe JavaScript, ou pendant la lecture d'une projection.

Exemple

Le gestionnaire `on keyDown` suivant vérifie si la touche enfoncée est la touche Ctrl et, le cas échéant, active le gestionnaire `on doControlKey`. L'argument `(_key.key)` identifie la touche enfoncée en plus de la touche Ctrl.

```
-- Syntaxe Lingo
on keyDown
  if (_key.controlDown) then
    doControlKey(_key.key)
  end if
end

on doControlKey(theKey)
  trace("La " & theKey & " touche est enfoncée")
end

// Syntaxe JavaScript
function keyDown() {
  if (_key.controlDown) {
    doControlKey(_key.key);
  }
}

function doControlKey(theKey) {
  trace("La " & theKey & " touche est enfoncée");
}
```

Voir aussi

Touche, key

controller

Utilisation

```
member(quelActeur).controller
the controller of member quelActeur
```

Description

Propriété d'acteur vidéo numérique ; détermine si un acteur animation vidéo numérique affiche ou masque son contrôleur. Le paramétrage de cette propriété sur 1 affiche le contrôleur, alors que 0 le masque.

La propriété d'acteur `controller` s'applique uniquement à la vidéo numérique QuickTime.

- La définition de la propriété d'acteur `the controller` pour une vidéo numérique Vidéo pour Windows ne donne aucun résultat et n'entraîne aucun message d'erreur.
- Le test de la propriété d'acteur `controller` pour une vidéo numérique Vidéo pour Windows renvoie toujours la valeur `FALSE`.

La vidéo numérique doit être en mode de lecture au premier plan pour afficher le contrôleur.

Exemple

L'instruction suivante entraîne l'affichage du contrôleur de l'acteur QuickTime Démo :

Syntaxe à points :

```
member("Démo").controller = 1
```

Syntaxe verbose :

```
set the controller of member "Démo" to 1
```

Voir aussi

`directToStage`

copyrightInfo (animation)

Utilisation

```
-- Syntaxe Lingo
_movie.copyrightInfo

// Syntaxe JavaScript
_movie.copyrightInfo;
```

Description

Propriété d'animation ; permet de saisir une chaîne dans la boîte de dialogue Propriétés de l'animation pendant la programmation. Cette propriété permettra des améliorations dans de futures versions de Shockwave Player. En lecture seule.

Voir aussi

`aboutInfo`, `Animation`

copyrightInfo (SWA)

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.copyrightInfo

// Syntaxe JavaScript
réfObjActeur.copyrightInfo;
```

Description

Propriété d'acteur Shockwave Audio (SWA) ; affiche le texte de copyright d'un fichier SWA. Cette propriété n'est disponible qu'après le début de la lecture du son SWA ou après le préchargement du fichier avec la commande `preLoadBuffer`.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante entraîne l'affichage par Director de l'information de copyright du fichier Shockwave Audio SWA dans un acteur champ nommé Infos :

```
-- Syntaxe Lingo
quelEtat = member("fichierSWA").state
if quelEtat > 1 AND quelEtat < 9 then
  put(member("Infos") = member("fichierSWA").copyrightInfo)
end if

// Syntaxe JavaScript
var quelEtat = member("fichierSWA").state;
if (quelEtat > 1 && quelEtat < 9) {
  put(member("Infos") = member("fichierSWA").copyrightInfo);
}
```

count

Utilisation

```
liste.count
count (liste)
count(quelObjet)
quelObjet.count
expressionTexte.count
```

Description

Propriété (Lingo uniquement) ; renvoie le nombre d'entrées d'une liste linéaire ou de propriétés, le nombre de propriétés d'un script parent sans compter les propriétés d'un script ancêtre ou les sous-chaînes d'une expression texte telles que caractères, lignes ou mots.

La commande `count` fonctionne avec les listes linéaires et de propriétés, les objets créés avec des scripts parents et la propriété `the globals`.

Vous pourrez voir un exemple de `count()` dans une animation en consultant l'animation Text du dossier Learning/Lingo, lui-même dans le dossier de Director.

Exemple

L'instruction suivante affiche la valeur 3, qui correspond au nombre d'entrées :

```
put [10,20,30].count
-- 3
```

Voir aussi

globals

count (3D)

Utilisation

```
member(quelActeur).light.count
member(quelActeur).camera.count
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  bone.count
member(quelActeur).model.count
member(quelActeur).group.count
member(quelActeur).shader.count
member(quelActeur).texture.count
member(quelActeur).modelResource.count
member(quelActeur).motion.count
member(quelActeur).light.child.count
member(quelActeur).camera.child.count
member(quelActeur).model.child.count
member(quelActeur).group.child.count
sprite(quelleImageObjet).camera{(index)}.backdrop.count
member(quelActeur).camera(quelleCaméra).backdrop.count
sprite(quelleImageObjet).camera{(index)}.overlay.count
member(quelActeur).camera(quelleCaméra).overlay.count
member(quelActeur).model(quelModèle).modifier.count
member(quelActeur).model(quelModèle).keyframePlayer.\
  playlist.count
member(quelActeur).model(quelModèle).bonesPlayer.\
  playlist.count
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  face.count
member(quelActeur).model(quelModèle).meshDeform.\
  mesh[index].textureLayer.count
member(quelActeur).model(quelModèle).meshDeform.mesh.count
member(quelActeur).model(quelModèle).meshDeform.\
  mesh[index].face.count
```

Description

Propriété 3D ; renvoie le nombre d'éléments de la liste spécifiée associée à l'objet 3D indiqué. Utilisable avec n'importe quel type d'objet.

La propriété `face.count` permet d'obtenir le nombre de triangles dans la maille pour une ressource de modèle de type `#mesh`.

Cette propriété peut être testée, mais pas définie.

Exemple

Les exemples suivants déterminent le nombre de types d'objets au sein d'un acteur 3D appelé Univers 3D.

```
nombreDeCaméras = member("Univers 3D").camera.count
put member("Univers 3D").light.count
-- 3
nombreDeModèles = member("Univers 3D").model.count
nombreDeTextures = member("Univers 3D").texture.count
put member("Univers 3D").modelResource("maille2").face.count
-- 4
```

L'instruction suivante indique que la première maille du modèle Oreille est composée de 58 faces.

```
put member("Séquence").model("Oreille").meshdeform.mesh[1].face.count
-- 58
```

L'instruction suivante indique que le modèle Oreille est composé de trois mailles.

```
put member("Séquence").model("Oreille").meshdeform.mesh.count
-- 3
```

L'instruction suivante indique que la première maille du modèle Oreille compte deux couches de texture.

```
put member("Séquence").model("Oreille").meshdeform.mesh[1].\
  textureLayer.count
-- 2
```

Voir aussi

`cameraCount()`

cpuHogTicks

Utilisation

```
the cpuHogTicks
```

Description

Propriété système ; détermine la fréquence avec laquelle Director libère le contrôle du processeur afin de permettre à l'ordinateur de traiter des événements d'arrière-plan, tels que ceux qui se produisent dans d'autres applications, des événements réseau, des mises à jour de l'horloge et d'autres événements de clavier.

La valeur par défaut est de 20 battements. Pour accorder un délai supplémentaire à Director avant de libérer le processeur pour les autres événements ou en vue du contrôle des opérations réseau, donnez une valeur supérieure à `cpuHogTicks`.

Vous pouvez obtenir une répétition automatique de touches plus rapide en donnant une valeur plus faible à `cpuHogTicks`, mais cela ralentit l'animation. Dans une animation, lorsque l'utilisateur maintient une touche enfoncée pour créer une suite rapide de pressions de touches, Director vérifie généralement moins fréquemment les répétitions automatiques de pression de touches que la fréquence définie dans le tableau de bord de l'ordinateur.

La propriété `cpuHogTicks` ne fonctionne que sur Macintosh.

Exemple

L'instruction suivante demande à Director de libérer le processeur tous les 6 battements, ce qui représente un dixième de seconde :

```
the cpuHogTicks = 6
```

Voir aussi

milliseconds

creaseAngle

Utilisation

```
member(quelActeur).model(quelModèle).inker.creaseAngle
member(quelActeur).model(quelModèle).toon.creaseAngle
```

Description

Propriété 3D de modificateur `inker` et `toon` ; indique la sensibilité de la fonction de traçage de ligne du modificateur à la présence de plis dans la géométrie du modèle. Des paramètres plus élevés entraînent plus de lignes (détails) dessinées aux plis.

La propriété `creases` du modificateur doit avoir la valeur `TRUE` pour que la propriété `creaseAngle` soit effective.

La plage de `CreaseAngle` s'étend de -1.0 à +1.0. Le paramètre par défaut est 0,01.

Exemple

L'instruction suivante donne à la propriété `creaseAngle` du modificateur `inker` appliqué au modèle Théière la valeur 0.10. Une ligne sera dessinée pour tous les plis du modèle dépassant ce seuil. Ce paramètre ne sera effectif que si la propriété `creases` du modificateur `inker` a pour valeur `TRUE`.

```
member("formes").model("Théière").inker.creaseAngle = 0.10
```

Voir aussi

creases, lineColor, lineOffset, useLineOffset

creases

Utilisation

```
member(quelActeur).model(quelModèle).inker.creases
member(quelActeur).model(quelModèle).toon.creases
```

Description

Propriété 3D de modificateur `toon` et `inker` ; détermine si des lignes sont dessinées aux plis, à la surface du modèle.

Le paramètre par défaut de cette propriété est `TRUE`.

Exemple

L'instruction suivante donne à la propriété `creases` du modificateur `inker` du modèle Théière la valeur `TRUE`. Une ligne sera dessinée pour tous les plis du modèle dépassant le seuil défini par la propriété `creaseAngle` du modificateur `inker`.

```
member("formes").model("Théière").inker.creases = TRUE
```

Voir aussi

`creaseAngle`, `lineColor`, `lineOffset`, `useLineOffset`

creationDate

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.creationDate

// Syntaxe JavaScript
réfObjActeur.creationDate;
```

Description

Propriété d'acteur ; enregistre la date de la création initiale de l'acteur, à l'aide de la date système de l'ordinateur. En lecture seule.

Vous pouvez utiliser cette propriété pour programmer un projet ; elle n'est pas utilisée par Director.

Exemple

Bien que la propriété `creationDate` soit généralement vérifiée à l'aide de l'inspecteur des propriétés ou de la fenêtre Distribution en mode d'affichage sous forme de liste, vous pouvez également la vérifier dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(member(1).creationDate)

// Syntaxe JavaScript
put(member(1).creationDate);
```

Voir aussi

`Acteur`

crop

Utilisation

```
member(quelActeur).crop
the crop of member quelActeur
```

Description

Propriété d'acteur ; met à l'échelle un acteur vidéo numérique pour le faire tenir exactement dans le rectangle de l'image-objet dans lequel il apparaît (`FALSE`) ou le recadre, sans en modifier la taille, pour le faire tenir dans le rectangle de l'image-objet (`TRUE`).

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante indique à Lingo de recadrer toute image-objet faisant référence à l'acteur vidéo numérique Interview.

Syntaxe à points :

```
member("Interview").crop = TRUE
```

Syntaxe verbose :

```
set the crop of member "Interview" to TRUE
```

Voir aussi

center

cuePointNames

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.cuePointNames

// Syntaxe JavaScript
réfObjActeur.cuePointNames;
```

Description

Propriété d'acteur ; crée une liste des noms de points de repère ou, si un point de repère n'a pas de nom, une chaîne vide ("") est insérée comme repère dans la liste. Les noms de points de repère sont pratiques pour la synchronisation du son, des acteurs QuickTime et de l'animation.

Cette propriété est prise en charge par les acteurs SoundEdit, les acteurs vidéo numérique QuickTime et les acteurs Xtras contenant des points de repère. La liste des points de repère peut ne pas être disponible pour les Xtras créant des points de repère au moment de l'exécution de l'animation.

Exemple

L'instruction suivante obtient le nom du troisième point de repère d'un acteur.

```
-- Syntaxe Lingo
put member("symphonie").cuePointNames[3]

// Syntaxe JavaScript
put(member("symphonie").cuePointNames[3]);
```

Voir aussi

cuePointTimes, mostRecentCuePoint

cuePointTimes

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.cuePointTimes

// Syntaxe JavaScript
réfObjActeur.cuePointTimes;
```

Description

Propriété d'acteur ; fournit une liste des positions des points de repère d'un acteur donné, en millisecondes. Les positions des points de repère sont pratiques pour la synchronisation du son, des acteurs QuickTime et de l'animation.

Cette propriété est prise en charge par les acteurs SoundEdit, les acteurs vidéo numérique QuickTime et les acteurs Xtras prenant en charge des points de repère. La liste des points de repère peut ne pas être disponible pour les Xtras créant des points de repère au moment de l'exécution de l'animation.

Exemple

L'instruction suivante obtient la position du troisième point de repère d'un acteur son.

```
-- Syntaxe Lingo
put member("symphonie").cuePointTimes[3]

// Syntaxe JavaScript
put(member("symphonie").cuePointTimes[3]);
```

Voir aussi

cuePointNames, mostRecentCuePoint

currentLoopState

Utilisation

```
member(quelActeur).model(quelModèle).keyframePlayer.\
  currentLoopState
member(quelActeur).model(quelModèle).bonesPlayer.\
  currentLoopState
```

Description

Propriété 3D de modificateur `keyframePlayer` et `bonesPlayer` ; indique si le mouvement exécuté par le modèle est répété continuellement (`TRUE`) ou s'il est exécuté jusqu'à la fin puis remplacé par le mouvement suivant dans la liste de lecture du modificateur (`FALSE`).

Le paramètre par défaut de cette propriété est la valeur du paramètre de boucle de la commande `play()` qui a démarré la lecture du mouvement ou la valeur de la commande `queue()` qui a ajouté le mouvement à la liste de lecture du modificateur. Le fait de modifier la propriété `currentLoopState` change également la valeur de la propriété `#looped` de l'entrée du mouvement dans la liste de lecture du modificateur.

Exemple

L'instruction suivante entraîne la lecture continue du mouvement exécuté par le modèle Monstre.

```
member("nouveauMartien").model("Monstre").keyframePlayer.\
  currentLoopState = TRUE
```

Voir aussi

loop (3D), play() (3D), queue() (3D), playlist

currentSpriteNum

Utilisation

```
-- Syntaxe Lingo
_player.currentSpriteNum

// Syntaxe JavaScript
_player.currentSpriteNum;
```

Description

Propriété de lecteur ; indique le numéro de piste de l'image-objet dont le script est en cours d'exécution. En lecture seule.

Cette propriété peut être utilisée dans les scripts d'acteurs et les comportements. Lorsqu'elle est utilisée dans les scripts d'images ou d'animations, la propriété `currentSpriteNum` a la valeur 0.

La propriété `currentSpriteNum` est similaire à la propriété `spriteNum` de l'objet Image-objet.

Remarque : Cette propriété était plus utile pour le passage des anciennes animations à Director 6, lors de l'apparition des comportements. Elle permettait une fonction de type comportement sans vous contraindre à réécrire complètement le script. Elle n'est pas nécessaire pour la programmation avec des comportements et est donc moins utile que par le passé.

Exemple

Le gestionnaire suivant dans un script d'acteur ou d'animation remplace l'acteur affecté à l'image-objet impliquée dans l'événement `mouseDown` :

```
-- Syntaxe Lingo
on mouseDown
  sprite(_player.currentSpriteNum).member = member("imageEnfoncée")
end

// Syntaxe JavaScript
function mouseDown() {
  sprite(_player.currentSpriteNum).member = member("imageEnfoncée");
}
```

Voir aussi

Lecteur, spriteNum

currentTime (3D)

Utilisation

```
member(quelActeur).model(quelModèle).keyframePlayer.\
  currentTime
member(quelActeur).model(quelModèle).bonesPlayer.\
  currentTime
```

Description

Propriété 3D de modificateur `keyframePlayer` et `bonesPlayer` ; indique la position locale du mouvement exécuté par le modèle. La propriété `currentTime` est mesurée en millisecondes, mais ne correspond à la position réelle que lorsque le mouvement est lu à sa cadence originale.

La lecture d'un mouvement par un modèle est le résultat de la commande `play()` ou `queue()`. Le paramètre `scale` de la commande `play()` ou `queue()` est multiplié par la propriété `playRate` du modificateur, et la valeur qui en résulte est multipliée par la cadence originale du mouvement pour déterminer la cadence à laquelle le modèle exécutera le mouvement. Ainsi, si le paramètre `scale` a pour valeur 2 et la propriété `playRate` du modificateur pour valeur 3, le modèle exécutera le mouvement six fois plus vite qu'à la cadence originale.

La propriété `currentTime` réinitialise la valeur à celle du paramètre `cropStart` de la commande `play()` ou `queue()` au début de chaque itération d'un mouvement en boucle.

Exemple

L'instruction suivante indique la position locale du mouvement exécuté par le modèle Martien3.

```
put member("nouveauMartien").model("Martien3").keyframePlayer.currentTime
-- 1393.8599
```

Voir aussi

`play() (3D)`, `queue() (3D)`, `playlist`

currentTime (DVD)

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.currentTime

// Syntaxe JavaScript
réfObjDvd.currentTime;
```

Description

Propriété DVD ; revoie le temps écoulé, en millisecondes. Lecture/écriture.

Exemple

Cette instruction renvoie le temps écoulé :

```
-- Syntaxe Lingo
trace (member(1).currentTime)  -- 11500

// Syntaxe JavaScript
trace (member(1).currentTime);  // 11500
```

Cette instruction définit `currentTime` sur un point spécifique du titre courant :

```
-- Syntaxe Lingo
member(1).currentTime = 22000

// Syntaxe JavaScript
member(1).currentTime = 22000
```

Voir aussi

DVD

currentTime (QuickTime, AVI)

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.currentTime

// Syntaxe JavaScript
réfObjImageObjet.currentTime;
```

Description

Propriété d'image-objet vidéo numérique ; détermine la position temporelle courante de l'animation vidéo numérique exécutée dans la piste spécifiée par *réfObjImageObjet*. La valeur de `movieTime` est mesurée en battements.

Cette propriété peut être testée et définie.

Vous pourrez voir un exemple de `currentTime` dans une animation en consultant l'animation QT et Flash du dossier Learning/Lingo, lui-même dans le dossier de Director.

Exemple

L'instruction suivante affiche dans la fenêtre Messages la durée actuelle de l'animation QuickTime de la piste 9 :

```
-- Syntaxe Lingo
put(sprite(9).currentTime)

// Syntaxe JavaScript
put(sprite(9).currentTime);
```

L'instruction suivante affecte à la position temporelle actuelle de l'animation QuickTime de la piste 9 la valeur de la variable `Poster` :

```
-- Syntaxe Lingo
sprite(9).currentTime = Poster

// Syntaxe JavaScript
sprite(9).currentTime = Poster;
```

Voir aussi

duration (acteur)

currentTime (RealMedia)

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.currentTime

// Syntaxe JavaScript
réfObjActeurOuImageObjet.currentTime;
```

Description

Propriété d'image-objet ou d'acteur RealMedia ; permet d'obtenir ou de définir la position du flux RealMedia, en millisecondes. Si l'acteur RealMedia n'est pas en cours de lecture, la valeur de cette propriété est 0, qui constitue la valeur par défaut. Il s'agit d'une propriété de lecture qui n'est pas enregistrée.

Si le flux est en cours de lecture lors de la définition ou de la modification de la propriété `currentTime`, une recherche est lancée, ainsi que la remise en tampon du flux, avant que la lecture ne reprenne à la nouvelle position définie. Si le flux est interrompu (valeur `#paused mediaStatus`) lors de la définition ou de la modification de la propriété `currentTime`, le flux redessine l'image à la nouvelle position, et la lecture reprend si la propriété `pausedAtStart` a pour valeur `FALSE`. Lorsque le flux est interrompu ou arrêté dans la fenêtre RealMedia, la propriété `mediaStatus` a pour valeur `#paused`. Lorsque le flux est arrêté par la commande Lingo `stop`, `mediaStatus` a pour valeur `#closed`. Cette propriété n'a aucun effet lorsque la valeur de `mediaStatus` est `#closed`. Lorsque vous définissez des nombres entiers, ils sont ajoutés à la plage à partir de 0 pour la durée du flux.

La définition de `currentTime` est l'équivalent de l'invocation de la commande `seek` : `x.seek(n)` est l'équivalent de `x.currentTime = n`. La modification de `currentTime` ou l'appel de `seek` nécessite la remise en tampon du flux.

Exemple

Les exemples suivants indiquent que la valeur de position temporelle de l'image-objet 2 et de l'acteur Real est 15 534 millisecondes (15,534 secondes) depuis le début du flux.

```
-- Syntaxe Lingo
put(sprite(2).currentTime) -- 15534
put(member("Real").currentTime) -- 15534

// Syntaxe JavaScript
put(sprite(2).currentTime) // 15534
put(member("Real").currentTime) // 15534
```

Les exemples suivants provoquent un saut de lecture de 20 000 millisecondes (20 secondes) dans le flux de l'image-objet 2 et de l'acteur Real.

```
-- Syntaxe Lingo
sprite(2).currentTime = 20000
member("Real").currentTime = 20000

// Syntaxe JavaScript
sprite(2).currentTime = 20000
member("Real").currentTime = 20000
```

Voir aussi

`duration (RealMedia, SWA)`, `seek()`, `mediaStatus (RealMedia, Windows Media)`

currentTime (image-objet)

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.currentTime

// Syntaxe JavaScript
réfObjImageObjet.currentTime;
```

Description

Propriété d'image-objet et de piste audio ; renvoie la position temporelle de lecture, en millisecondes, d'une image-objet audio ou vidéo numérique QuickTime ou de n'importe quelle extension Xtra supportant les points de repère. Pour une piste audio, renvoie la position temporelle de lecture de l'acteur son actuellement en cours de lecture dans la piste audio concernée.

Cette propriété ne peut être définie que pour les acteurs son traditionnels (*.wav, *.aif, *.snd). Lorsque cette propriété est définie, la gamme des valeurs admises s'étend de zéro à la valeur `duration` de l'acteur.

Les sons Shockwave Audio (SWA) peuvent apparaître sous forme d'images-objets dans les pistes d'images-objets, mais sont lus dans les pistes audio. Il est conseillé de faire référence aux images-objets audio SWA par le numéro de leur piste d'image-objet plutôt que par celui de leur piste audio.

Exemple

L'instruction suivante indique la position temporelle courante, en secondes, de l'image-objet dans la piste 10.

```
-- Syntaxe Lingo
member("position").text = string(sprite (10).currentTime/ 1000)

// Syntaxe JavaScript
member("position").text = (sprite(10).currentTime / 1000).toString();
```

L'instruction suivante entraîne le passage du son de la piste audio 2 au point 2,7 secondes à partir du début de l'acteur son :

```
-- Syntaxe Lingo
sound(2).currentTime = 2700

// Syntaxe JavaScript
sound(2).currentTime = 2700;
```

Voir aussi

`duration (acteur)`

cursor

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.cursor

// Syntaxe JavaScript
réfObjImageObjet.cursor;
```

Description

Propriété d'image-objet ; détermine le curseur utilisé lorsque le pointeur se trouve au-dessus d'une image-objet. Lecture/écriture.

Cette propriété reste en vigueur jusqu'à ce que vous la désactiviez en lui attribuant la valeur 0. Utilisez la propriété `cursor` pour changer le pointeur de la souris lorsqu'il est sur des zones spécifiques de l'écran, ainsi que pour indiquer les zones où certaines actions sont possibles lorsque l'utilisateur clique.

Lorsque vous définissez la propriété `cursor` pour une image spécifique, Director examine le rectangle de l'image avant de décider de modifier ou non le curseur. Ce rectangle ne change pas lorsque l'animation passe à l'image suivante, sauf si vous attribuez une valeur nulle à la propriété `cursor` de cette piste.

- Utilisez la syntaxe suivante pour spécifier le numéro d'acteur à utiliser comme curseur et son masque facultatif.

```
-- Syntaxe Lingo
réfObjImageObjet.cursor = [réfObjActeur, réfObjActeurMasque]

// Syntaxe JavaScript
réfObjImageObjet.cursor = [réfObjActeur, réfObjActeurMasque];
```

- Utilisez la syntaxe suivante pour spécifier les curseurs par défaut fournis par le système.

```
-- Syntaxe Lingo
réfObjImageObjet.cursor = réfObjActeur

// Syntaxe JavaScript
réfObjImageObjet.cursor = réfObjActeur;
```

La propriété `cursor` peut avoir l'une des valeurs (nombre entier) suivantes :

Valeur	Description
-1, 0	Flèche
1	Curseur en I
2	Croix
3	Croix épaisse
4	Montre (Macintosh) ou sablier (Windows)
5	Nord Sud Est Ouest (NSEO)
6	Nord Sud (NS)
200	Vide (masque le curseur)
254	Aide

Valeur	Description
256	Crayon
257	Gomme
258	Sélection
259	Pot de peinture
260	Main
261	Outil Rectangle
262	Outil Rectangle arrondi
263	Outil Cercle
264	Outil Ligne
265	Outil RTF
266	Outil Champ de texte
267	Outil Bouton
268	Outil Case à cocher
269	Outil Bouton radio
270	Outil Position
271	Outil Point d'alignement
272	Lasso
280	Doigt
281	Pipette
282	Attente bouton de souris enfoncé 1
283	Attente bouton de souris enfoncé 2
284	Taille verticale
285	Taille horizontale
286	Taille diagonale
290	Main fermée
291	Main sans déposer
292	Copie (main fermée)
293	Flèche inversée
294	Rotation
295	Inclinaison
296	Double flèche horizontale
297	Double flèche verticale
298	Double flèche sud-ouest nord-est

Valeur	Description
299	Double flèche nord-ouest sud-est
300	Pinceau pour enduire/estomper
301	Aérographe
302	Zoom avant
303	Zoom arrière
304	Annulation du zoom
305	Démarrer la forme
306	Ajouter un point
307	Fermer la forme
308	Zoom de la caméra
309	Déplacement de la caméra
310	Rotation de la caméra
457	Personnalisé

Pour utiliser des curseurs personnalisés, affectez la propriété `cursor` à une liste contenant l'acteur à utiliser comme curseur ou au numéro spécifiant un curseur du système. Sous Windows, un curseur doit être un acteur et non une ressource ; si aucun curseur n'est disponible puisqu'il s'agit d'une ressource, Director affiche à la place le curseur flèche standard. Pour garantir des résultats optimaux, veillez à ne pas utiliser de curseurs lorsque vous créez des animations multiplates-formes.

Les acteurs curseur personnalisé ne doivent pas dépasser 16 x 16 pixels et doivent avoir une résolution de 1 bit.

Si l'image-objet est un bitmap avec une encre Dessin seul, le curseur ne change que lorsqu'il se trouve sur la portion Dessin seul de l'image-objet.

Lorsque le pointeur se trouve à l'emplacement d'une image-objet qui a été supprimée, le survol se produit quand même. Vous pouvez éviter ce problème en ne faisant pas de survol à ces endroits ou en déplaçant l'image-objet au-dessus de la barre des menus avant de la supprimer.

Sur le Macintosh, vous pouvez utiliser une ressource de curseur numérotée dans l'animation en cours à la place du curseur en affectant à `cursor` le numéro de la ressource de curseur.

Exemple

L'instruction suivante change en montre (Macintosh) ou sablier (Windows) le curseur qui apparaît sur l'image-objet 20 :

```
-- Syntaxe Lingo
sprite(20).cursor = 4

// Syntaxe JavaScript
sprite(20).cursor = 4;
```

Voir aussi

Image-objet

cursorSize

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.cursorSize

// Syntaxe JavaScript
réfObjActeur.cursorSize;
```

Description

Propriété d'acteur curseur ; spécifie la taille de l'acteur curseur couleur animé `quelActeurCurseur`.

Spécifiez la taille :	Pour des curseurs pouvant atteindre :
`16`	16 x 16 pixels
`32`	32 x 32 pixels

Les acteurs bitmap inférieurs à la taille spécifiée sont affichés en taille normale, tandis que les acteurs plus grands sont ramenés à la taille spécifiée.

La valeur par défaut est 32 pour Windows et 16 pour Macintosh. Si vous choisissez une valeur incorrecte, un message d'erreur s'affiche pendant la lecture de l'animation (mais pas à la compilation).

Cette propriété peut être testée et définie.

Exemple

La commande suivante redimensionne le curseur couleur animé stocké dans l'acteur curseur couleur 20 à 32 x 32 pixels.

```
-- Syntaxe Lingo
member(20).cursorSize = 32

// Syntaxe JavaScript
member(20).cursorSize = 32;
```

curve

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.curve[indiceDeListeDeCourbe]

// Syntaxe JavaScript
réfObjActeur.curve[indiceDeListeDeCourbe];
```

Description

Cette propriété contient la liste `vertexList` d'une courbe individuelle (forme) provenant d'un acteur forme vectorielle. Vous pouvez utiliser la propriété `curve` conjointement à la propriété `vertex` pour obtenir les sommets individuels d'une courbe spécifique dans une forme vectorielle.

Une `vertexList` est une liste de sommets, chaque sommet étant une liste de propriétés comprenant un maximum de trois propriétés : une propriété `#vertex`, avec l'emplacement du sommet, une propriété `#handle1` avec l'emplacement du premier point de contrôle de ce sommet et une propriété `#handle2`, avec l'emplacement du second point de contrôle de ce sommet. Pour plus d'informations, consultez `vertexList`.

Exemple

L'instruction suivante affiche un exemple de liste des sommets de la troisième courbe de l'acteur forme vectorielle CourbesSimples :

```
-- Syntaxe Lingo
put(member("CourbesSimples").curve[3])
  -- [[#vertex: point(113.0000, 40.0000), #handle1: point(32.0000, 10.0000),
  \
  #handle2: point(-32.0000, -10.0000)], [#vertex: point(164.0000, 56.0000)]]

// Syntaxe JavaScript
put(member("CourbesSimples").curve[3]);
  // [[#vertex: point(113.0000, 40.0000), #handle1: point(32.0000, 10.0000),
  #handle2: point(-32.0000, -10.0000)], [#vertex: point(164.0000, 56.0000)]]
```

L'instruction suivante déplace le premier sommet de la première courbe dans une forme vectorielle, en opérant un déplacement de 10 pixels vers le bas à droite :

```
-- Syntaxe Lingo
member(1).curve[1].vertex[1] = member(1).curve[1].vertex[1] + point(10, 10)

// Syntaxe JavaScript
member(1).curve[1].vertex[1] = member(1).curve[1].vertex[1] + point(10, 10);
```

Le code suivant déplace une image-objet vers l'emplacement du premier sommet de la première courbe d'une forme vectorielle. La propriété `originMode` de la forme vectorielle doit être définie sur `#topLeft` pour cette opération.

```
-- Syntaxe Lingo
emplacementDuSommet = member(1).curve[1].vertex[1]
emplacementDeLimageObjet = mapMemberToStage(sprite(3), emplacementDuSommet)
sprite(7).loc = spriteLoc

// Syntaxe JavaScript
var emplacementDuSommet = member(1).curve[1].vertex[1];
var emplacementDeLimageObjet = mapMemberToStage(sprite(3), vertexLoc);
sprite(7).loc = spriteLoc;
```

debug

Utilisation

`member(`*`quelActeur`*`).model(`*`quelModèle`*`).debug`

Description

Propriété 3D de modèle ; indique si la sphère de délimitation et les axes locaux du modèle sont affichés.

Exemple

L'instruction suivante donne à la propriété `debug` du modèle Chien la valeur `TRUE`.

```
member("Parc").model("Chien").debug = TRUE
```

Voir aussi

`boundingSphere`

debugPlaybackEnabled

Utilisation

```
-- Syntaxe Lingo
_player.debugPlaybackEnabled

// Syntaxe JavaScript
_player.debugPlaybackEnabled;
```

Description

Propriété de lecteur ; sous Windows, ouvre une fenêtre Messages à des fins de débogage dans Shockwave et les projections. Sur le Macintosh, un fichier journal est ouvert ; des instructions `put` y sont générées pour placer des données à des fins de débogage. Lecture/écriture.

Sous Windows, cette propriété n'a aucun effet lorsque utilisée dans l'application Director. Une fois la fenêtre Messages fermée, elle ne peut pas être rouverte dans une session Shockwave Player ou projection particulière. Si plusieurs animations comportant un contenu Shockwave utilisent ce script dans un seul navigateur, seule la première ouvrira une fenêtre Messages, qui sera liée uniquement à cette animation.

Sur le Macintosh, le fichier journal généré se trouve dans le dossier Shockwave Player, au niveau de DisqueDur/Dossier Système/Extensions/Macromedia/Shockwave.

Pour ouvrir la fenêtre Messages, donnez à la propriété `debugPlaybackEnabled` la valeur `TRUE`. Pour fermer la fenêtre Messages, donnez à la propriété `debugPlaybackEnabled` la valeur `FALSE`.

Exemple

Cette instruction provoque l'ouverture de la fenêtre Messages dans Shockwave Player ou dans une projection :

```
-- Syntaxe Lingo
_player.debugPlaybackEnabled = TRUE

// Syntaxe JavaScript
_player.debugPlaybackEnabled = true;
```

Voir aussi

`Lecteur`, `put()`

decayMode

Utilisation

```
member(quelActeur).camera(quelleCaméra).fog.decayMode
sprite(quelleImageObjet).camera{(index)}.fog.decayMode
```

Description

Propriété 3D ; indique la façon dont la densité du brouillard évolue, d'une densité minimum à une densité maximum, lorsque la propriété `fog.enabled` de la caméra a pour valeur `TRUE`.

Les valeurs possibles de cette propriété sont les suivantes :

- `#linear` : la densité du brouillard est interpolée de façon linéaire entre `fog.near` et `fog.far`.
- `#exponential` : `fog.far` est le point de saturation ; `fog.near` est ignoré.
- `#exponential2` : `fog.near` est le point de saturation ; `fog.far` est ignoré.

Le paramètre par défaut de cette propriété est `#exponential`.

Exemple

L'instruction suivante donne à la propriété `decayMode` du brouillard de la caméra vueParDéfaut la valeur `#linear`. Si la propriété `enabled` du brouillard a pour valeur `TRUE`, la densité du brouillard augmentera de façon constante entre les distances définies par les propriétés `near` et `far` du brouillard. Si la propriété `near` a pour valeur 100 et que la propriété `far` a pour valeur 1000, le brouillard commencera à 100 unités d'univers devant la caméra et augmentera régulièrement en densité jusqu'à une distance de 1000 unités d'univers devant la caméra.

```
member("Univers 3D").camera("vueParDéfaut").fog.decayMode = #linear
```

Voir aussi

`fog`, `near (brouillard)`, `far (brouillard)`, `enabled (brouillard)`

defaultRect

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.defaultRect

// Syntaxe JavaScript
réfObjActeur.defaultRect;
```

Description

Propriété d'acteur ; contrôle la taille par défaut utilisée pour toutes les nouvelles images-objets créées depuis un acteur animation Flash ou forme vectorielle. Le paramètre `defaultRect` s'applique également à toutes les images-objets existantes qui n'ont pas été étirées sur la scène. Vous spécifiez les valeurs de propriété comme un rectangle Director ; par exemple, `rect(0,0,32,32)`.

La propriété d'acteur `defaultRect` est affectée par la propriété d'acteur `defaultRectMode` de l'acteur. La propriété `defaultRectMode` a toujours la valeur `#Flash` lorsqu'une animation est insérée dans une distribution, ce que signifie que le paramètre `defaultRect` original a toujours la taille de l'animation originale créée dans Flash. La définition de `defaultRect` après ce stade affecte implicitement la valeur `#fixed` à la propriété `defaultRectMode` de l'acteur.

Cette propriété peut être testée et définie.

Exemple

Le gestionnaire suivant accepte une référence de distribution et un rectangle comme paramètres. Il recherche ensuite les acteurs Flash dans la distribution spécifiée et règle leur propriété `defaultRect` sur le rectangle spécifié.

```
-- Syntaxe Lingo
on réglerLeRectFlashParDéfaut(quelleDistribution, quelRect)
  repeat with i = 1 to castLib(quelleDistribution).member.count
    if member(i, quelleDistribution).type = #flash then
      member(i, quelleDistribution).defaultRect = quelRect
    end if
  end repeat
end

// Syntaxe JavaScript
function réglerLeRectFlashParDéfaut(quelleDistribution, quelRect) {
  var i = 1;
  while( i < (castLib(quelleDistribution).member.count) + 1) {
    var tp = member(i, quelleDistribution).toString();
    if (tp = "#flash") {
      member(i, quelleDistribution).defaultRect = whichRect;
      i++;
    }
  }
}
```

Voir aussi

`defaultRectMode`, `flashRect`

defaultRectMode

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.defaultRectMode

// Syntaxe JavaScript
réfObjActeur.defaultRectMode;
```

Description

Propriété d'acteur ; contrôle la méthode par laquelle la taille par défaut est définie pour toutes les nouvelles images-objets créées depuis les acteurs animation Flash ou forme vectorielle. Vous spécifiez les valeurs de propriété comme un rectangle Director ; par exemple, `rect(0,0,32,32)`.

La propriété `defaultRectMode` ne définit pas la taille réelle du rectangle par défaut d'une animation Flash, mais détermine uniquement le réglage du rectangle par défaut. La propriété `defaultRectMode` peut avoir l'une des valeurs suivantes :

- `#flash` (valeur par défaut) – Affecte au rectangle par défaut la taille de l'animation originale créée dans Flash.
- `#fixed` – Affecte au rectangle par défaut la taille fixe spécifiée par la propriété d'acteur `defaultRect`.

La propriété d'acteur `defaultRect` est affectée par la propriété d'acteur `defaultRectMode` de l'acteur. La propriété `defaultRectMode` a toujours la valeur `#flash` lorsqu'une animation est insérée dans une distribution, ce que signifie que le paramètre `defaultRect` original a toujours la taille de l'animation originale créée dans Flash. La définition de `defaultRect` après ce stade affecte implicitement la valeur `#fixed` à la propriété `defaultRectMode` de l'acteur.

Cette propriété peut être testée et définie.

Exemple

Le gestionnaire suivant accepte une référence de distribution et un rectangle comme paramètres. Il recherche ensuite les acteurs Flash dans la distribution spécifiée, règle leur propriété `defaultRectMode` sur `#fixed`, puis affecte à leur propriété `defaultRect` la valeur `rect(0,0,320,240)`.

```
-- Syntaxe Lingo
on réglerLaTailleParDéfautDeRect(quelleDistribution)
  repeat with i = 1 to castLib(quelleDistribution).member.count
    if member(i, quelleDistribution).type = #flash then
      member(i, quelleDistribution).defaultRectMode = #fixed
      member(i, quelleDistribution).defaultRect = rect(0,0,320,240)
    end if
  end repeat
end

// Syntaxe JavaScript
function réglerLaTailleParDéfautDeRect(quelleDistribution) {
  var i = 1;
  while( i < (castLib(quelleDistribution).member.count) + 1) {
    var tp = member(i, quelleDistribution).toString();
    if (tp = "#flash") {
      member(i, quelleDistribution).defaultRectMode = symbol("fixed");;
      member(i, quelleDistribution).defaultRect = rect(0,0,320,240);
      i++;
    }
  }
}
```

Voir aussi

`flashRect`, `defaultRect`

density

Utilisation

```
member(quelActeur).shader(quelMatériau).density
member(quelActeur).model(quelModèle).shader.density
member(quelActeur).model(quelModèle).shaderList{[index]}.\
  density
```

Description

Propriété 3D de matériau `#engraver` et `#newsprint` ; ajuste le nombre de lignes ou de points utilisés pour créer les effets de ces types de matériaux spécialisés. Plus les valeurs sont élevées plus le nombre de lignes ou de points est élevé.

Pour les matériaux `#engraver`, cette propriété ajuste le nombre de lignes utilisées pour créer l'image. La valeur peut être comprise entre 0 et 100, la valeur par défaut étant 40.

Pour les matériaux #newsprint, cette propriété ajuste le nombre de points utilisés pour créer l'image. La valeur peut être comprise entre 0 et 100, la valeur par défaut étant 45.

Exemple

L'instruction suivante donne à la propriété density du matériau matériauMoteur la valeur 10. Les lignes utilisées par ce matériau #engraver pour créer son image stylisée seront grossières et éloignées les unes des autres.

```
member("Séquence").shader("matériauMoteur").density = 10
```

L'instruction suivante donne à la propriété density du matériau gbMatériau la valeur 100. Les points utilisés par ce matériau #newsprint pour créer son image stylisée seront fins et rapprochés.

```
member("Séquence").shader("gbMatériau").density = 100
```

Voir aussi

newShader

depth (3D)

Utilisation

member(*quelActeur*).model(*quelModèle*).sds.depth

Description

Propriété 3D de fractionnement de surface (sds) ; spécifie le nombre maximum de niveaux de résolution que le modèle peut afficher lorsque le modificateur sds est utilisé.

Si les paramètres error et tension du modificateur sds sont bas, le fait d'augmenter la valeur de la propriété depth aura un effet plus prononcé sur la géométrie du modèle.

Le modificateur sds ne peut pas être utilisé avec les modificateurs inker ou toon ; vous devrez également faire preuve de prudence lorsque vous utilisez le modificateur sds avec le modificateur lod.

Exemple

L'instruction suivante donne à la propriété depth du modificateur sds du modèle Bébé la valeur 3. Si les paramètres error et tension du modificateur sds sont trop faibles, cette instruction à un effet très prononcé sur la géométrie de Bébé.

```
member("Séquence").model("Bébé").sds.depth = 3
```

Voir aussi

sds (modificateur), error, tension

depth (Bitmap)

Utilisation

```
objetImage.depth
member(quelActeur).depth
the depth of member quelActeur
```

Description

Propriété d'objet image ou d'acteur bitmap ; affiche le codage de couleur de l'objet image ou de l'acteur bitmap donné.

Depth	Nombre de couleurs
1	Noir et blanc
2	4 couleurs
4, 8	Couleurs de palettes 16 ou 256 couleurs ou niveaux de gris
16	Milliers de couleurs
32	Millions de couleurs

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante affiche le codage des couleurs de l'objet image stocké dans la variable *nouvelleImage*. Le résultat apparaît dans la fenêtre Messages.

```
-- Syntaxe Lingo
put(nouvelleImage.depth)

// Syntaxe JavaScript
trace(nouvelleImage.depth);
```

L'instruction suivante affiche le codage des couleurs de l'acteur Temple dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(member("Temple").depth)

// Syntaxe JavaScript
put(member("Temple").depth);
```

depthBufferDepth

Utilisation

```
getRendererServices().depthBufferDepth
```

Description

Propriété 3D `rendererServices` ; indique la précision du tampon de codage matériel du système de l'utilisateur. La valeur est 16 ou 24, en fonction du matériel de l'utilisateur.

Exemple

L'instruction suivante indique que la valeur `depthBufferDepth` de la carte vidéo de l'utilisateur est 16.

```
put getRendererServices().depthBufferDepth
-- 16
```

Voir aussi

`getRendererServices()`, `getHardwareInfo()`, `colorBufferDepth`

deskTopRectList

Utilisation

```
-- Syntaxe Lingo
_system.deskTopRectList

// Syntaxe JavaScript
_system.deskTopRectList;
```

Description

Propriété système ; affiche la taille, et la position sur le bureau, des moniteurs connectés à l'ordinateur. En lecture seule.

Cette propriété est pratique pour vérifier si des objets tels que des fenêtres, des images-objets et des fenêtres contextuelles apparaissent entièrement sur un écran.

Le résultat est une liste de rectangles, dans laquelle chaque rectangle indique la limite physique d'un moniteur. Les coordonnées de chaque moniteur sont relatives à l'angle supérieur gauche du moniteur 1, qui a la valeur (0,0). Le premier ensemble de coordonnées de rectangle correspond à la taille du premier moniteur. Si un second moniteur est présent, un second ensemble de coordonnées indique où les coins du second moniteur sont placés par rapport au premier moniteur.

Exemple

L'instruction suivante teste la taille des moniteurs connectés à l'ordinateur et affiche le résultat dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(_system.deskTopRectList)

// Syntaxe JavaScript
put(_system.deskTopRectList);
```

Le gestionnaire suivant indique le nombre de moniteurs du système courant :

```
-- Syntaxe Lingo
on countMonitors
  return _system.deskTopRectList
end
```

```
// Syntaxe JavaScript
function countMonitors() {
  return _system.deskTopRectList;
}
```

Voir aussi

Système

diffuse

Utilisation

```
member(quelActeur).shader(quelMatériau).diffuse
member(quelActeur).model(quelModèle).shader.diffuse
member(quelActeur).model(quelModèle).shaderList[[index]].\
  diffuse
```

Description

Propriété 3D de matériau #standard ; indique la couleur qui est fusionnée à la première texture du matériau lorsque les conditions suivantes sont réunies :

- la propriété useDiffuseWithTexture du matériau a pour valeur TRUE et, soit
- la propriété blendFunction du matériau a la valeur #add ou #multiply ou
- la propriété blendFunction du matériau a pour valeur #blend, la propriété blendSource du matériau a pour valeur #constant et la valeur de la propriété blendConstant du matériau est inférieure à 100.

La valeur par défaut de cette propriété est rgb(255, 255, 255).

Exemple

L'instruction suivante donne à la propriété diffuse du matériau Globe la valeur rgb(255, 0, 0).

```
member("mondeMystérieux").shader("Globe").diffuse = rgb(255, 0, 0)
```

Voir aussi

diffuseColor, useDiffuseWithTexture, blendFunction, blendSource, blendConstant

diffuseColor

Utilisation

```
member(quelActeur).diffuseColor
```

Description

Propriété 3D d'acteur ; indique la couleur qui est fusionnée à la première texture du premier matériau de l'acteur lorsque les conditions suivantes sont réunies :

- la propriété useDiffuseWithTexture du matériau a pour valeur TRUE et, soit
- la propriété blendFunction du matériau a la valeur #add ou #multiply ou
- la propriété blendFunction du matériau a pour valeur #blend, la propriété blendSource du matériau a pour valeur #constant et la valeur de la propriété blendConstant du matériau est inférieure à 100.

La valeur par défaut de la propriété diffuseColor est rgb(255, 255, 255).

Exemple

L'instruction suivante donne à la propriété `diffuseColor` de l'acteur Pièce la valeur `rgb(255, 0, 0)`.

```
member("Pièce").diffuseColor = rgb(255, 0, 0)
```

Voir aussi

`diffuse`, `useDiffuseWithTexture`, `blendFunction`, `blendSource`, `blendConstant`

diffuseLightMap

Utilisation

```
member(quelActeur).shader(quelMatériau).diffuseLightMap
member(quelActeur).model(quelModèle).shader.diffuseLightMap
member(quelActeur).model(quelModèle).shaderList{[index]}.\
  diffuseLightMap
```

Description

Propriété 3D de matériau `#standard` ; spécifie la texture à utiliser pour la lumière diffuse.

Les propriétés suivantes sont automatiquement définies avec cette propriété :

- La seconde couche de texture du matériau reçoit la texture que vous spécifiez.
- La valeur de `textureModeList[2]` est établie sur `#diffuse`.
- La valeur de `blendFunctionList[2]` est établie sur `#multiply`.
- La valeur de `blendFunctionList[1]` est établie sur `#replace`.

Exemple

L'instruction suivante donne la texture Ovale comme propriété `diffuseLightMap` au matériau utilisé par le modèle boîteEnVerre.

```
member("planète3D").model("boîteEnVerre").shader.diffuseLightMap = \
  member("planète3D").texture("Ovale")
```

Voir aussi

`blendFunctionList`, `textureModeList`, `glossMap`, `region`, `specularLightMap`

digitalVideoTimeScale

Utilisation

```
-- Syntaxe Lingo
_player.digitalVideoTimeScale

// Syntaxe JavaScript
_player.digitalVideoTimeScale;
```

Description

Propriété de lecteur ; détermine l'échelle temporelle, en unités par seconde, que le système utilise pour mesurer la durée des acteurs vidéo numérique. Lecture/écriture.

La propriété `digitalVideoTimeScale` peut recevoir n'importe quelle valeur.

La valeur de cette propriété détermine la fraction de seconde utilisée pour suivre la progression de la vidéo, comme indiqué dans les exemples suivants :

- 100 – l'échelle temporelle est de 1/100ème de seconde (et la séquence est mesurée à 100 unités par seconde).
- 500 – l'échelle temporelle est de 1/500ème de seconde (et la séquence est mesurée à 500 unités par seconde).
- 0 – Director utilise l'échelle temporelle de l'animation en cours de lecture.

Définissez `digitalVideoTimeScale` pour accéder précisément aux pistes en vous assurant que l'unité temporelle du système pour la vidéo est un multiple de l'unité temporelle de la vidéo numérique. Définissez la propriété `digitalVideoTimeScale` sur une valeur supérieure pour permettre un contrôle plus précis de la lecture de la vidéo.

Exemple

L'instruction suivante définit l'échelle temporelle utilisée par le système pour mesurer la vidéo numérique à 600 unités par seconde :

```
-- Syntaxe Lingo
_player.digitalVideoTimeScale = 600

// Syntaxe JavaScript
_player.digitalVideoTimeScale = 600;
```

Voir aussi

Lecteur

digitalVideoType

Utilisation

```
member(quelActeur).digitalVideoType
the digitalVideoType of member quelActeur
```

Description

Propriété d'acteur ; indique le format de la vidéo numérique spécifiée. Les valeurs possibles sont `#quickTime` ou `#videoForWindows`.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante teste si l'acteur Evénements du jour est une vidéo numérique QuickTime ou AVI et affiche le résultat dans la fenêtre Messages :

```
put member("Evénements du jour").digitalVideoType
```

Voir aussi

quickTimeVersion()

direction

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  emitter.direction
```

Description

Propriété 3D d'émission ; vecteur indiquant la direction dans laquelle les particules d'un système sont émises. Un système de particules est une ressource de modèle de type `#particle`.

La principale direction de l'émission des particules est le vecteur défini par la propriété `direction` de l'émetteur. Toutefois, la direction de l'émission d'une particule donnée déviera de ce vecteur selon un angle aléatoire compris entre 0 et la valeur de la propriété `angle (3D)` de l'émetteur.

Le fait de donner à la `direction` la valeur `vector(0,0,0)` provoque l'émission des particules dans toutes les directions.

La valeur par défaut de cette propriété est `vector(1.0,0)`.

Exemple

Dans l'exemple suivant, systèmeThermique est une ressource de modèle de type `#particle`. L'instruction suivante donne à la propriété `direction` de l'émetteur de systèmeThermique la valeur `vector(1, 0, 0)`, ce qui provoque l'émission des particules de systèmeThermique dans une région conique dont l'axe est l'axe des x de l'univers 3D.

```
member("Feux").modelResource("systèmeThermique").emitter.\
  direction = vector(1,0,0)
```

Voir aussi

`emitter`, `angle (3D)`

directionalColor

Utilisation

```
member(quelActeur).directionalColor
```

Description

Propriété 3D d'acteur ; indique la couleur RVB de la lumière directionnelle par défaut de l'acteur.

La valeur par défaut de cette propriété est `rgb(255, 255, 255)`.

Exemple

L'instruction suivante donne à la propriété `directionalColor` de l'acteur Pièce la valeur `rgb(0, 255, 0)`. La lumière directionnelle par défaut de l'acteur sera verte. Cette propriété peut également être définie dans l'inspecteur des propriétés.

```
member("Pièce").directionalcolor = rgb(0, 255, 0)
```

Voir aussi

`directionalPreset`

directionalPreset

Utilisation

```
member(quelActeur).directionalPreset
```

Description

Propriété 3D d'acteur ; indique la direction depuis laquelle provient la lumière directionnelle par défaut, par rapport à la caméra de l'image-objet.

La modification de la valeur de cette propriété entraîne la modification des propriétés `position` et `rotation` de la transformation de la lumière.

Les valeurs possibles de `directionalPreset` sont les suivantes :

- `#topLeft`
- `#topCenter`
- `#topRight`
- `#middleLeft`
- `#middleCenter`
- `#middleRight`
- `#bottomLeft`
- `#bottomCenter`
- `#bottomRight`
- `#None`

La valeur par défaut de cette propriété est `#topCenter`.

Exemple

L'instruction suivante donne à la propriété `directionalPreset` de l'acteur Pièce la valeur `#middleCenter`. La lumière par défaut de Pièce est pointée vers le centre de la vue courante de la caméra de l'image-objet. Cette propriété peut également être définie dans l'inspecteur des propriétés.

```
member("Pièce").directionalpreset = #middleCenter
```

Voir aussi

`directionalColor`

directToStage

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.directToStage

// Syntaxe JavaScript
réfObjActeurOuImageObjet.directToStage;
```

Description

Propriété d'acteur et d'image-objet ; détermine le plan sur lequel un acteur vidéo numérique, GIF animé, forme vectorielle, 3D, Windows Media ou Flash est lu.

Si cette propriété a pour valeur `TRUE`, l'acteur est lu devant toutes les autres couches de la scène et les effets d'encre n'ont aucun effet.

Si cette propriété a pour valeur `FALSE`, l'acteur peut apparaître dans n'importe quelle couche des plans d'animation de la scène et les effets d'encre affectent l'apparence de l'image-objet.

- Utilisez la syntaxe `member(`*`quelActeur`*`).directToStage` pour les vidéos numériques ou les GIF animés.
- Utilisez la syntaxe `sprite(`*`quelleImageObjet`*`).directToStage` pour Flash ou les formes vectorielles.
- Utilisez n'importe quelle syntaxe pour les acteurs ou les images-objets 3D.

L'utilisation de cette propriété améliore les performances de lecture des acteurs ou images-objets.

Aucun autre acteur ne peut apparaître devant une image-objet `directToStage`. Les effets d'encre n'ont également aucun effet sur l'apparence d'une image-objet `directToStage`.

Lorsque la propriété `directToStage` d'une image-objet a pour valeur `TRUE`, Director dessine l'image-objet directement à l'écran sans la composer d'abord dans son tampon hors-écran. Le résultat peut être semblable à un effet de traces sur la scène.

Vous pouvez rafraîchir une zone contenant des traces en activant/désactivant la propriété `directToStage`, à l'aide d'une transition plein-écran ou du passage d'une autre image-objet sur cette zone. Sous Windows, si vous ne le faites pas, vous pouvez passer à un écran semblable sans que la vidéo disparaisse complètement.

Vous pourrez voir un exemple de `directToStage` dans une animation en consultant l'animation QT et Flash du dossier Learning/Lingo, lui-même dans le dossier de Director.

Exemple

L'instruction suivante entraîne la lecture de l'animation QuickTime Résidents sur la couche supérieure de la scène :

```
-- Syntaxe Lingo
member("Résidents").directToStage = 1

// Syntaxe JavaScript
member("Résidents").directToStage = 1;
```

disableImagingTransformation

Utilisation

```
-- Syntaxe Lingo
_player.disableImagingTransformation

// Syntaxe JavaScript
_player.disableImagingTransformation;
```

Description

Propriété de lecteur ; détermine si Director prend automatiquement en compte le défilement ou le zoom sur la scène, en capturant l'image de la scène. Lecture/écriture.

Lorsque cette propriété a pour valeur TRUE, elle empêche Director de prendre automatiquement en compte le défilement ou le zoom sur la scène lorsque la propriété image est utilisée pour obtenir l'image de la scène. Le zoom et le défilement de la scène affectent l'apparence de l'image capturée par image.

Lorsque cette propriété a pour valeur FALSE, Director capture toujours l'image de la scène comme si la fenêtre de la scène était à 100 % et n'était pas déplacée du centre de la fenêtre de la scène. FALSE est la valeur par défaut.

Exemple

L'instruction suivante donne à la propriété disableImagingTransformation la valeur TRUE :

```
-- Syntaxe Lingo
_player.disableImagingTransformation = TRUE

// Syntaxe JavaScript
_player.digitalVideoTimeScale = true;
```

Voir aussi

image (image), Lecteur

displayFace

Utilisation

```
member(quelActeurTexte).displayFace
member(quelActeur3D).modelResource(quelleRessourceDeModèle).\
  displayFace
```

Description

Propriété 3D de texte ; liste linéaire indiquant la ou les faces du texte 3D à afficher. Les valeurs possibles sont #front, #tunnel et #back. Vous pouvez afficher n'importe quelle combinaison de faces et la liste peut être dans n'importe quel ordre.

La valeur par défaut de cette propriété est [#front, #back, #tunnel].

Pour les acteurs texte, il s'agit d'une propriété d'acteur. Pour le texte extrudé d'un acteur 3D, il s'agit d'une propriété de ressource de modèle.

Exemple

Dans l'exemple suivant, l'acteur Panneau est un acteur texte. L'instruction suivante donne à la propriété displayFace de Panneau la valeur [#tunnel]. Lorsque Panneau est affiché en mode 3D, ses faces avant et arrière n'apparaissent pas.

```
member("Panneau").displayFace = [#tunnel]
```

Dans l'exemple suivant, la ressource du modèle Slogan est du texte extrudé. L'instruction suivante donne à la propriété displayFace de la ressource de modèle de Slogan la valeur [#back, #tunnel]. La face avant de Slogan ne sera pas tracée.

```
member("Séquence").model("Slogan").resource.displayFace = \
  [#back, #tunnel]
```

Voir aussi

extrude3D, displayMode

displayMode

Utilisation

```
member(quelActeurTexte).displayMode
```

Description

Propriété d'acteur texte ; spécifie si le texte sera rendu en 2D ou en 3D.

Lorsque cette propriété a pour valeur #Mode3D, le texte est affiché en 3D. Vous pouvez définir les propriétés 3D du texte (telles que displayFace et bevelDepth), ainsi que les propriétés de texte habituelles (telles que text et font). L'image-objet contenant cet acteur devient une image-objet 3D.

Lorsque cette propriété a pour valeur #ModeNormal, le texte est affiché en 2D.

La valeur par défaut de cette propriété est #ModeNormal.

Exemple

Dans l'exemple suivant, l'acteur Logo est un acteur texte. L'instruction suivante entraîne l'affichage de Logo en 3D.

```
member("Logo").displayMode = #mode3D
```

Voir aussi

extrude3D

displayRealLogo

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.displayRealLogo

// Syntaxe JavaScript
réfObjActeurOuImageObjet.displayRealLogo;
```

Description

Propriété d'acteur ou image-objet RealMedia ; permet de savoir ou de définir si le logo RealNetworks est affiché (TRUE) ou non (FALSE). Lorsque cette propriété a pour valeur TRUE, le logo RealNetworks apparaît au début du flux et lorsque la vidéo est arrêtée ou rembobinée.

La valeur par défaut de cette propriété est TRUE. Les valeurs entières autres que 1 ou 0 sont traitées comme TRUE.

Exemple

Les exemples suivants indiquent que la propriété displayRealLogo de l'image-objet 2 et de l'acteur Real a pour valeur TRUE, ce qui signifie que le logo RealNetworks est affiché au début de la lecture de l'animation et lorsque cette dernière est arrêtée ou rembobinée.

```
-- Syntaxe Lingo
put(sprite(2).displayRealLogo) -- 1
put(member("Real").displayRealLogo) -- 1

// Syntaxe JavaScript
trace(sprite(2).displayRealLogo); // 1
put(member("Real").displayRealLogo); // 1
```

Les exemples suivants donnent à la propriété `displayRealLogo` de l'image-objet 2 et de l'acteur Real la valeur `FALSE`, ce qui signifie que le logo RealNetworks ne sera pas affiché.

```
-- Syntaxe Lingo
sprite(2).displayRealLogo = FALSE
member("Real").displayRealLogo = FALSE

// Syntaxe JavaScript
sprite(2).displayRealLogo = 0;
member("Real").displayRealLogo = 0;
```

displayTemplate

Utilisation

```
-- Syntaxe Lingo
_movie.displayTemplate

// Syntaxe JavaScript
_movie.displayTemplate;
```

Description

Propriété d'animation ; donne accès à une liste de propriétés qui sont appliquées à la fenêtre dans laquelle une animation est lue. Lecture/écriture.

La propriété `displayTemplate` donne accès aux propriétés de l'objet Window qui sont utilisées pour spécifier les paramètres par défaut de la fenêtre. La propriété `displayTemplate` est donc utilisée sur l'objet Animation pour renvoyer ou définir les paramètres par défaut de la fenêtre. Elle fonctionne de la même façon que les propriétés `appearanceOptions` et `titlebarOptions` sur l'objet Window.

La propriété `displayTemplate` donne accès aux propriétés suivantes.

Propriété	Description
`appearanceOptions`	Une liste de propriétés qui répertorie les options d'aspect d'une fenêtre. Les options d'aspect sont `mask`, `border`, `metal`, `dragRegionMask`, `shadow` et `liveresize`. Pour plus d'informations, consultez `appearanceOptions`.
`dockingEnabled`	Détermine si une animation dans une fenêtre (MIAW) est ancrable si elle est ouverte au cours de la programmation. Si sa valeur est `TRUE`, la fenêtre peut être ancrée. Si sa valeur est `FALSE`, la fenêtre ne peut pas être ancrée. La valeur par défaut est `FALSE`. Pour plus d'informations, consultez `dockingEnabled`.
`resizable`	Détermine si une fenêtre est redimensionnable. Si sa valeur est `TRUE`, la fenêtre est redimensionnable. Si sa valeur est `FALSE`, la fenêtre ne l'est pas. La valeur par défaut est `TRUE`. Pour plus d'informations, consultez `resizable`.
`title`	Renvoie ou définit le titre du gabarit d'affichage. Pour plus d'informations, consultez `title`.
`titlebarOptions`	Une liste de propriétés qui répertorie les options de barre de titre d'une fenêtre. Les options de barre de titre sont `icon`, `visible`, `closebox`, `minimizebox`, `maximizebox` et `sideTitlebar`. Pour plus d'informations, consultez `titlebarOptions`.
`systemTrayIcon`	(Microsoft Windows uniquement) Détermine si une icône est associée à une fenêtre dans la barre d'état système du bureau d'un utilisateur.

Propriété	Description
`systemTrayTooltip`	(Microsoft Windows uniquement) Détermine la chaîne qui s'affiche dans l'info-bulle de l'icône de la barre d'état système.
`type`	Renvoie ou définit le type d'une fenêtre. Si le type d'une fenêtre est défini, toutes les propriétés appartenant à ce type de fenêtre le sont également. Les types de fenêtres sont `tool`, `document` et `dialog`. Pour plus d'informations, consultez `type`.

Exemple

Ces instructions affichent les propriétés `displayTemplate` et les valeurs correspondantes dans la fenêtre Messages.

```
-- Syntaxe Lingo
trace(_movie.displayTemplate)

// Syntaxe JavaScript
trace(_movie.displayTemplate);
```

Ces instructions définissent diverses propriétés `displayTemplate`.

```
-- Syntaxe Lingo
_movie.displayTemplate.dockingEnabled = TRUE
_movie.displayTemplate.resizable = FALSE
_movie.displayTemplate.appearanceOptions.mask = member("masque")
_movie.displayTemplate.titlebarOptions.sideTitlebar = TRUE

// Syntaxe JavaScript
_movie.displayTemplate.dockingEnabled = true;
_movie.displayTemplate.resizable = false;
_movie.displayTemplate.appearanceOptions.mask = member("masque");
_movie.displayTemplate.titlebarOptions.sideTitlebar = true;
```

Voir aussi

`appearanceOptions`, `dockingEnabled`, `Animation`, `resizable`, `systemTrayIcon`, `title (fenêtre)`, `titlebarOptions`, `type (fenêtre)`, `Fenêtre`

distribution

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  emitter.distribution
```

Description

Propriété 3D d'émission ; indique la façon dont les particules d'un système de particules sont réparties dans la région de l'émetteur lors de leur création. Les valeurs possibles de cette propriété sont `#gaussian` ou `#linear`. La valeur par défaut est `#linear`.

Exemple

Dans l'exemple suivant, systèmeThermique est une ressource de modèle de type `#particle`. L'instruction suivante donne à la propriété `distribution` de l'émetteur de systèmeThermique la valeur `#linear`, ce qui entraîne la répartition uniforme des particules de systèmeThermique dans leur région d'origine lorsqu'elles sont créées.

```
member("Feux").modelResource("systèmeThermique").emitter.\
  distribution = #linear
```

Voir aussi

`emitter`, `region`

dither

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.dither

// Syntaxe JavaScript
réfObjActeur.dither;
```

Description

Propriété d'acteur bitmap ; trame l'acteur lorsqu'il est affiché avec un codage de couleur sur 8 bits ou moins (256 couleurs) si l'écran doit afficher une nuance de couleurs qui n'est pas dans l'acteur (`TRUE`) ou indique à Director de choisir la couleur la plus proche parmi celles de la palette courante (`FALSE`).

Pour des raisons de performance comme de qualité, vous ne devriez donner à `dither` la valeur `TRUE` que lorsqu'une qualité d'affichage supérieure est nécessaire. Le tramage est plus lent qu'une conversion des couleurs et des détails ennuyeux peuvent être plus apparents lors de l'animation sur une image tramée.

Si le codage des couleurs est supérieur à 8 bits, cette propriété n'a aucun effet.

Voir aussi

`depth (Bitmap)`

dockingEnabled

Utilisation

```
-- Syntaxe Lingo
_movie.displayTemplate.dockingEnabled
réfObjFenêtre.dockingEnabled

// Syntaxe JavaScript
_movie.displayTemplate.dockingEnabled;
réfObjFenêtre.dockingEnabled;
```

Description

Propriété d'animation et de fenêtre ; indique si une animation dans une fenêtre (MIAW) est une fenêtre ancrable si elle est ouverte au cours de la programmation. Lecture/écriture.

Vous ne pouvez pas accéder à cette propriété directement à partir d'un objet Animation, mais à partir de la propriété `displayTemplate` de cet objet.

La valeur par défaut de cette propriété est `FALSE` ; elle indique qu'une fenêtre MIAW ne sera pas ancrable si elle est ouverte au cours de la programmation. Lorsque cette propriété a pour valeur `TRUE`, la valeur de la propriété `type` de l'objet Window détermine le mode d'affichage de la fenêtre lors de la programmation.

- Si `dockingEnabled` a pour valeur `TRUE` et que `type` a pour valeur `#document`, l'aspect de la fenêtre MIAW, tout comme son comportement, sera similaire à celui des fenêtres de type Document de Director. La fenêtre apparaîtra dans la zone de « l'espace principal » et sera ancrable avec la fenêtre Scène, Scénario et Distribution, les éditeurs de médias et les fenêtres Messages. Elle ne pourra cependant être regroupée avec aucune de ces fenêtres.
- Si `dockingEnabled` a pour valeur `TRUE` et que `type` a pour valeur `#tool`, l'aspect de la fenêtre MIAW, tout comme son comportement, sera similaire à celui des fenêtres de type Outil de Director. La fenêtre pourra se regrouper avec toutes les fenêtres de type Outil, à l'exception de l'inspecteur des propriétés et de la palette des outils.
- Si `dockingEnabled` a pour valeur `TRUE` et que `type` a pour valeur `#dialog`, le type sera ignoré et la fenêtre sera une fenêtre de programmation.

Cette propriété est ignorée dans les projections.

Exemple

Les instructions suivantes donnent à la propriété `dockingEnabled` la valeur `TRUE`.

```
-- Syntaxe Lingo
_movie.displayTemplate.dockingEnabled = TRUE - à partir de l'objet Animation
window("Instructions").dockingEnabled = TRUE - à partir de l'objet Fenêtre

// Syntaxe JavaScript
_movie.displayTemplate.dockingEnabled = true; // à partir de l'objet Animation
window("Instructions").dockingEnabled = true; // à partir de l'objet Fenêtre
```

Voir aussi

`appearanceOptions`, `displayTemplate`, `Animation`, `titlebarOptions`, `type (fenêtre)`, `Fenêtre`

domain

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.domain

// Syntaxe JavaScript
réfObjDvd.domain;
```

Description

Propriété DVD ; renvoie un symbole qui désigne l'image courante. En lecture seule.

Voir aussi

`DVD`

doubleClick

Utilisation

```
-- Syntaxe Lingo
_mouse.doubleClick

// Syntaxe JavaScript
_mouse.doubleClick;
```

Description

Propriété de souris ; indique si deux clics de souris dans un laps de temps prédéfini comme un double-clic correspondent effectivement à un double-clic plutôt qu'à deux clics simples (`TRUE`) ou, s'ils n'ont pas eu lieu dans le temps déterminé, les traite comme des clics simples (`FALSE`). En lecture seule.

Exemple

L'instruction suivante envoie la tête de lecture sur l'image Entrée de l'offre lorsque l'utilisateur double-clique sur le bouton de la souris :

```
-- Syntaxe Lingo
if (_mouse.doubleClick) then
  _movie.go("Entrée de l'offre")
end if

// Syntaxe JavaScript
if (_mouse.doubleClick) {
  _movie.go("Entrée de l'offre");
}
```

Voir aussi

`clickLoc`, `clickOn`, `Souris`

drag

Utilisation

`member(`*`quelActeur`*`).modelResource(`*`quelleRessourceDeModèle`*`).drag`

Description

Propriété de la ressource de modèle `#particle` ; indique le pourcentage de vélocité de chaque particule qui est perdu à chaque étape de simulation. La valeur de cette propriété peut être comprise entre 0 (aucune perte de vélocité) et 100 (toute la vélocité est perdue et la particule arrête de bouger). La valeur par défaut est 0.

Exemple

Dans l'exemple suivant, systèmeThermique est une ressource de modèle de type `#particle`. L'instruction suivante donne à la propriété `drag` de systèmeThermique la valeur 5, ce qui applique une forte résistance au mouvement des particules de systèmeThermique et les empêche de se déplacer très loin.

```
member("Feux").modelResource("systèmeThermique").drag = 5
```

Voir aussi

`wind`, `gravity`

drawRect

Utilisation

```
-- Syntaxe Lingo
réfObjFenêtre.drawRect

// Syntaxe JavaScript
réfObjFenêtre.drawRect;
```

Description

Propriété de fenêtre ; identifie les coordonnées rectangulaires de la scène de l'animation qui apparaît dans une fenêtre. Lecture/écriture.

Les coordonnées sont données sous la forme d'un rectangle, avec les entrées dans l'ordre gauche, haut, droite et bas.

Cette propriété est pratique pour mettre les animations à l'échelle ou faire un panorama, mais ne redimensionne pas les acteurs texte et champ. La mise à l'échelle des bitmaps peut affecter la performance.

Exemple

L'instruction suivante affiche les coordonnées courantes de la fenêtre d'animation intitulée Tableau de commande :

```
-- Syntaxe Lingo
put(window("Tableau de commande").drawRect)

// Syntaxe JavaScript
put(window("Tableau de commande").drawRect);
```

L'instruction suivante donne au rectangle de l'animation les valeurs du rectangle intitulé rectangleDanimation. La partie de l'animation contenue dans le rectangle correspond à la zone affichée dans la fenêtre.

```
-- Syntaxe Lingo
rectangleDanimation = rect(10, 20, 200, 300)
window("Tableau de commande").drawRect = rectangleDanimation

// Syntaxe JavaScript
var rectangleDanimation = rect(10, 20, 200, 300);
window("Tableau de commande").drawRect = rectangleDanimation;
```

Les lignes suivantes entraînent le remplissage de la zone principale du moniteur par la scène :

```
-- Syntaxe Lingo
_movie.stage.drawRect = _system.deskTopRectList[1]
_movie.stage.rect = _system.deskTopRectList[1]

// Syntaxe JavaScript
_movie.stage.drawRect = _system.deskTopRectList[1];
_movie.stage.rect = _system.deskTopRectList[1];
```

Voir aussi

rect(), Fenêtre

dropShadow

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.dropShadow

// Syntaxe JavaScript
réfObjActeur.dropShadow;
```

Description

Propriété d'acteur ; détermine la taille de l'ombre, en pixels, du texte d'un acteur champ.

Exemple

L'instruction suivante donne à l'ombre de l'acteur champ Commentaire une taille de 5 pixels :

```
-- Syntaxe Lingo
member("Commentaire").dropShadow = 5

// Syntaxe JavaScript
member("Commentaire").dropShadow = 5;
```

duration (3D)

Utilisation

```
member(quelActeur).motion(quelMouvement).duration
référenceDobjetDeDéplacement.duration
```

Description

Propriété 3D ; permet d'obtenir le temps, en millisecondes, nécessaire à la lecture complète du mouvement spécifié dans le paramètre *quelMouvement*. Cette propriété est toujours supérieure ou égale à 0.

Exemple

L'instruction suivante indique la durée, en millisecondes, du mouvement Coup.

```
put member("GbActeur").motion("Coup").duration
-- 5100.0000
```

Voir aussi

motion, currentTime (3D), play() (3D), queue() (3D)

duration (DVD)

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.duration

// Syntaxe JavaScript
réfObjDvd.duration;
```

Description

Propriété DVD ; renvoie la durée totale du titre, en millisecondes. En lecture seule.

Voir aussi

DVD

duration (acteur)

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.duration

// Syntaxe JavaScript
réfObjActeur.duration;
```

Description

Propriété d'acteur; détermine la durée des acteurs Shockwave Audio (SWA), transition, Windows Media et QuickTime spécifiés.

- Si *réfObjActeur* est un fichier audio lu en flux continu, cette propriété indique la durée du son. La propriété `duration` renvoie 0 jusqu'au démarrage de la lecture en flux continu. Le fait de donner à `preLoadTime` la valeur d'une seconde permet le renvoi de la durée réelle.
- Si *réfObjActeur* est un acteur vidéo numérique, cette propriété indique la durée de la vidéo numérique. La valeur est exprimée en battements.
- Si *réfObjActeur* est un acteur transition, cette propriété indique la durée de la transition. La valeur de la transition est exprimée en millisecondes. Au cours de la lecture, ce paramètre a le même effet que le paramètre Durée de la boîte de dialogue Préférences de l'image : Transition.

Cette propriété peut être testée pour tous les acteurs qui la prennent en charge, mais n'est définissable que pour les transitions.

Vous pourrez voir un exemple de `duration` dans une animation en consultant l'animation QT and Flash du dossier Learning/Lingo, lui-même dans le dossier de Director.

Exemple

Si l'acteur SWA Louie Prima a été préchargé, l'instruction suivante affiche la durée du son dans l'acteur champ Affichage de la durée :

```
-- Syntaxe Lingo
on exitFrame
  if member("Louie Prima").state = 2 then
    member("Affichage de la durée").text =  \
  string(member("Louie Prima").duration)
  end if
end
```

```
// Syntaxe JavaScript
function exitFrame() {
  if (member("Louie Prima").state == 2) {
    member("Affichage de la durée").text =
  member("Louie Prima").duration.toString()
  }
}
```

duration (RealMedia, SWA)

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.duration

// Syntaxe JavaScript
réfObjActeurOuImageObjet.duration;
```

Description

Propriété d'acteur ou d'image-objet audio RealMedia ou Shockwave ; renvoie la durée d'un flux RealMedia ou Shockwave Audio, en millisecondes. La durée du flux reste inconnue jusqu'au démarrage de la lecture de l'acteur. Si le flux provient d'un flux en direct ou n'a pas été lu, la valeur de cette propriété est 0. Cette propriété peut être testée, mais pas définie.

Exemple

Les exemples suivants indiquent que l'état du flux RealMedia de l'image-objet 2 et de l'acteur Real est 100 500 millisecondes (100,50 secondes).

```
-- Syntaxe Lingo
put(sprite(2).duration) -- 100500
put(member("Real").duration) -- 100500

// Syntaxe JavaScript
put(sprite(2).duration); // 100500
put(member("Real").duration); // 100500
```

Voir aussi

play() (RealMedia, SWA, Windows Media), seek(), currentTime (RealMedia)

editable

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.editable

// Syntaxe JavaScript
réfObjImageObjet.editable;
```

Description

Propriété d'image-objet ; détermine si une image-objet spécifiée peut être modifiée sur la scène (TRUE) ou non (FALSE). Lecture/écriture.

Lorsque la propriété d'acteur est définie, le paramètre est appliqué à toutes les images-objets contenant le champ.

Lorsque cette propriété est définie, seule l'image-objet spécifiée est affectée.

Vous pouvez également rendre une image-objet champ modifiable à l'aide de l'option Modifiable dans la boîte de dialogue Propriétés de l'acteur champ.

Vous pouvez rendre une image-objet champ modifiable à l'aide de l'option Modifiable dans le scénario.

Pour que la valeur définie par un script dure au-delà de l'image-objet courante, un script doit être affecté à l'image-objet.

Exemple

Le gestionnaire suivant fait de la piste d'image-objet un esclave et rend l'image-objet champ modifiable :

```
-- Syntaxe Lingo
on mesNotes
  _movie.puppetSprite(5, TRUE)
  sprite(5).editable = TRUE
end

// Syntaxe JavaScript
function mesNotes() {
  _movie.puppetSprite(5, true);
  sprite(5).editable = true;
}
```

L'instruction suivante vérifie si une image-objet champ est modifiable et affiche un message le cas échéant :

```
-- Syntaxe Lingo
if (sprite(13).editable = TRUE) then
  member("Notice").text = "Veuillez saisir votre réponse ci-dessous."
end if

// Syntaxe JavaScript
if (sprite(13).editable == true) {
  member("Notice").text = "Veuillez saisir votre réponse ci-dessous.";
}
```

Voir aussi

Image-objet

editShortCutsEnabled

Utilisation

```
-- Syntaxe Lingo
_movie.editShortCutsEnabled

// Syntaxe JavaScript
_movie.editShortCutsEnabled;
```

Description

Propriété d'animation ; détermine si les opérations Couper, Copier et Coller, ainsi que leurs raccourcis clavier, fonctionnent dans l'animation courante. Lecture/écriture.

Ces opérations de texte fonctionnent si elles sont définies sur `TRUE`. Ces opérations ne sont pas autorisées si elles sont définies sur `FALSE`. La valeur par défaut est `TRUE` pour les animations créées dans Director 8 et plus récent, et `FALSE` pour les animations créées dans les versions précédentes de Director.

Exemple

L'instruction suivante désactive les opérations Couper, Copier et Coller :

```
-- Syntaxe Lingo
_movie.editShortCutsEnabled = 0

// Syntaxe JavaScript
_movie.editShortCutsEnabled = 0;
```

Voir aussi

Animation

elapsedTime

Utilisation

```
-- Syntaxe Lingo
réfObjPisteAudio.elapsedTime

// Syntaxe JavaScript
réfObjPisteAudio.elapsedTime;
```

Description

Propriété de piste audio ; indique la durée écoulée, en millisecondes, depuis le début de la lecture de l'acteur son courant dans une piste audio. En lecture seule.

Le temps écoulé commence à 0 au début de la lecture du son et augmente au fur et à mesure de la lecture, indépendamment de la lecture en boucle, du paramètre `currentTime` ou de toute autre manipulation. Utilisez `currentTime` pour tester la durée absolue courante au sein du son.

La valeur de cette propriété est un nombre à virgule flottante, ce qui permet de mesurer la lecture du son en fraction de millisecondes.

Exemple

Ce gestionnaire affiche le temps écoulé pour la piste audio 4 dans un champ de la scène en période d'inactivité :

```
-- Syntaxe Lingo
on idle
  member("durée").text = string(sound(4).elapsedTime)
end idle

// Syntaxe JavaScript
function idle() {
  member("position").text = sound(4).elapsedTime.toString();
}
```

Voir aussi

currentTime (image-objet), Piste audio

emissive

Utilisation

```
member(quelActeur).shader(quelMatériau).emissive
member(quelActeur).model(quelModèle).shader.emissive
member(quelActeur).model(quelModèle).shaderList[[index]].\
  emissive
```

Description

Propriété 3D de matériau `#standard` ; ajoute de la lumière au matériau indépendamment de l'éclairage de la scène. Par exemple, un modèle utilisant un matériau dont la propriété `emissive` a pour valeur `rgb(255, 255, 255)` apparaîtra comme étant illuminé par une lumière blanche, même si la scène ne contient aucune lumière. Cependant, le modèle n'éclairera par les autres modèles et n'apportera aucune lumière à la scène.

La valeur par défaut de cette propriété est `rgb(0, 0, 0)`.

Exemple

L'instruction suivante donne à la propriété `emissive` du matériau Globe la valeur `rgb(255, 0, 0)`. Les modèles utilisant ces matériaux apparaîtront comme étant éclairés par une lumière rouge.

```
member("mondeMystérieux").shader("Globe").emissive = rgb(255, 0, 0)
```

Voir aussi

`silhouettes`

emitter

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  emitter.numParticles
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  emitter.mode
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  emitter.loop
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  emitter.direction
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  emitter.region
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  emitter.distribution
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  emitter.angle
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  emitter.path
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  emitter.pathStrength
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  emitter.minSpeed
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  emitter.maxSpeed
```

Description

Elément 3D de système de particules ; contrôle la propulsion initiale de particules à partir d'une ressource de modèle de type `#particle`.

La section « Voir aussi » de cette entrée contient une liste complète des propriétés d'émetteur. Pour plus d'informations, consultez les entrées des différentes propriétés.

Voir aussi

numParticles, loop (émetteur), direction, distribution, region, angle (3D), path (3D), pathStrength, minSpeed, maxSpeed

emulateMultibuttonMouse

Utilisation

```
-- Syntaxe Lingo
_player.emulateMultibuttonMouse

// Syntaxe JavaScript
_player.emulateMultibuttonMouse;
```

Description

Propriété de lecteur ; détermine si une animation considère un clic de la souris exécuté en enfonçant la touche Ctrl d'un Macintosh comme un clic sur le bouton droit de la souris sous Windows (TRUE) ou non (FALSE, valeur par défaut). Lecture/écriture.

Les clics du bouton droit n'ont aucun équivalent direct sur Macintosh.

Le fait de donner à cette propriété la valeur TRUE vous permet d'apporter des réponses constantes aux clics de la souris dans les animations lues sur différentes plates-formes.

Exemple

Les instructions suivantes donnent à la propriété emulateMultibuttonMouse la valeur TRUE :

```
-- Syntaxe Lingo
_player.emulateMultibuttonMouse = TRUE

// Syntaxe JavaScript
_player.emulateMultibuttonMouse = true;
```

Voir aussi

Lecteur

enabled

Utilisation

```
the enabled of menuItem quelElément of menu quelMenu
```

Description

Propriété d'élément de menu ; détermine si l'élément de menu spécifié par *quelElément* est affiché en texte normal et peut être sélectionné (TRUE, valeur par défaut) ou apparaît en texte grisé et ne peut pas être sélectionné (FALSE).

L'expression *quelElément* peut être le nom d'un élément de menu ou son numéro. L'expression *quelMenu* peut être le nom d'un menu ou son numéro.

Cette propriété peut être testée et définie.

Remarque : Les menus ne sont pas disponibles dans Shockwave Player.

Exemple

Le gestionnaire suivant active ou désactive tous les éléments du menu spécifié. L'argument `leMenu` spécifie le menu ; l'argument `Paramètre` spécifie `TRUE` ou `FALSE`. Par exemple, l'instruction d'appel `activerLeMenu ("Spécial", FALSE)` désactive tous les éléments du menu Spécial.

```
on activerLeMenu leMenu, Paramètre
  set n = the number of menuItems of menu leMenu
  repeat with i = 1 to n
    set the enabled of menuItem i of menu leMenu to Paramètre
  end repeat
end activerLeMenu
```

Voir aussi

`name (propriété de menu)`, `number (menus)`, `checkMark`, `script`, `number (éléments de menu)`

enabled (collision)

Utilisation

```
member(quelActeur).model(quelModèle).collision.enabled
```

Description

Propriété 3D de collision ; permet de savoir ou de définir si les collisions sont détectées au niveau des modèles (`TRUE`) ou non (`FALSE`). La valeur `FALSE` désactive temporairement le modificateur `collision` sans le supprimer du modèle.

Le paramètre par défaut de cette propriété est `TRUE`.

Exemple

L'instruction suivante active le modificateur `collision` pour le modèle Boîte :

```
member("Univers 3D").model("Boîte").collision.enabled = TRUE
```

Voir aussi

`addModifier`, `collision (modificateur)`, `modifier`

enabled (brouillard)

Utilisation

```
member(quelActeur).camera(quelleCaméra).fog.enabled
sprite(quelleImageObjet).camera{(index)}.fog.enabled
```

Description

Propriété 3D de caméra ; indique si la caméra ajoute du brouillard à la vue. Le paramètre par défaut de cette propriété est `FALSE`.

Exemple

L'instruction suivante crée du brouillard dans la vue de la caméra vueDeLaBaie :

```
member("monTerrain").camera("vueDeLaBaie").fog.enabled = TRUE
```

Voir aussi

`fog`

enabled (sds)

Utilisation

```
member(quelActeur).model(quelModèle).sds.enabled
```

Description

Propriété 3D de modificateur `sds` ; indique si le modificateur `sds` associé au modèle est utilisé par le modèle.

Le paramètre par défaut de cette propriété est `TRUE`.

Une tentative d'ajout du modificateur `sds` à un modèle déjà associé au modificateur `inker` ou `toon` entraîne un échec sans message d'erreur. De même, une tentative d'ajout du modificateur `inker` ou `toon` à un modèle déjà associé au modificateur `sds` entraîne un échec sans message d'erreur. Faites également attention lorsque vous utilisez le modificateur `sds` avec le modificateur `lod`. Pour plus d'informations, consultez l'entrée du modificateur `sds`.

Exemple

L'instruction suivante active le modificateur `sds` associé au modèle Bébé :

```
member("Séquence").model("Bébé").sds.enabled = TRUE
```

Voir aussi

`sds (modificateur)`, `modifier`, `addModifier`

enableFlashLingo

Utilisation

```
-- Syntaxe Lingo
_movie.enableFlashLingo

// Syntaxe JavaScript
_movie.enableFlashLingo;
```

Description

Propriété d'animation ; détermine si une image-objet possédant un contenu Flash peut passer des appels de programmation directs lors de l'utilisation de la méthode `getURL()`. Lecture/écriture.

La méthode `getURL()` Flash charge une nouvelle URL dans une fenêtre de navigateur vide.

Si `enableFlashLingo` a pour valeur `TRUE`, une image-objet possédant un contenu Flash peut exécuter n'importe quelle commande de script valide (à condition de respecter les règles de sécurité Shockwave Player standard) lors de l'appel de `getURL()`.

Si `enableFlashLingo` a pour valeur `FALSE`, une image-objet possédant un contenu Flash ne peut exécuter aucune commande de script lors de l'appel `getURL()`. La valeur par défaut de cette propriété est `FALSE`.

Cette propriété est pratique pour créer une animation qui propose un contenu Flash d'origine inconnue, par exemple une projection qui recherche des fichiers SWF dans un dossier système, ou une animation possédant un contenu Shockwave et qui accepte l'URL d'un fichier SWF communiquée par un utilisateur final.

Exemple

L'instruction suivante donne à la propriété enableFlashLingo la valeur `TRUE` :

```
-- Syntaxe Lingo
_movie.enableFlashLingo = TRUE

// Syntaxe JavaScript
_movie.enableFlashLingo = true;
```

Voir aussi

`Animation`

endAngle

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  endAngle
```

Description

Propriété 3D de ressource de modèle `#cylinder` ou `#sphere` ; indique la quantité dessinée de la sphère ou du cylindre.

La surface d'une sphère est générée en balayant un arc de demi-cercle 2D autour de l'axe des y de la sphère, de `startAngle` à `endAngle`. Si `startAngle` a pour valeur 0 et `endAngle` a pour valeur 360, le résultat est une sphère complète. Pour dessiner une section de sphère, donnez à `endAngle` une valeur inférieure à 360.

La surface d'un cylindre est générée en balayant une ligne 2D autour de l'axe des y du cylindre, de `startAngle` à `endAngle`. Si `startAngle` a pour valeur 0 et `endAngle` a pour valeur 360, le résultat est un cylindre complet. Pour dessiner une section de cylindre, donnez à `endAngle` une valeur inférieure à 360.

Le paramètre par défaut de cette propriété est 360.

Exemple

Pour l'exemple suivant, l'acteur monActeur contient un modèle qui utilise la ressource de modèle sphère4, dont la valeur `endAngle` est 310, ce qui laisse une ouverture de 50°. Le gestionnaire `fermetureDeSphère` ferme cette ouverture de façon à donner l'impression d'une porte coulissante. La boucle de répétition change la valeur `endAngle` de la sphère d'un degré à la fois. La commande `updateStage` de la boucle de répétition force la mise à jour de la scène après chaque palier d'un degré.

```
on fermetureDeSphère
  monAngle = member("MonActeur").modelresource("Sphère4").endAngle
  repeat with r = 1 to 50
    monAngle = monAngle + 1
    member("MonActeur").modelresource("Sphère4").endAngle = monAngle
    updateStage
  end repeat
end
```

Voir aussi

`state (3D)`

endColor

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.endColor

// Syntaxe JavaScript
réfObjActeur.endColor;
```

Description

Propriété d'acteur forme vectorielle ; couleur finale d'un remplissage en dégradé spécifiée sous la forme d'une valeur RVB.

`endColor` est uniquement valide lorsque `fillMode` a pour valeur `#gradient` et que la couleur de départ est définie avec `fillColor`.

Cette propriété peut être testée et définie.

Vous pourrez voir un exemple de `endColor` dans une animation en consultant l'animation Vector Shapes du dossier Learning/Lingo, lui-même dans le dossier de Director.

Voir aussi

`color()`, `fillColor`, `fillMode`

endFrame

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.endFrame

// Syntaxe JavaScript
réfObjImageObjet.endFrame;
```

Description

Propriété d'image-objet ; renvoie le numéro de la dernière image de l'étendue de l'image-objet. En lecture seule.

Cette propriété est pratique pour déterminer l'étendue d'une image-objet particulière dans le scénario.

Cette propriété est uniquement disponible dans une image contenant l'image-objet. Elle ne peut pas être appliquée aux images-objets situées dans d'autres images de l'animation.

Exemple

L'instruction suivante indique quelle est l'image finale de l'image-objet de la piste 5 dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(sprite(5).endFrame)

// Syntaxe JavaScript
put(sprite(5).endFrame);
```

Voir aussi

`Image-objet`, `startFrame`

endTime

Utilisation

```
-- Syntaxe Lingo
réfObjPisteAudio.endTime

// Syntaxe JavaScript
réfObjPisteAudio.endTime;
```

Description

Propriété de piste audio ; spécifie la position temporelle de fin du son en cours, en pause ou en file d'attente. Lecture/écriture.

Il s'agit de la position temporelle, au sein de l'acteur son, à laquelle la lecture du son s'arrêtera. Il s'agit d'une valeur à virgule flottante, permettant de mesurer et de contrôler la lecture du son en fraction de millisecondes. La valeur par défaut est la fin normale du son.

Cette propriété peut être définie sur une valeur différente de la fin normale du son uniquement si elle est passée comme paramètre dans les méthodes `queue()` ou `setPlayList()`.

Exemple

Les instructions suivantes vérifient si l'acteur son Annonce est défini pour une lecture sans interruption dans la piste audio 1 :

```
-- Syntaxe Lingo
if (sound(1).startTime > 0 and sound(1).endTime < member("Annonce").duration)
  \
  then
  _player.alert("Lecture d'un son incomplet.")
end if

// Syntaxe JavaScript
if (sound(1).startTime > 0 && sound(1).endTime < member("Annonce").duration) {
  _player.alert("Lecture d'un son incomplet.");
}
```

Voir aussi

`queue()`, `setPlayList()`, `Piste audio`

environmentPropList

Utilisation

```
-- Syntaxe Lingo
_system.environmentPropList

// Syntaxe JavaScript
_system.environmentPropList;
```

Description

Propriété système ; contient une liste d'informations concernant l'environnement dans lequel le contenu Director est lu. En lecture seule.

Ceci permet à Macromedia d'ajouter ultérieurement des informations à la propriété `environmentPropList`, sans affecter les animations existantes.

Les informations sont présentées sous la forme de paires de valeurs et de propriétés pour cette zone.

#shockMachine	Valeur TRUE ou FALSE indiquant si l'animation est lue dans ShockMachine.
#shockMachineVersion	Chaîne indiquant le numéro de la version de ShockMachine installée.
#platform	Chaîne contenant "Macintosh,PowerPC" ou "Windows,32". Ceci est déterminé d'après le système d'exploitation et le matériel sur lesquels l'animation est exécutée.
#runMode	Chaîne contenant Author (environnement auteur), Projector (projection) ou Plugin (module de navigateur web). Ceci est déterminé en fonction de l'application courante dans laquelle l'animation est exécutée.
#colorDepth	Nombre entier représentant le codage des couleurs du moniteur sur lequel la scène apparaît. Les valeurs possibles sont 1, 2, 4, 8, 16 ou 32.
#internetConnected	Symbole indiquant si l'ordinateur sur lequel l'animation est lue est connecté à Internet. Les valeurs possibles sont #online et #offline.
#uiLanguage	Chaîne indiquant la langue qu'utilise l'ordinateur pour l'affichage de son interface utilisateur. Cette langue peut être différente de #osLanguage sur les systèmes équipés de kits de langues spécifiques.
#osLanguage	Chaîne indiquant la langue native du système d'exploitation de l'ordinateur.
#productBuildVersion	Chaîne indiquant le numéro de série interne de l'application de lecture.

Les propriétés contiennent exactement les mêmes informations que les propriétés et fonctions du même nom.

Exemple

L'instruction suivante affiche la liste d'environnement dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(_system.environmentPropList)

// Syntaxe JavaScript
put(_system.environmentPropList);
```

Voir aussi

Système

error

Utilisation

member(*quelActeur*).model(*quelModèle*).sds.error

Description

Propriété 3D de modificateur #sds ; indique le pourcentage d'erreurs toléré par le modificateur lors de la synthèse des détails géométriques des modèles.

Cette propriété ne fonctionne que lorsque la propriété subdivision du modèle a pour valeur #adaptive. Les propriétés tension et depth (3D) du modificateur sont combinées à la propriété error pour contrôler l'importance du fractionnement réalisé par le modificateur.

Exemple

L'instruction suivante donne à la propriété `error` du modificateur `sds` du modèle Bébé la valeur 0. Si le paramètre `tension` du modificateur est trop faible, que son paramètre `depth` est trop élevé, et que sa valeur `subdivision` est `#adaptive`, cette instruction à un effet très prononcé sur la géométrie de Bébé.

```
member("Séquence").model("Bébé").sds.error = 0
```

Voir aussi

`sds (modificateur)`, `subdivision`, `depth (3D)`, `tension`

eventPassMode

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.eventPassMode

// Syntaxe JavaScript
réfObjActeurOuImageObjet.eventPassMode;
```

Description

Propriété d'acteur et d'image-objet Flash ; contrôle le moment auquel une animation Flash passe les événements de souris aux comportements associés aux images-objets placées sous l'image-objet Flash. La propriété `eventPassMode` peut avoir les valeurs suivantes :

- `#passAlways` (valeur par défaut) – Passe toujours les événements de souris.
- `#passButton` – Passe les événements de souris uniquement lorsque l'utilisateur clique sur un bouton dans l'animation Flash.
- `#passNotButton` – Passe les événements de souris uniquement lorsque l'utilisateur clique sur un objet qui n'est pas un bouton.
- `#passNever` – Ne passe jamais les événements de souris.

Cette propriété peut être testée et définie.

Exemple

Le script d'image suivant vérifie si les boutons d'une image-objet d'animation Flash sont activés et, le cas échéant, donne à `eventPassMode` la valeur `#passNotButton` ; si les boutons sont désactivés, le script donne à `eventPassMode` la valeur `#passAlways`. Ce script a l'effet suivant :

- Les événements de souris sur des objets autres que des boutons sont toujours passés aux scripts d'images-objets.
- Les événements de souris sur des objets boutons sont passés aux scripts d'images-objets lorsque les boutons sont désactivés. Lorsque les boutons sont activés, les événements de souris sur les boutons sont arrêtés.

```
-- Syntaxe Lingo
on enterFrame
  if sprite(5).buttonsEnabled = TRUE then
    sprite(5).eventPassMode= #passNotButton
  else
    sprite(5).eventPassMode = #passAlways
  end if
end
```

```
// Syntaxe JavaScript
function enterFrame() {
  var btEn = sprite(5).buttonsEnabled;
  if (btEn = 1) {
    sprite(5).eventPassMode= symbol("passNotButton");
  } else {
    sprite(5).eventPassMode = symbol("passAlways");
  }
}
```

exitLock

Utilisation

```
-- Syntaxe Lingo
_movie.exitLock

// Syntaxe JavaScript
_movie.exitLock;
```

Description

Propriété d'animation ; détermine si un utilisateur peut quitter et revenir au bureau Windows ou au Finder Macintosh à partir de projections (`FALSE`, valeur par défaut) ou non (`TRUE`). Lecture/écriture.

L'utilisateur peut quitter et revenir au bureau en appuyant sur Ctrl+point (Windows) ou Cmd+point (Macintosh), Ctrl+Q (Windows) ou Cmd+Q (Macintosh) ou Ctrl+W (Windows) ou Cmd+W (Macintosh) (la touche d'échappement est également supportée sous Windows).

Exemple

L'instruction suivante donne à la propriété `exitLock` la valeur `TRUE` :

```
-- Syntaxe Lingo
_movie.exitLock = TRUE

// Syntaxe JavaScript
_movie.exitLock = true;
```

Si `exitLock` est défini sur `TRUE`, rien ne se produira automatiquement lors de l'utilisation des raccourcis clavier Ctrl+point/Q/W, Echap ou Cmd+point/Q/W. Ce gestionnaire vérifie les informations saisies sur le clavier pour exécuter une sortie et présente une séquence personnalisée à l'utilisateur :

```
-- Syntaxe Lingo
on checkExit
  if ((_key.commandDown) and (_key.key = "." or _key.key = "q") and \
  (_movie.exitLock = TRUE)) then _movie.go("séquence de sortie")
end checkExit
```

```
// Syntaxe JavaScript
function checkExit() {
  if ((_key.commandDown) && (_key.key == "." | _key.key == "q") &&
  (_movie.exitLock == true)) {
    _movie.go("séquence de sortie");
  }
}
```

Voir aussi

Animation

externalParamCount

Utilisation

```
-- Syntaxe Lingo
_player.externalParamCount

// Syntaxe JavaScript
_player.externalParamCount;
```

Description

Propriété de lecteur ; renvoie le nombre de paramètres passés par une balise HTML <EMBED> ou <OBJECT> à une animation possédant un contenu Shockwave. En lecture seule.

Cette propriété est valide uniquement pour les animations possédant un contenu Shockwave exécutées dans un navigateur web. Elle ne fonctionne pas pour les animations en cours de création ou dans des projections.

Pour plus d'informations sur les paramètres externes valides, consultez `externalParamName()` et `externalParamValue()`.

Exemple

Le gestionnaire suivant détermine si une balise <OBJECT> ou <EMBED> passe des paramètres externes à une animation possédant un contenu Shockwave et exécute des instructions Lingo si les paramètres sont passés :

```
-- Syntaxe Lingo
if (_player.externalParamCount > 0) then
  -- une action quelconque
end if

// Syntaxe JavaScript
if (_player.externalParamCount > 0) {
  // une action quelconque;
}
```

Voir aussi

externalParamName(), externalParamValue(), Lecteur

face

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).\
   face.count
member(quelActeur).modelResource(quelleRessourceDeModèle).\
   face[index].colors
member(quelActeur).modelResource(quelleRessourceDeModèle).\
   face[index].normals
member(quelActeur).modelResource(quelleRessourceDeModèle).\
   face[index].shader
member(quelActeur).modelResource(quelleRessourceDeModèle).\
   face[index].textureCoordinates
member(quelActeur).modelResource(quelleRessourceDeModèle).\
   face[index].vertices
member(quelActeur).model(quelModèle).meshdeform.\
   face.count
member(quelActeur).model(quelModèle).meshdeform.\
   mesh[index].face.count
member(quelActeur).model(quelModèle).meshdeform.\
   mesh[indexDeMaille].face[indexDeFaces]
member(quelActeur).model(quelModèle).meshdeform.\
   mesh[indexDeMaille].face[indexDeFaces].neighbor{[indexDeVoisinage]}
```

Description

Propriété 3D de ressource de modèle #mesh et de modificateur meshdeform. Toutes les ressources de modèle sont des mailles composées de triangles. Chaque triangle est une face.

Vous pouvez accéder aux propriétés des faces des ressources de modèle de type #mesh. Les modifications apportées à ces propriétés n'entrent pas en vigueur jusqu'à l'appel de la commande build().

Remarque : Pour plus d'informations, consultez les entrées des différentes propriétés.

- count indique le nombre de triangles de la maille.
- colors indique les valeurs d'index de la liste des couleurs de la ressource de modèle à utiliser pour chacun des sommets de la face.
- normals indique les valeurs d'index de la liste des normales de la ressource de modèle à utiliser pour chacun des sommets de la face.
- shadowPercentage identifie le matériau utilisé lorsque la face est rendue.
- textureCoordinates indique les valeurs d'index de la liste des coordonnées de texture de la ressource de modèle à utiliser pour chacun des sommets de la face.
- vertices indique les valeurs d'index de la liste des sommets de la ressource de modèle à utiliser pour définir la face.

Consultez l'entrée de meshDeform pour plus d'informations sur ces propriétés.

Voir aussi

build(), newMesh, meshDeform (modificateur)

face[ ]

Utilisation

```
member(quelActeur).model(quelModèle).meshdeform.\
  mesh[indexDeMaille].face[indexDeFaces]
```

Description

Propriété 3D de modificateur `meshdeform` ; indique les valeurs d'index de la liste des sommets de la ressource de modèle utilisées pour définir la face.

Cette propriété peut être testée, mais pas définie. Vous pouvez spécifier les sommets d'une face de la ressource de modèle #`mesh` en définissant ses propriétés `vertexList` et `vertices` et en appelant la commande `build`.

Exemple

L'instruction suivante indique que la première face de la première maille du modèle Sol est définie par les trois premiers vecteurs de la liste des sommets de la ressource de modèle utilisée par Sol :

```
put member("Séquence").model("Sol").meshdeform.mesh[1].face[1]
-- [1, 2, 3]
```

Voir aussi

`meshDeform (modificateur)`, `face`, `vertexList (déformation de maille)`, `vertices`

far (brouillard)

Utilisation

```
member(quelActeur).camera(quelleCaméra).fog.far
sprite(quelleImageObjet).camera{(index)}.fog.far
```

Description

Propriété 3D de caméra ; indique la distance à partir de la caméra, en unités de l'univers, à partir de laquelle le brouillard atteint sa densité maximum lorsque la propriété `fog.enabled` a pour valeur `TRUE`.

La valeur par défaut de cette propriété est 1000.

Exemple

L'instruction suivante donne à la propriété de brouillard `far` de la caméra vueDeLaBaie la valeur 5000. Si la propriété `enabled` du brouillard a pour valeur `TRUE`, le brouillard atteindra sa densité maximale à 5000 unités (de l'univers) en face de la caméra.

```
member("monTerrain").camera("vueDeLaBaie").fog.far = 5000
```

Voir aussi

`fog`, `near (brouillard)`

fieldOfView

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.fieldOfView

// Syntaxe JavaScript
réfObjImageObjet.fieldOfView;
```

Description

Propriété d'image-objet QTVR ; donne le champ de vue, en degrés, de l'image-objet spécifiée.

Cette propriété peut être testée et définie.

fieldOfView (3D)

Utilisation

```
member(quelActeur).camera(quelleCaméra).fieldOfView
sprite(quelleImageObjet).camera{(index)}.fieldOfView
```

Description

Propriété 3D de caméra ; indique l'angle formé par deux rayons : un tracé de la caméra vers le haut du plan de projection et l'autre de la caméra vers le bas du plan de projection.

Les images des modèles de l'univers 3D sont plaquées sur le plan de projection, qui est positionné en face de la caméra, tel un écran en face d'un projecteur. Le plan de projection est ce que vous voyez dans l'image-objet 3D. Le haut et le bas du plan de projection sont définis par la propriété `fieldOfView`. Remarquez cependant que l'image-objet n'est pas redimensionnée en fonction des changements de la propriété `fieldOfView`. L'image du plan de projection est en fait redimensionnée pour tenir dans le rectangle de l'image-objet.

La valeur de cette propriété n'a d'importance que lorsque la valeur de la propriété `projection` de la caméra est `#perspective`. Lorsque la propriété `projection` a pour valeur `#orthographic`, vous devrez utiliser la propriété `orthoHeight` de la caméra pour définir le haut et le bas du plan de projection.

Le paramètre par défaut de cette propriété est 30.0.

Exemple

L'instruction suivante donne à la propriété `fieldOfView` de la caméra 1 la valeur 90 :

```
member("Univers 3D").camera[1].fieldOfView = 90
```

Voir aussi

`orthoHeight`

fileFreeSize

Utilisation

```
-- Syntaxe Lingo
_movie.fileFreeSize

// Syntaxe JavaScript
_movie.fileFreeSize;
```

Description

Propriété d'animation ; renvoie le nombre d'octets inutilisés dans l'animation courante à cause de modifications apportées aux bibliothèques de distributions et aux acteurs de cette animation. En lecture seule.

Les commandes `Enregistrer` et `compresser` et `Enregistrer sous` suppriment automatiquement les espaces inutilisés du fichier de l'animation.

Si l'animation ne contient aucun espace inutilisé, `fileFreeSize` renvoie 0.

Exemple

L'instruction suivante affiche le nombre d'octets inutilisés dans le fichier de l'animation courante :

```
-- Syntaxe Lingo
put(_movie.fileFreeSize)

// Syntaxe JavaScript
put(_movie.fileFreeSize);
```

Voir aussi

`Animation`

fileName (distribution)

Utilisation

```
-- Syntaxe Lingo
réfObjDistribution.fileName

// Syntaxe JavaScript
réfObjDistribution.fileName;
```

Description

Propriété de distribution ; renvoie ou définit le nom de fichier d'une bibliothèque de distribution. Lecture seule pour les bibliothèques de distributions internes, lecture/écriture pour les bibliothèques de distributions externes.

Pour les bibliothèques de distributions externes, `fileName` renvoie le chemin d'accès complet et le nom de la distribution.

Pour les bibliothèques de distributions internes, la valeur que renvoie `fileName` dépend de la bibliothèque de distribution interne spécifiée.

- Si la première bibliothèque de distribution interne est spécifiée, `fileName` renvoie le nom de l'animation.

- Si toute autre bibliothèque de distribution interne est spécifiée, `fileName` renvoie une chaîne vide.

Cette propriété accepte les URL comme références. Cependant, pour utiliser une bibliothèque de distribution depuis Internet et minimiser la durée de téléchargement, utilisez la méthode `downloadNetThing()` ou `preloadNetThing()` pour télécharger le fichier de la distribution sur un disque local, puis définissez `fileName` comme le fichier sur le disque.

Si une animation définit le nom de fichier d'une distribution externe, n'utilisez pas l'option permettant de dupliquer les acteurs pour un chargement plus rapide dans la boîte de dialogue Options du projet.

Exemple

L'instruction suivante affiche le chemin d'accès et le nom de fichier de la distribution externe Boutons dans la fenêtre Messages :

```
-- Syntaxe Lingo
trace(castLib("Boutons").fileName)

// Syntaxe JavaScript
trace(castLib("Boutons").fileName);
```

L'instruction suivante affecte le nom de fichier Contenu.cst à la distribution externe Boutons :

```
-- Syntaxe Lingo
castLib("Boutons").fileName = _movie.path & "Contenu.cst"

// Syntaxe JavaScript
castLib("Boutons").fileName = _movie.path + "Contenu.cst";
```

L'animation utilise ensuite le fichier de distribution externe Contenu.cst comme distribution Boutons.

Les instructions suivantes téléchargent une distribution externe depuis une URL dans le dossier de Director et font de ce fichier la distribution externe intitulée Distribution :

```
-- Syntaxe Lingo
downloadNetThing("http://wwwcbDeMille.com/Thousands.cst", \
  _player.applicationPath & "Thousands.cst")
castLib("Distribution").fileName = _player.applicationPath & \
  "Thousands.cst"

// Syntaxe JavaScript
downloadNetThing("http://wwwcbDeMille.com/Thousands.cst",
  _player.applicationPath + "Thousands.cst");
castLib("Distribution").fileName = _player.applicationPath +
  "Thousands.cst";
```

Voir aussi

`Bibliothèque de distribution`, `downloadNetThing`, `preloadNetThing()`

fileName (acteur)

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.fileName

// Syntaxe JavaScript
réfObjActeur.fileName;
```

Description

Propriété d'acteur ; fait référence au nom du fichier associé à un acteur lié. Lecture/écriture.

Cette propriété est pratique pour changer le fichier lié externe affecté à un acteur pendant la lecture d'une animation, d'une façon semblable au changement d'acteurs. Lorsque le fichier lié se trouve dans un dossier différent de celui de l'animation, vous devez inclure son chemin d'accès.

Vous pouvez également faire des médias non liés des médias liés en définissant le nom de fichier des types d'acteurs supportant le média lié.

Cette propriété accepte également les URL comme référence. Cependant, pour utiliser un fichier d'une URL et minimiser la durée de téléchargement, utilisez la méthode `downloadNetThing()` ou `preloadNetThing()` pour télécharger le fichier sur un disque local, puis définissez la propriété `fileName` comme fichier sur le disque local.

Une fois le nom de fichier défini, Director utilise ce fichier à la prochaine utilisation de l'acteur.

Exemple

L'instruction suivante lie l'acteur séquence QuickTime ChaiseAnimée à l'acteur 40 :

```
-- Syntaxe Lingo
member(40).fileName = "ChaiseAnimée"

// Syntaxe JavaScript
member(40).fileName = "ChaiseAnimée";
```

Les instructions suivantes téléchargent un fichier externe depuis une URL dans le dossier de Director et font de ce fichier le média pour l'acteur son Norma :

```
-- Syntaxe Lingo
downloadNetThing("http://wwwcbDeMille.com/CinémaParlant.AIF", \
  _player.applicationPath & "CinémaParlant.AIF")
member("Norma Desmond parle").fileName = _player.applicationPath & \
  "CinémaParlant.AIF"

// Syntaxe JavaScript
downloadNetThing("http://wwwcbDeMille.com/CinémaParlant.AIF",
  _player.applicationPath + "CinémaParlant.AIF");
member("Norma Desmond parle").fileName = _player.applicationPath +
  "CinémaParlant.AIF";
```

Voir aussi

`downloadNetThing`, `Acteur`, `preloadNetThing()`

fileName (fenêtre)

Utilisation

```
-- Syntaxe Lingo
réfObjFenêtre.fileName

// Syntaxe JavaScript
réfObjFenêtre.fileName;
```

Description

Propriété de fenêtre ; fait référence au nom de fichier de l'animation affectée à une fenêtre. Lecture/écriture.

Lorsque le fichier lié se trouve dans un dossier différent de celui de l'animation, vous devez inclure son chemin d'accès.

Pour pouvoir lire l'animation dans une fenêtre, vous devez affecter la propriété `fileName` au nom de fichier de l'animation.

La propriété `fileName` accepte les URL comme références. Cependant, pour utiliser un fichier d'animation depuis une URL et minimiser la durée de téléchargement, utilisez la méthode `downloadNetThing()` ou `preloadNetThing()` pour télécharger le fichier d'animation sur un disque local, puis définissez la propriété `fileName` comme le fichier sur le disque local.

Exemple

L'instruction suivante affecte le fichier appelé Tableau de commande à la fenêtre Boîte à outils :

```
-- Syntaxe Lingo
window("Boîte à outils").fileName = "Tableau de commande"

// Syntaxe JavaScript
window("Boîte à outils").fileName = "Tableau de commande";
```

L'instruction suivante affiche le nom de fichier affecté à la fenêtre intitulée Navigateur :

```
-- Syntaxe Lingo
trace(window("Navigateur").fileName)

// Syntaxe JavaScript
trace(window("Navigateur").fileName);
```

Les instructions suivantes téléchargent un fichier d'animation depuis une URL dans le dossier de Director et affectent ce fichier à la fenêtre Mon gros plan :

```
-- Syntaxe Lingo
downLoadNetThing("http://www.cbDeMille.com/Final.DIR", \
  _player.applicationPath & "Final.DIR")
window("Mon gros plan").fileName = _player.applicationPath & "Final.DIR"

// Syntaxe JavaScript
downLoadNetThing("http://www.cbDeMille.com/Final.DIR",
  _player.applicationPath + "Final.DIR");
window("Mon gros plan").fileName = _player.applicationPath + "Final.DIR";
```

Voir aussi

`downloadNetThing`, `preloadNetThing()`, `Fenêtre`

fileSize

Utilisation

```
-- Syntaxe Lingo
_movie.fileSize

// Syntaxe JavaScript
_movie.fileSize;
```

Description

Propriété d'animation ; renvoie le nombre d'octets du fichier de l'animation courante enregistré sur le disque dur. En lecture seule.

Il s'agit du même nombre que celui qui est affiché lorsque vous utilisez Propriétés du fichier (Windows) ou Lire les informations (dans le Finder du Macintosh).

Exemple

L'instruction suivante affiche le nombre d'octets de l'animation courante :

```
-- Syntaxe Lingo
put(_movie.fileSize)

// Syntaxe JavaScript
put(_movie.fileSize);
```

Voir aussi

Animation

fileVersion

Utilisation

```
-- Syntaxe Lingo
_movie.fileVersion

// Syntaxe JavaScript
_movie.fileVersion;
```

Description

Propriété d'animation ; indique, sous forme de chaîne, la version de Director dans laquelle l'animation a été le plus récemment enregistrée. En lecture seule.

Exemple

L'instruction suivante affiche la version de Director dans laquelle l'animation a été enregistrée pour la dernière fois :

```
-- Syntaxe Lingo
put(_movie.fileVersion)

// Syntaxe JavaScript
put(_movie.fileVersion);
```

Voir aussi

Animation

fillColor

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.fillColor

// Syntaxe JavaScript
réfObjActeur.fillColor;
```

Description

Propriété d'acteur forme vectorielle ; la couleur de remplissage est spécifiée en valeur RVB.

Il est possible d'utiliser `fillColor` lorsque la propriété `fillMode` de la forme est définie sur `#solid` ou `#gradient`, mais pas si elle est définie sur `#none`. Si `fillMode` est `#gradient`, `fillColor` spécifie la couleur de départ du dégradé. La couleur finale est spécifiée par `endColor`.

Cette propriété peut être testée et définie.

Vous pourrez voir un exemple de `fillColor` dans une animation en consultant l'animation Vector Shapes du dossier Learning/Lingo, lui-même dans le dossier de Director.

Exemple

L'instruction suivante donne à la couleur de remplissage de l'acteur Archie une nouvelle valeur RVB :

```
-- Syntaxe Lingo
member("Archie").fillColor = color( 24, 15, 153)

// Syntaxe JavaScript
member("Archie").fillColor = color( 24, 15, 153);
```

Voir aussi

`endColor`, `fillMode`

fillCycles

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.fillCycles

// Syntaxe JavaScript
réfObjActeur.fillCycles;
```

Description

Propriété d'acteur forme vectorielle ; nombre de cycles de remplissage dégradé d'une forme vectorielle, spécifié par un nombre entier compris entre 1 et 7.

Cette propriété n'est valable que si la propriété `fillMode` de la forme a pour valeur `#gradient`.

Cette propriété peut être testée et définie.

Vous pourrez voir un exemple de `fillCycles` dans une animation en consultant l'animation Vector Shapes du dossier Learning/Lingo, lui-même dans le dossier de Director.

Exemple

L'instruction suivante donne à `fillCycles` de l'acteur Archie la valeur 3 :

```
-- Syntaxe Lingo
member("Archie").fillCycles = 3

// Syntaxe JavaScript
member("Archie").fillCycles = 3;
```

Voir aussi

`endColor`, `fillColor`, `fillMode`

fillDirection

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.fillDirection

// Syntaxe JavaScript
réfObjActeur.fillDirection;
```

Description

Propriété d'acteur forme vectorielle ; spécifie le degré de rotation du remplissage de la forme.

Cette propriété n'est valable que si la propriété `fillMode` de la forme a pour valeur `#gradient`.

Cette propriété peut être testée et définie.

Vous pourrez voir un exemple de `fillDirection` dans une animation en consultant l'animation Vector Shapes du dossier Learning/Lingo, lui-même dans le dossier de Director.

Voir aussi

`fillMode`

filled

Utilisation

```
member(quelActeur).filled
the filled of member quelActeur
```

Description

Propriété d'acteur forme ; indique si l'acteur spécifié contient un motif de remplissage (`TRUE`) ou non (`FALSE`).

Exemple

Les instructions suivantes donnent à l'acteur forme Cible une forme pleine et lui affectent le motif numéro 1, qui est une couleur unie :

```
member("Zone cible").filled = TRUE
member("Zone cible").pattern = 1
```

Voir aussi

`fillColor`, `fillMode`

fillMode

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.fillMode

// Syntaxe JavaScript
réfObjActeur.fillMode;
```

Description

Propriété d'acteur forme vectorielle ; indique la méthode de remplissage de la forme, à l'aide des valeurs suivantes :

- `#none` – La forme est transparente.
- `#solid` – La forme utilise une seule couleur de remplissage.
- `#gradient` – La forme utilise un dégradé entre deux couleurs.

Cette propriété peut être testée et définie lorsque la forme est fermée ; les formes ouvertes n'ont pas de remplissage.

Vous pourrez voir un exemple de `fillMode` dans une animation en consultant l'animation Vector Shapes du dossier Learning/Lingo, lui-même situé dans le dossier de Director.

Exemple

L'instruction suivante donne à `fillMode` de l'acteur Archie la valeur de dégradé :

```
-- Syntaxe Lingo
member("Archie").fillMode = #gradient

// Syntaxe JavaScript
member("Archie").fillMode = symbol("gradient");
```

Voir aussi

`endColor`, `fillColor`

fillOffset

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.fillOffset

// Syntaxe JavaScript
réfObjActeur.fillOffset;
```

Description

Propriété d'acteur forme vectorielle ; spécifie le nombre de pixels horizontaux et verticaux (dans l'espace `defaultRect`) utilisés pour décaler le remplissage de la forme.

Cette propriété n'est valable que si la propriété `fillMode` de la forme a pour valeur `#gradient`, mais peut être testée et définie.

Vous pourrez voir un exemple de `fillOffset` dans une animation en consultant l'animation Vector Shapes du dossier Learning/Lingo, lui-même dans le dossier de Director.

Exemple

L'instruction suivante change le décalage de remplissage de l'acteur forme vectorielle Miette à un décalage horizontal de 33 pixels et un décalage vertical de 27 pixels :

```
-- Syntaxe Lingo
member("Miette").fillOffset = point(33, 27)

// Syntaxe JavaScript
member("Miette").fillOffset = point(33, 27);
```

Voir aussi

`defaultRect`, `fillMode`

fillScale

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.fillScale

// Syntaxe JavaScript
réfObjActeur.fillScale;
```

Description

Propriété d'acteur forme vectorielle ; spécifie le degré de mise à l'échelle du remplissage de la forme. Cette propriété porte également le nom Echelle dans la fenêtre Forme vectorielle.

Cette propriété n'est valable que si la propriété `fillMode` de la forme a pour valeur `#gradient`, mais peut être testée et définie.

Vous pourrez voir un exemple de `fillScale` dans une animation en consultant l'animation Vector Shapes du dossier Learning/Lingo, lui-même dans le dossier de Director.

Exemple

L'instruction suivante donne à `fillScale` de l'acteur Archie la valeur 33 :

```
-- Syntaxe Lingo
member("Archie").fillScale = 33.00

// Syntaxe JavaScript
member("Archie").fillScale = 33.00;
```

Voir aussi

`fillMode`

firstIndent

Utilisation

```
-- Syntaxe Lingo
expressionSousChaîne.firstIndent

// Syntaxe JavaScript
expressionSousChaîne.firstIndent;
```

Description

Propriété d'acteur texte ; contient le nombre de pixels de décalage correspondant au premier retrait de *expressionSousChaîne* à partir de la marge gauche de *expressionSousChaîne*.

La valeur est un nombre entier : un nombre inférieur à 0 indique un retrait négatif, 0 indique l'absence de retrait et un nombre supérieur à 0 indique un retrait normal.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante définit le retrait de la première ligne de l'acteur Bureau sur 0 pixel :

```
-- Syntaxe Lingo
member("Bureau").firstIndent = 0

// Syntaxe JavaScript
member("Bureau").firstIndent = 0;
```

Voir aussi

leftIndent, rightIndent

fixedLineSpace

Utilisation

```
-- Syntaxe Lingo
expressionSousChaîne.fixedLineSpace

// Syntaxe JavaScript
expressionSousChaîne.fixedLineSpace;
```

Description

Propriété d'acteur texte ; contrôle la hauteur de chaque ligne dans la partie de *expressionSousChaîne* de l'acteur texte.

La valeur elle-même est un nombre entier, indiquant la hauteur en pixels absolus de chaque ligne.

La valeur par défaut est 0, qui a pour résultat une hauteur naturelle.

Exemple

L'instruction suivante définit la hauteur, en pixels, de chaque ligne de l'acteur Bureau sur 24 :

```
-- Syntaxe Lingo
member("Bureau").fixedLineSpace = 24

// Syntaxe JavaScript
member("Bureau").fixedLineSpace = 24;
```

fixedRate

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.fixedRate

// Syntaxe JavaScript
réfObjActeurOuImageObjet.fixedRate;
```

Description

Propriété d'acteur et d'image-objet ; contrôle la cadence d'image d'une animation Flash ou d'un GIF animé. La propriété `fixedRate` peut avoir des valeurs entières. La valeur par défaut est 15.

Cette propriété est ignorée si la propriété `playbackMode` de l'image-objet a une valeur différente de `#fixed`.

Cette propriété peut être testée et définie.

Exemple

Le gestionnaire suivant règle la cadence d'image d'une image-objet d'animation Flash. Comme paramètres, le gestionnaire accepte une référence d'image-objet, une indication d'accélération ou de ralentissement de l'animation Flash et l'importance du réglage de la cadence.

```
-- Syntaxe Lingo
on ajustementDeLaCadenceFixée(quelleImageObjet, typeDajustement, Decombien)
  case typeDajustement of
    #accélération:
      sprite(quelleImageObjet).fixedRate = sprite(quelleImageObjet).fixedRate
  + Decombien
    #décélération:
      sprite(quelleImageObjet).fixedRate = sprite(quelleImageObjet).fixedRate
  - Decombien
  end case
end

// Syntaxe JavaScript
function ajustementDeLaCadenceFixée(quelleImageObjet, typeDajustement,
  Decombien) {
  switch(typeDajustement) {
    case "accéleration":
      sprite(quelleImageObjet).fixedRate = sprite(quelleImageObjet).fixedRate
  + Decombien;
      break();
    case "décélération":
      sprite(quelleImageObjet).fixedRate = sprite(quelleImageObjet).fixedRate
  - Decombien;
      break();
  }
}
```

Voir aussi

`playBackMode`

fixStageSize

Utilisation

```
-- Syntaxe Lingo
_movie.fixStageSize

// Syntaxe JavaScript
_movie.fixStageSize;
```

Description

Propriété d'animation ; détermine si la taille de la scène reste la même lorsque vous chargez une nouvelle animation (`TRUE`, valeur par défaut) ou non (`FALSE`), quels que soient le paramètre `centerStage` et la taille de la scène enregistrée avec cette animation. Lecture/écriture.

La propriété `fixStageSize` ne peut pas changer la taille de la scène d'une animation en cours de lecture.

Exemple

L'instruction suivante détermine si la propriété `fixStageSize` est activée. Si `fixStageSize` est `FALSE`, elle envoie la tête de lecture vers une image spécifiée.

```
-- Syntaxe Lingo
if (_movie.fixStageSize = FALSE) then
  _movie.go("taille correcte")
end if

// Syntaxe JavaScript
if (_movie.fixStageSize == false) {
  _movie.go("taille correcte");
}
```

L'instruction suivante donne à la propriété `fixStageSize` la valeur opposée au paramètre courant :

```
-- Syntaxe Lingo
_movie.fixStageSize = not(_movie.fixStageSize)

// Syntaxe JavaScript
_movie.fixStageSize = !(_movie.fixStageSize);
```

Voir aussi

`centerStage`, `Animation`

flashRect

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.flashRect

// Syntaxe JavaScript
réfObjActeur.flashRect;
```

Description

Propriété d'acteur ; indique la taille initiale d'un acteur animation Flash ou forme vectorielle. Les valeurs sont indiquées sous la forme de rectangle Director : par exemple, `rect(0,0,32,32)`.

Pour les acteurs Flash liés, la propriété d'acteur `FlashRect` renvoie une valeur valide uniquement lorsque l'en-tête de l'acteur a été complètement chargé en mémoire.

Cette propriété peut être testée, mais pas définie.

Exemple

Ce script d'image-objet redimensionne une image-objet d'animation Flash pour qu'elle ait une taille identique à la taille d'origine de son acteur animation Flash :

```
-- Syntaxe Lingo
property spriteNum

on beginSprite me
  sprite(spriteNum).rect = sprite(spriteNum).member.FlashRect
end

// Syntaxe JavaScript
function beginSprite() {
  sprite(this.spriteNum).rect = sprite(this.spriteNum).member.FlashRect;
}
```

Voir aussi

`defaultRect`, `defaultRectMode`, `state (Flash, SWA)`

flat

Utilisation

```
member(quelActeur).shader(quelMatériau).flat
member(quelActeur).model(quelModèle).shader.flat
member(quelActeur).model(quelModèle).shaderList[index].flat
```

Description

Propriété 3D de matériau `#standard` ; indique si la maille doit être rendue avec un matériau plat (`TRUE`) ou un matériau de Gouraud (`FALSE`).

Le matériau plat utilise une couleur par face de la maille. La couleur utilisée pour la face est celle du premier sommet. Le matériau plat est plus rapide que le matériau de Gouraud.

Le matériau de Gouraud affecte une couleur à chaque sommet d'une face et les interpole sur la face dans un dégradé. Le matériau de Gouraud nécessite un plus grand effort de calcul mais produit une surface plus lisse.

La valeur par défaut de cette propriété est `FALSE`.

Exemple

L'instruction suivante donne à la propriété `flat` du matériau Mur la valeur `TRUE`. La maille d'un modèle qui utilise ce matériau sera rendue avec une couleur par face.

```
member("mondeMystérieux").shader("Mur").flat = TRUE
```

Voir aussi

`mesh (propriété)`, `colors`, `vertices`, `generateNormals()`

flipH

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.flipH

// Syntaxe JavaScript
réfObjImageObjet.flipH;
```

Description

Propriété d'image-objet ; indique si l'image d'une image-objet a été renversée horizontalement sur la scène (`TRUE`) ou non (`FALSE`). En lecture seule.

L'image même est renversée autour de son point d'alignement.

Cela signifie que les rotations ou inclinaisons restent constantes, seules les données de l'image étant renversées.

Exemple

L'instruction suivante affichage la valeur `flipH` de l'image-objet 5 :

```
-- Syntaxe Lingo
put(sprite(5).flipH)

// Syntaxe JavaScript
put(sprite(5).flipH);
```

Voir aussi

`flipV`, `rotation`, `skew`, `Image-objet`

flipV

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.flipV

// Syntaxe JavaScript
réfObjImageObjet.flipV;
```

Description

Propriété d'image-objet ; indique si l'image d'une image-objet a été renversée verticalement sur la scène (`TRUE`) ou non (`FALSE`). En lecture seule.

L'image même est renversée autour de son point d'alignement.

Cela signifie que les rotations ou inclinaisons restent constantes, seules les données de l'image étant renversées.

Exemple

L'instruction suivante affichage la valeur `flipV` de l'image-objet 5 :

```
-- Syntaxe Lingo
put(sprite(5).flipV)

// Syntaxe JavaScript
put(sprite(5).flipV);
```

Voir aussi

`flipH`, `rotation`, `skew`, `Image-objet`

floatPrecision

Utilisation

```
the floatPrecision
```

Description

Propriété d'animation ; arrondit l'affichage des nombres à virgule flottante au nombre de chiffres après la virgule spécifié. La valeur de `floatPrecision` doit être un nombre entier. La valeur maximum est 15 chiffres utiles ; la valeur par défaut étant 4.

La propriété `floatPrecision` détermine uniquement le nombre de chiffres utilisés pour afficher les nombres à virgule flottante et n'affecte pas le nombre de chiffres utilisés pour les calculs.

- Si `floatPrecision` est compris entre 1 et 15, les nombres à virgule flottante sont affichés avec cette quantité de chiffres après la virgule. Les zéros ne sont pas tronqués.
- Si `floatPrecision` est zéro, les nombres à virgule flottante sont arrondis à l'entier le plus proche. Aucun chiffre n'apparaît après la virgule.
- Si `floatPrecision` est un chiffre négatif, les nombres à virgule flottante sont arrondis à la valeur absolue du nombre de chiffres après la virgule. Les zéros sont alors tronqués.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante arrondit la racine carrée de 3.0 à trois chiffres après la virgule :

```
the floatPrecision = 3
x = sqrt(3.0)
put x
-- 1.732
```

L'instruction suivante arrondit la racine carrée de 3.0 à huit chiffres après la virgule :

```
the floatPrecision = 8
put x
-- 1.73205081
```

fog

Utilisation

```
member(quelActeur).camera(quelleCaméra).fog.color
sprite(quelleImageObjet).camera{(index)}.fog.color
member(quelActeur).camera(quelleCaméra).fog.decayMode
sprite(quelleImageObjet).camera{(index)}.fog.decayMode
member(quelActeur).camera(quelleCaméra).fog.enabled
sprite(quelleImageObjet).camera{(index)}.fog.enabled
member(quelActeur).camera(quelleCaméra).fog.far
sprite(quelleImageObjet).camera{(index)}.fog.far
member(quelActeur).camera(quelleCaméra).fog.near
sprite(quelleImageObjet).camera{(index)}.fog.near
```

Description

Propriété 3D de caméra ; le brouillard crée un flou qui augmente avec la distance. L'effet est semblable à celui de la réalité, à l'exception que le brouillard peut être de n'importe quelle couleur.

Voir aussi

color (brouillard), decayMode, enabled (brouillard), far (brouillard), near (brouillard)

folder

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.folder

// Syntaxe JavaScript
réfObjDvd.folder;
```

Description

Propriété DVD. Définit le chemin du dossier à partir duquel un DVD est exécuté. Lecture/écriture.

Le chemin doit être une chaîne.

Exemple

Cette instruction définit le nom du chemin d'accès au dossier DVD :

```
-- Syntaxe Lingo
member(2).folder = "C:\DVDs\"

// Syntaxe JavaScript
member(2).folder = "C:\\DVDs\\";
```

Voir aussi

DVD

font

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.font

// Syntaxe JavaScript
réfObjActeur.font;
```

Description

Propriété d'acteur texte et champ ; définit la police utilisée pour afficher l'acteur spécifié et requiert la présence de caractères dans l'acteur, ne serait-ce qu'une espace. Le paramètre *quelActeur* peut être un nom ou un numéro d'acteur.

La propriété d'acteur `font` peut être testée et définie.

Vous pourrez voir un exemple de `font` dans une animation en consultant l'animation Text du dossier Learning/Lingo, lui-même dans le dossier de Director.

Exemple

L'instruction suivante donne à la variable `anciennePolice` la valeur `font` courante de l'acteur champ Pierre :

```
-- Syntaxe Lingo
anciennePolice = member("Pierre").font

// Syntaxe JavaScript
var anciennePolice = member("Pierre").font;
```

Voir aussi

`text`, `alignment`, `fontSize`, `fontStyle`, `lineHeight`

fontSize

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.fontSize

// Syntaxe JavaScript
réfObjActeur.fontSize;
```

Description

Propriété d'acteur champ ; détermine la taille de la police utilisée pour afficher l'acteur champ spécifié et requiert la présence de caractères dans l'acteur, ne serait-ce qu'une espace. Le paramètre *quelActeur* peut être un nom ou un numéro d'acteur.

Cette propriété peut être testée et définie. Lorsqu'elle est testée, elle renvoie la hauteur de la première ligne du champ. Lorsqu'elle est définie, elle affecte chaque ligne du champ.

Vous pourrez voir un exemple de `fontSize` dans une animation en consultant l'animation Text du dossier Learning/Lingo, lui-même dans le dossier de Director.

Exemple

L'instruction suivante donne à la variable `ancienneTaille` la valeur `fontSize of member` courante de l'acteur champ Pierre :

```
-- Syntaxe Lingo
ancienneTaille = member("Pierre").fontSize

// Syntaxe JavaScript
var ancienneTaille = member("Pierre").fontSize;
```

L'instruction suivante définit la troisième ligne de l'acteur texte monMenu sur 12 points :

```
member("monMenu").fontSize = 12

// Syntaxe JavaScript
member("monMenu").fontSize = 12;
```

Voir aussi

`text`, `alignment`, `font`, `fontStyle`, `lineHeight`

fontStyle

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.fontStyle
réfObjActeur.char[quelCaractère].fontStyle
réfObjActeur.line[quelleLigne].fontStyle
réfObjActeur.word[quelMot].fontStyle

// Syntaxe JavaScript
réfObjActeur.fontStyle
réfObjActeur.getPropRef("car", quelCaractère).fontStyle;
réfObjActeur.getPropRef("line", quelleLigne).fontStyle;
réfObjActeur.getPropRef("word", quelMot).fontStyle;
```

Description

Propriété d'acteur champ ; détermine les styles appliqués à la police utilisée pour l'affichage de l'acteur champ, du caractère, de la ligne, du mot ou de toute autre expression de sous-chaîne et requiert la présence de caractères dans l'acteur, ne serait-ce qu'une espace.

Cette propriété a pour valeur une chaîne de styles délimités par des virgules. Lingo utilise une police combinant les styles de cette chaîne. Les styles disponibles sont normal, gras, italique, souligné, ombré et étendu, le style condensé étant également disponible sur Macintosh.

Utilisez le style normal pour supprimer tous les styles déjà appliqués. Le paramètre *`quelActeur`* peut être un nom ou un numéro d'acteur.

Cette propriété peut être testée et définie.

Vous pourrez voir un exemple de `fontStyle` dans une animation en consultant l'animation Text du dossier Learning/Lingo, lui-même dans le dossier de Director.

Exemple

L'instruction suivante donne à la variable `ancienStyle` la valeur de la propriété `fontStyle` courante de l'acteur champ Pierre :

```
-- Syntaxe Lingo
ancienStyle = member("Pierre").fontStyle

// Syntaxe JavaScript
var ancienStyle = member("Pierre").fontStyle;
```

L'instruction suivante affecte les valeurs gras et italique à la propriété d'acteur `fontStyle` de l'acteur champ Poème :

```
-- Syntaxe Lingo
member("Poème").fontStyle = "gras, italique"

// Syntaxe JavaScript
member("Poème").fontStyle = "gras, italique";
```

L'instruction suivante affecte la valeur italique à la propriété `fontStyle` du troisième nom de l'acteur champ Noms :

```
-- Syntaxe Lingo
member("Noms").word[3].fontStyle = "italique"

// Syntaxe JavaScript
member("Noms").getPropRef("word", 3).fontStyle = "italique";
```

L'instruction suivante affecte la valeur gras et italique à la propriété `fontStyle` des mots 1 à 4 de l'acteur texte mesNotes :

```
-- Syntaxe Lingo
member("mesNotes").word[1..4].fontstyle = "gras, italique"

// Syntaxe JavaScript
for (var i = 1; i <= 4; i++) {
  member("mesNotes").getPropRef("word", i).fontStyle = "gras, italique";
}
```

Voir aussi

`text`, `alignment`, `fontSize`, `font`, `lineHeight`

foreColor

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.foreColor

// Syntaxe JavaScript
réfObjImageObjet.foreColor;
```

Description

Propriété d'image-objet ; renvoie ou définit la couleur de premier plan d'une image-objet. Lecture/écriture.

Il n'est pas recommandé d'appliquer cette propriété à des acteurs bitmap supérieurs à 1 bit, les résultats pouvant être difficiles à prévoir.

Il est recommandé d'utiliser la nouvelle propriété `color` à la place de la propriété `foreColor`.

Exemple

L'instruction suivante donne à la variable `ancienneCouleur` la couleur de premier plan de l'image-objet 5 :

```
-- Syntaxe Lingo
ancienneCouleur = sprite(5).foreColor

// Syntaxe JavaScript
var ancienneCouleur = sprite(5).foreColor;
```

L'instruction suivante fait de 36 la couleur de premier plan d'une image-objet aléatoire entre les images-objets 11 et 13 :

```
-- Syntaxe Lingo
sprite(10 + random(3)).foreColor = 36

// Syntaxe JavaScript
sprite(10 + random(3)).foreColor = 36;
```

Voir aussi

backColor, color(), Image-objet

frame

Utilisation

```
-- Syntaxe Lingo
_movie.frame

// Syntaxe JavaScript
_movie.frame;
```

Description

Propriété d'animation ; renvoie le numéro de l'image courante de l'animation. En lecture seule.

Exemple

L'instruction suivante envoie la tête de lecture à l'image qui précède l'image en cours :

```
-- Syntaxe Lingo
_movie.go(_movie.frame - 1)

// Syntaxe JavaScript
_movie.go(_movie.frame - 1);
```

Voir aussi

go(), Animation

frameCount

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.frameCount

// Syntaxe JavaScript
réfObjActeur.frameCount;
```

Description

Propriété d'acteur Flash ; indique le nombre d'images dans l'acteur animation Flash. La propriété d'acteur `frameCount` peut avoir des valeurs entières.

Cette propriété peut être testée, mais pas définie.

Exemple

Ce script d'image-objet affiche, dans la fenêtre Messages, le numéro de piste et le nombre d'images d'une animation Flash :

```
-- Syntaxe Lingo
property spriteNum

on beginSprite me
  put(""L'animation Flash de la piste"" && spriteNum && contient"" &&
  sprite(spriteNum).member.frameCount && ""images."")
end

// Syntaxe JavaScript
function beginSprite() {
  trace("L'animation Flash de la piste " + (this.spriteNum) + " contient " +
  sprite(this.spriteNum).member.frameCount + " images.");
}
```

frameLabel

Utilisation

```
-- Syntaxe Lingo
_movie.frameLabel

// Syntaxe JavaScript
_movie.frameLabel;
```

Description

Propriété d'animation ; identifie le libellé affecté à l'image courante. Lecture/écriture au cours d'une session d'enregistrement du scénario uniquement.

Lorsque l'image courante n'a pas de libellé, la valeur de la propriété `frameLabel` est 0.

Exemple

L'instruction suivante vérifie le libellé de l'image courante. Dans ce cas, la valeur de `frameLabel` est Démarrer :

```
-- Syntaxe Lingo
put(_movie.frameLabel)

// Syntaxe JavaScript
put(_movie.frameLabel);
```

Voir aussi

`labelList`, `Animation`

framePalette

Utilisation

```
-- Syntaxe Lingo
_movie.framePalette

// Syntaxe JavaScript
_movie.framePalette;
```

Description

Propriété d'animation ; identifie le numéro d'acteur de la palette utilisée dans l'image courante, qui est soit la palette courante, soit la palette définie dans l'image courante. Lecture/écriture au cours d'une session d'enregistrement du scénario uniquement.

Lorsqu'un contrôle exact des couleurs est nécessaire, utilisez Shockwave Player.

Exemple

L'instruction suivante vérifie la palette utilisée dans l'image courante. Dans ce cas, la palette est l'acteur 45.

```
-- Syntaxe Lingo
put(_movie.framePalette)

// Syntaxe JavaScript
put(_movie.framePalette);
```

L'instruction suivante fait de l'acteur palette 45 la palette de l'image courante :

```
-- Syntaxe Lingo
_movie.framePalette = 45

// Syntaxe JavaScript
_movie.framePalette = 45;
```

Voir aussi

`Animation`

frameRate

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.frameRate

// Syntaxe JavaScript
réfObjActeur.frameRate;
```

Description

Propriété d'acteur ; spécifie la cadence de lecture de l'acteur vidéo numérique ou animation Flash indiqué.

Les valeurs possibles de cadence d'image d'un acteur vidéo numérique correspondent aux boutons radio de sélection des options de lecture vidéo numérique.

- Lorsque la propriété d'acteur `frameRate` est comprise entre 1 et 255, la séquence vidéo numérique lit chaque image à cette cadence d'image. La propriété d'acteur `frameRate` ne peut pas être supérieure à 255.
- Lorsque la propriété d'acteur `frameRate` a pour valeur -1 ou 0, la séquence vidéo numérique lit chaque image à sa cadence normale. Ceci permet la synchronisation de la vidéo avec sa piste audio. Lorsque la propriété `frameRate` est définie sur une valeur autre que -1 ou 0, la piste audio de la vidéo numérique ne sera pas lue.
- Lorsque la propriété d'acteur `frameRate` a pour valeur -2, la séquence vidéo numérique lit chaque image aussi vite que possible.

Pour les acteurs animation Flash, la propriété indique la cadence d'image de l'animation créée dans Flash.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante fixe la cadence d'image de l'acteur vidéo numérique QuickTime Chaise pivotante à 30 images par seconde :

```
-- Syntaxe Lingo
member("Chaise pivotante").frameRate = 30

// Syntaxe JavaScript
member("Chaise pivotante").frameRate = 30;
```

L'instruction suivante indique à l'acteur vidéo numérique QuickTime Chaise pivotante de lire chaque image aussi vite que possible :

```
-- Syntaxe Lingo
member("Chaise pivotante").frameRate = -2
// Syntaxe JavaScript
member("Chaise pivotante").frameRate = -2;
```

Le script d'image-objet suivant vérifie si l'acteur d'image-objet a été créé dans Flash avec une cadence inférieure à 15 images par seconde. Si la cadence de l'animation est inférieure à 15 images par seconde, le script définit la propriété `playBackMode` de l'image-objet de façon à pouvoir lui affecter une autre cadence. Le script donne ensuite à la propriété `fixedRate` de l'image-objet la valeur de 15 images par seconde.

```
-- Syntaxe Lingo
property spriteNum

on beginSprite me
  if sprite(spriteNum).member.frameRate < 15 then
    sprite(spriteNum).playBackMode = #fixed
    sprite(spriteNum).fixedRate = 15
  end if
end

// Syntaxe JavaScript
function beginSprite() {
  var fr = sprite(this.spriteNum).member.frameRate;
  if (fr < 15) {
    sprite(this.spriteNum).playBackMode = symbol("fixed");
    sprite(this.spriteNum).fixedRate = 15;
  }
}
```

Voir aussi

fixedRate, playRate (QuickTime, AVI), currentTime (QuickTime, AVI), playBackMode

frameRate (DVD)

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.frameRate

// Syntaxe JavaScript
réfObjDvd.frameRate;
```

Description

Propriété DVD. Renvoie la valeur `frameRate` du DVD. En lecture seule.

La valeur `frameRate` est renvoyée comme l'un des nombres à virgule flottante suivants :

Valeur flottante	Description
`0.0`	La valeur `frameRate` n'a pas pu être déterminée, soit parce qu'elle ne se trouve pas dans le domaine du titre ou parce que le titre n'est pas un titre de vidéo séquentielle.
`25.0`	Le DVD est programmé pour être exécuté à 25 images par seconde.
`30.0`	Le DVD est programmé pour être exécuté à 30 images par seconde.
`29.97`	Le DVD est programmé pour être exécuté à 29,97 images par seconde.

Voir aussi

DVD

frameScript

Utilisation

```
-- Syntaxe Lingo
_movie.frameScript

// Syntaxe JavaScript
_movie.frameScript;
```

Description

Propriété d'animation ; contient le numéro d'acteur unique du script d'image affecté à l'image courante. Lecture/écriture au cours d'une session d'enregistrement du scénario uniquement.

Au cours d'une session de création de scénario, vous pouvez également affecter un script à l'image courante en définissant la propriété `frameScript`.

Si aucun script d'image n'est affecté à l'image courante, cette propriété renvoie 0.

Exemple

L'instruction suivante affiche le numéro du script affecté à l'image courante. Dans ce cas, le numéro de script est 25.

```
-- Syntaxe Lingo
put(_movie.frameScript)

// Syntaxe JavaScript
put(_movie.frameScript);
```

L'instruction suivante fait de l'acteur script Réponse des boutons le script d'image pour l'image courante :

```
-- Syntaxe Lingo
_movie.frameScript = member("Réponse des boutons")

// Syntaxe JavaScript
_movie.frameScript = member("Réponse des boutons");
```

Voir aussi

Animation

frameSound1

Utilisation

```
-- Syntaxe Lingo
_movie.frameSound1

// Syntaxe JavaScript
_movie.frameSound1;
```

Description

Propriété d'animation ; détermine le numéro de l'acteur affecté à la première piste audio de l'image courante. Lecture/écriture.

Cette propriété peut également être définie au cours d'une session d'enregistrement du scénario.

Exemple

Lors de la session d'enregistrement du scénario, l'instruction suivante affecte l'acteur son Jazz à la première piste audio :

```
-- Syntaxe Lingo
_movie.frameSound1 = member("Jazz").number

// Syntaxe JavaScript
_movie.frameSound1 = member("Jazz").number;
```

Voir aussi

frameSound2, Animation

frameSound2

Utilisation

```
-- Syntaxe Lingo
_movie.frameSound2

// Syntaxe JavaScript
_movie.frameSound2;
```

Description

Propriété d'animation ; détermine le numéro de l'acteur affecté à la seconde piste audio de l'image courante. Lecture/écriture.

Cette propriété peut également être définie au cours d'une session d'enregistrement du scénario.

Exemple

Lors de la session d'enregistrement du scénario, l'instruction suivante affecte l'acteur son Jazz à la seconde piste audio :

```
-- Syntaxe Lingo
_movie.frameSound2 = member("Jazz").number

// Syntaxe JavaScript
_movie.frameSound2 = member("Jazz").number;
```

Voir aussi

frameSound1, Animation

frameTempo

Utilisation

```
-- Syntaxe Lingo
_movie.frameTempo

// Syntaxe JavaScript
_movie.frameTempo;
```

Description

Propriété d'animation ; indique la cadence affectée à l'image courante. Lecture/écriture au cours d'une session d'enregistrement du scénario uniquement.

Exemple

L'instruction suivante vérifie la cadence utilisée dans l'image courante. Dans ce cas, la cadence est de 15 images par seconde.

```
-- Syntaxe Lingo
put(_movie.frameTempo)

// Syntaxe JavaScript
put(_movie.frameTempo);
```

Voir aussi

Animation, puppetTempo()

frameTransition

Utilisation

```
-- Syntaxe Lingo
_movie.frameTransition

// Syntaxe JavaScript
_movie.frameTransition;
```

Description

Propriété d'animation ; spécifie le numéro de l'acteur de transition affecté à l'image courante. Lecture/écriture uniquement au cours d'une session d'enregistrement du scénario pour spécifier les transitions.

Exemple

Lorsqu'elle est utilisée au cours d'une session d'enregistrement du scénario, l'instruction suivante fait de l'acteur Brouillard la transition pour l'image que Lingo est en train d'enregistrer :

```
-- Syntaxe Lingo
_movie.frameTransition = member("Brouillard")

// Syntaxe JavaScript
_movie.frameTransition = member("Brouillard");
```

Voir aussi

Animation

front

Utilisation

member(*quelActeur*).modelResource(*quelleRessourceDeModèle*).front

Description

Propriété de ressource de modèle 3D #box ; indique si le côté de la boîte coupé par son axe des z négatif est fermé (TRUE) ou ouvert (FALSE).

La valeur par défaut de cette propriété est TRUE.

Exemple

L'instruction suivante donne à la propriété `front` de la ressource de modèle Caisse la valeur `FALSE`, ce qui signifie que l'avant de la caisse sera ouvert :

```
member("Univers 3D").modelResource("Caisse").front = FALSE
```

Voir aussi

back, bottom (3D), top (3D), left (3D), right (3D)

frontWindow

Utilisation

```
-- Syntaxe Lingo
_player.frontWindow

// Syntaxe JavaScript
_player.frontWindow;
```

Description

Propriété de lecteur ; indique l'animation dans une fenêtre actuellement au premier plan de l'écran. En lecture seule.

Lorsque la scène est au premier plan, `frontWindow` représente la scène. Lorsqu'un éditeur de média ou une palette flottante est au premier plan, `frontWindow` renvoie la valeur `VOID` (Lingo) ou `null` (syntaxe JavaScript).

Exemple

L'instruction suivante détermine si la fenêtre Musique est la fenêtre qui se trouve au premier plan et, le cas échéant, amène la fenêtre Ecoutez ça à l'avant :

```
-- Syntaxe Lingo
if (_player.frontWindow = "Musique") then
  window("Ecoutez ça").moveToFront()
end if

// Syntaxe JavaScript
if (_player.frontWindow = "Musique") {
  window("Ecoutez ça").moveToFront();
}
```

Voir aussi

Lecteur

fullScreen

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.fullScreen

// Syntaxe JavaScript
réfObjDvd.fullScreen;
```

Description

Propriété DVD ; spécifie que le DVD doit être lu en plein écran. Si vous appuyez sur la touche Echap, le mode d'affichage retourne à « non plein écran » et la propriété est définie sur `false`. A l'heure actuelle, pas de prise en charge sur Macintosh. Lecture/écriture.

Voir aussi

DVD

getBoneID

Utilisation

```
référenceDacteur.modelResource.getBoneID("nomDeSegment")
```

Description

Propriété 3D de ressource de modèle ; renvoie le numéro d'index du segment *nomDeSegment* de la ressource de modèle. Cette propriété renvoie 0 en l'absence d'un segment de ce nom.

Exemple

L'instruction suivante renvoie le numéro du segment tibiaG :

```
put member("Parc").modelResource("Gamin").getBoneId("tibiaG")
-- 40
```

Voir aussi

bone

globals

Utilisation

```
the globals
```

Description

Propriété système ; cette propriété contient une liste de propriétés spéciale constituée de toutes les variables globales ayant une valeur autre que `VOID`. Chaque variable globale est une propriété dans la liste, avec une valeur associée.

Vous pouvez utiliser les opérations de liste suivantes sur `globals` :

- `count()` – Renvoie le nombre d'entrées dans la liste.
- `getPropAt(n)` – Renvoie le nom de la *n*ième entrée.
- `getProp(x)` – Renvoie la valeur d'une entrée avec le nom spécifié.
- `getAProp(x)` – Renvoie la valeur d'une entrée avec le nom spécifié.

Remarque : La propriété `globals` contient automatiquement la propriété `#version`, qui est la version de Director en cours d'exécution. Cela signifie qu'il y aura toujours au moins une entrée dans la liste, même si aucune globale variable n'a encore été déclarée.

Cette propriété est différente de `showGlobals` dans le sens que `the globals` peut être utilisée dans des contextes autres que dans la fenêtre Messages. Utilisez `showGlobals` pour afficher `the globals` dans la fenêtre Messages.

Voir aussi

`showGlobals()`, `clearGlobals()`

glossMap

Utilisation

```
member(quelActeur).shader(quelMatériau).glossMap
member(quelActeur).model(quelModèle).shader.glossMap
member(quelActeur).model(quelModèle).shaderList[[index]].\
  glossMap
```

Description

Propriété 3D de matériau `#standard` ; spécifie la texture à utiliser pour le placage brillant.

Les propriétés suivantes sont automatiquement définies avec cette propriété :

- La quatrième couche de texture du matériau reçoit la texture que vous spécifiez.
- La valeur de `textureModeList[4]` est établie sur `#none`.
- La valeur de `blendFunctionList[4]` est établie sur `#multiply`.

Exemple

L'instruction suivante donne la texture Ovale comme valeur `glossMap` au matériau utilisé par le modèle boîteEnVerre.

```
member("planète3D").model("boîteEnVerre").shader.glossMap = \
  member("planète3D").texture("Ovale")
```

Voir aussi

`blendFunctionList`, `textureModeList`, `region`, `specularLightMap`, `diffuseLightMap`

gravity

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).gravity
```

Description

Propriété 3D de ressource de modèle de système de particules ; lorsqu'elle est utilisée avec une ressource de modèle de type `#particle`, cette propriété permet d'obtenir ou de définir la propriété `gravity` de la ressource, sous la forme d'un vecteur.

Cette propriété définit la force de gravité appliquée à toutes les particules de chaque palier de la simulation.

La valeur par défaut de cette propriété est `vector(0,0,0)`.

Exemple

Dans l'exemple suivant, systèmeThermique est une ressource de modèle de type #particle. L'instruction suivante donne à la propriété de gravité de systèmeThermique la valeur vector(0, -.1, 0), ce qui a pour effet de tirer lentement les particules de systèmeThermique le long de l'axe des y.

```
member("Feux").modelResource("systèmeThermique").gravity = \
  vector(0, -.1, 0)
```

Voir aussi

drag, wind

gradientType

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.gradientType

// Syntaxe JavaScript
réfObjActeur.gradientType;
```

Description

Propriété d'acteur forme vectorielle ; spécifie le dégradé utilisé dans le remplissage de l'acteur.

Les valeurs possibles sont #linear ou #radial. La propriété gradientType n'est valable que lorsque fillMode a pour valeur #gradient.

Cette propriété peut être testée et définie.

Exemple

Le gestionnaire suivant bascule entre les dégradés linéaires ou radiaux dans l'acteur Fond.

```
-- Syntaxe Lingo
on mouseUp me
  if member("Fond").gradientType = #radial then
    member("Fond").gradientType = #linear
  else
    member("fond").gradientType = #radial
  end if
end

// Syntaxe JavaScript
function mouseUp() {
  var gt = member("Fond").gradientType;
  if (gt == "radial") {
    member("Fond").gradientType = symbol("linear");
  } else {
    member("Fond").gradientType = symbol("radial");
  }
}
```

Voir aussi

fillMode

group

Utilisation

```
member(quelActeur).group(quelGroupe)
member(quelActeur).group[index]
```

Description

Elément 3D ; nœud de l'univers 3D qui a un nom, une transformation, un parent et des enfants, mais aucune autre propriété.

Chaque acteur 3D est associé à un groupe par défaut nommé World et qui ne peut pas être supprimé. La hiérarchie parent de tous les modèles, lumières, caméras et groupes qui existent dans l'univers 3D se terminent dans `group("world")`.

Exemple

L'instruction suivante indique que le quatrième groupe de l'acteur nouveauMartien est le groupe Direct01.

```
put member("nouveauMartien").group[4]
-- group("Direct01")
```

Voir aussi

`newGroup`, `deleteGroup`, `child (3D)`, `parent`

height

Utilisation

```
-- Syntaxe Lingo
réfObjImage.height
réfObjActeur.height
réfObjImageObjet.height

// Syntaxe JavaScript
réfObjImage.height;
réfObjActeur.height;
réfObjImageObjet.height;
```

Description

Propriété d'image, d'acteur et d'image-objet ; pour les acteurs forme vectorielle, Flash, GIF animé, RealMedia, Windows Media, bitmap et forme, détermine la hauteur, en pixels, de l'acteur affiché sur la scène. Lecture seule pour les acteurs et les objets image, lecture/écriture pour les images-objets.

Exemple

L'instruction suivante affecte la hauteur de l'acteur Titre à la variable `vHauteur` :

```
-- Syntaxe Lingo
vHauteur = member("Titre").height

// Syntaxe JavaScript
var vHauteur = member("Titre").height;
```

L'instruction suivante définit la hauteur de l'image-objet 10 à 26 pixels :

```
-- Syntaxe Lingo
sprite(10).height = 26

// Syntaxe JavaScript
sprite(10).height = 26;
```

Voir aussi

Acteur, Image-objet, width

height (3D)

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).height
member(quelActeur).texture(quelleTexture).height
```

Description

Propriété 3D de ressource de modèle #box, #cylinder et de texture ; indique la hauteur de l'objet.

La hauteur d'une ressource de modèle #box ou #cylinder est mesurée en unités de l'univers et peut être testée et définie. La valeur par défaut de cette propriété est 50.

La hauteur d'une texture est mesurée en pixels et peut être testée mais pas définie. La hauteur de la texture est arrondie à partir de la hauteur de la source de la texture à la puissance de 2 la plus proche.

Exemple

L'instruction suivante définit la hauteur de la ressource de modèle Tour à 225.0 unités de l'univers.

```
member("Univers 3D").modelResource("Tour").height = 225.0
```

L'instruction suivante indique que la hauteur de la texture placageMars est 512 pixels.

```
put member("Séquence").texture("placageMars").height
-- 512
```

Voir aussi

length (3D), width (3D)

heightVertices

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  heightVertices
```

Description

Proprieté 3D de ressource de modèle #box ; indique le nombre de sommets de la maille le long de la hauteur de la boîte. L'augmentation de cette valeur augmente le nombre de faces et donc la précision de la maille.

La hauteur d'une boîte est mesurée sur son axe des y.

Donnez à la propriété `renderStyle` du matériau d'un modèle la valeur `#wire` pour afficher toutes les faces de la maille de la ressource. Donnez à la propriété `renderStyle` la valeur `#point` pour n'afficher que les sommets de la maille.

La valeur de cette propriété doit être supérieure ou égale à 2. La valeur par défaut est 4.

Exemple

L'instruction suivante donne à la propriété `heightVertices` de la ressource de modèle Tour la valeur 10. Neuf polygones seront utilisés pour définir la géométrie de la ressource de modèle le long de son axe des z ; il y aura donc dix sommets.

```
member("Univers 3D").modelResource("Tour").heightVertices = 10
```

Voir aussi

`height (3D)`

highlightPercentage

Utilisation

```
member(quelActeur).model(quelModèle).toon.highlightPercentage
member(quelActeur).model(quelModèle).shader.highlight\
  Percentage
member(quelActeur).shader(quelMatériau).highlightPercentage
```

Description

Propriété 3D de modificateur `toon` et de matériau `#painter` ; indique le pourcentage de couleurs disponibles utilisé dans la région éclairée de la surface du modèle.

La plage de cette propriété s'étend de 0 à 100, la valeur par défaut étant 50.

Le nombre de couleurs utilisées par le modificateur `toon` et le matériau `painter` pour un modèle est déterminé par la propriété `colorSteps` du modificateur `toon` ou du matériau `#painter` du modèle.

Exemple

L'instruction suivante donne à la propriété `highlightPercentage` du modificateur `toon` du modèle Sphère la valeur 50. La moitié des couleurs disponibles pour le modificateur `toon` sera utilisée pour la région éclairée de la surface du modèle.

```
member("formes").model("Sphère").toon.highlightPercentage = 50
```

Voir aussi

`highlightStrength`, `brightness`

highlightStrength

Utilisation

```
member(quelActeur).model(quelModèle).toon.highlightStrength
member(quelActeur).model(quelModèle).shader.highlightStrength
member(quelActeur).shader(quelMatériau).highlightStrength
```

Description

Propriété 3D de modificateur `toon` et de matériau `#painter` ; indique la luminosité de la région éclairée de la surface du modèle.

La valeur par défaut de cette propriété est 1,0.

Exemple

L'instruction suivante donne à la propriété `highlightStrength` du modificateur `toon` du modèle Théière la valeur 0.5. Les régions éclairées du modèle seront modérément lumineuses.

```
member("formes").model("Théière").toon.highlightStrength = 0.5
```

Voir aussi

`highlightPercentage`, `brightness`

hilite

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.hilite

// Syntaxe JavaScript
réfObjActeur.hilite;
```

Description

Propriété d'acteur ; détermine si une case à cocher ou un bouton radio créé avec l'outil bouton est sélectionné (`TRUE`) ou non (`FALSE`, valeur par défaut). Lecture/écriture.

Exemple

L'instruction suivante vérifie si le bouton Son activé est sélectionné et, le cas échéant, augmente le volume de la piste audio 1 au maximum :

```
-- Syntaxe Lingo
if (member("Son Activé").hilite = TRUE) then
  sound(1).volume = 255
end if

// Syntaxe JavaScript
if (member("Son Activé").hilite == true) {
  sound(1).volume = 255;
}
```

L'instruction suivante sélectionne l'acteur bouton Interrupteur en donnant à la propriété d'acteur `hilite` la valeur `TRUE` :

```
-- Syntaxe Lingo
member("Interrupteur").hilite = TRUE

// Syntaxe JavaScript
member("Interrupteur").hilite = true;
```

Voir aussi

`Acteur`

hither

Utilisation

```
member(quelActeur).camera(quelleCaméra).hither
sprite(quelleImageObjet).camera{(index)}.hither
```

Description

Propriété 3D de caméra ; indique la distance, en unités de l'univers et à partir de la caméra, à partir de laquelle les modèles sont tracés. Les objets plus proches de la caméra que le point `hither` ne sont pas dessinés.

La valeur de cette propriété doit être supérieure ou égale à 1.0 et a une valeur par défaut de 5.0.

Exemple

L'instruction suivante donne à la propriété `hither` de la caméra 1 la valeur 1000. Les modèles plus proches que 1000 unités de l'univers de la caméra ne seront pas visibles.

```
member("systèmeSolaire").camera[1].hither = 1000
```

Voir aussi

`yon`

hotSpot

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.hotSpot

// Syntaxe JavaScript
réfObjActeur.hotSpot;
```

Description

Propriété d'acteur curseur ; détermine l'emplacement horizontal et vertical du pixel représentant la zone référencée dans l'acteur curseur couleur animé `quelActeurCurseur`. Director utilise ce point pour suivre la position du curseur à l'écran (par exemple, lorsqu'il renvoie les valeurs pour les fonctions Lingo `mouseH` et `mouseV`) et pour déterminer l'endroit auquel un survol (signalé par le message Lingo `mouseEnter`) a lieu.

L'angle supérieur gauche du curseur est le point(0,0), qui est la valeur par défaut de `hotSpot`. La définition d'un point en dehors des limites du curseur génère une erreur. Par exemple, le réglage de la zone référencée d'un curseur 16x16 pixels sur point(16,16) produit une erreur (le point de départ étant 0,0 et non 1,1).

Cette propriété peut être testée et définie.

Exemple

Le gestionnaire suivant définit la zone référencée d'un curseur de 32x32 pixels (dont le numéro d'acteur est enregistré dans la variable `numéroDeCurseur`) au milieu du curseur :

```
-- Syntaxe Lingo
on startMovie
  member(numéroDeCurseur).hotSpot = point(16,16)
end

// Syntaxe JavaScript
function startMovie() {
  member(numéroDeCurseur).hotSpot = point(16,16);
}
```

hotSpotEnterCallback

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.hotSpotEnterCallback

// Syntaxe JavaScript
réfObjImageObjet.hotSpotEnterCallback;
```

Description

Propriété d'image-objet QuickTime VR ; contient le nom du gestionnaire exécuté lorsque le curseur entre dans une zone référencée QuickTime VR visible sur la scène. L'image-objet QuickTime VR reçoit le message en premier. Ce message a deux arguments : le paramètre `me` et l'identifiant de la zone référencée dans laquelle le curseur est entré.

Pour annuler l'appel, donnez à cette propriété une valeur de 0.

Pour des performances optimales, ne définissez de propriété d'appel que lorsque absolument nécessaire.

Cette propriété peut être testée et définie.

Voir aussi

`hotSpotExitCallback`, `nodeEnterCallback`, `nodeExitCallback`, `triggerCallback`

hotSpotExitCallback

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.hotSpotExitCallback

// Syntaxe JavaScript
réfObjImageObjet.hotSpotExitCallback;
```

Description

Propriété d'image-objet QuickTime VR ; contient le nom du gestionnaire exécuté lorsque le curseur quitte une zone référencée QuickTime VR visible sur la scène. L'image-objet QuickTime VR reçoit le message en premier. Ce message a deux arguments : le paramètre `me` et l'identifiant de la zone référencée dans laquelle le curseur est entré.

Pour annuler l'appel, donnez à cette propriété une valeur de 0.

Pour des performances optimales, ne définissez de propriété d'appel que lorsque absolument nécessaire.

Cette propriété peut être testée et définie.

Voir aussi

hotSpotEnterCallback, nodeEnterCallback, nodeExitCallback, triggerCallback

HTML

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.HTML

// Syntaxe JavaScript
réfObjActeur.HTML;
```

Description

Propriété d'acteur ; accède au texte et aux balises contrôlant la disposition du texte dans un acteur texte au format HTML.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante affiche dans la fenêtre Messages l'information de format HTML intégrée à l'acteur texte Page d'accueil :

```
-- Syntaxe Lingo
put(member("Page d'accueil").HTML)

// Syntaxe JavaScript
trace(member("Page d'accueil").HTML);
```

Voir aussi

importFileInto(), RTF

hyperlink

Utilisation

```
-- Syntaxe Lingo
expressionSousChaîne.hyperlink

// Syntaxe JavaScript
expressionSousChaîne.hyperlink;
```

Description

Propriété d'acteur texte ; renvoie la chaîne de lien hypertexte pour l'expression de sous-chaîne spécifiée dans l'acteur texte.

Cette propriété peut être testée et définie.

Lors de la récupération de cette propriété, le lien contenant le premier caractère de *expressionSousChaîne* est utilisé.

Les liens hypertexte ne peuvent pas se chevaucher. La définition d'un lien hypertexte sur un lien existant (même partiel) remplace le lien initial par le nouveau.

La définition d'un lien hypertexte en chaîne vide supprime ce lien.

Exemple

Le gestionnaire suivant crée un hyperlien dans le premier mot de l'acteur texte LienMacromedia. Ce texte est lié au site web de Macromedia.

```
-- Syntaxe Lingo
on startMovie
  member("LienMacromedia").word[1].hyperlink = "http://www.macromedia.com"
end

// Syntaxe JavaScript
function startMovie() {
  member("LienMacromedia").getPropRef("word", 1).hyperlink =
  "http://www.macromedia.com";
}
```

Voir aussi

hyperlinkRange, hyperlinkState

hyperlinkRange

Utilisation

```
-- Syntaxe Lingo
expressionSousChaîne.hyperlinkRange

// Syntaxe JavaScript
expressionSousChaîne.hyperlinkRange;
```

Description

Propriété d'acteur texte ; renvoie la plage du lien hypertexte contenant le premier caractère de l'expression de sous-chaîne.

Cette propriété peut être testée, mais pas définie.

De même que `hyperLink` et `hyperLinkState`, la plage du lien renvoyée est celle qui contient le premier caractère de *`expressionSousChaîne`*.

Voir aussi

hyperlink, hyperlinkState

hyperlinks

Utilisation

```
-- Syntaxe Lingo
expressionSousChaîne.hyperlinks

// Syntaxe JavaScript
expressionSousChaîne.hyperlinks;
```

Description

Propriété d'acteur texte ; renvoie une liste linéaire contenant toutes les plages de liens hypertexte pour l'expression de sous-chaîne spécifiée de l'acteur texte. Chaque plage est donnée sous la forme d'une liste linéaire avec deux éléments, un pour le caractère de début du lien et un pour le caractère de fin.

Exemple

L'instruction suivante indique tous les liens de l'acteur texte Glossaire dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(member("Glossaire").hyperlinks) -- [[3, 8], [10, 16], [41, 54]]

// Syntaxe JavaScript
trace(member("Glossaire").hyperlinks); // [[3, 8], [10, 16], [41, 54]]
```

hyperlinkState

Utilisation

```
-- Syntaxe Lingo
expressionSousChaîne.hyperlinkState

// Syntaxe JavaScript
expressionSousChaîne.hyperlinkState;
```

Description

Propriété d'acteur texte ; contient l'état courant du lien hypertexte. Les valeurs possibles de l'état sont : `#normal`, `#active` et `#visited`.

Cette propriété peut être testée et définie.

De même que `hyperLink` et `hyperLinkRange`, la plage du lien renvoyée est celle qui contient le premier caractère de *`expressionSousChaîne`*.

Exemple

Le gestionnaire suivant vérifie si l'hyperlien sélectionné est une adresse web. Le cas échéant, l'état du lien hypertexte passe à `#visited` et l'animation est transférée à l'adresse web.

```
-- Syntaxe Lingo
property spriteNum

on hyperlinkClicked me, données, plage
  if data starts "http://" then
    acteurCourant = sprite(spriteNum).member
    acteurCourant.word[4].hyperlinkState = #visited
    gotoNetPage(données)
  end if
end

// Syntaxe JavaScript
function hyperlinkClicked(données, plage) {
  var st = data.slice(0,7);
  var ht = "http://";
  if (st == ht) {
    acteurCourant = sprite(spriteNum).member;
    acteurCourant.getPropRef("word", 4).hyperlinkState = symbol("visited");
    gotoNetPage(données) ;
  }
}
```

Voir aussi

`hyperlink`, `hyperlinkRange`

idleHandlerPeriod

Utilisation

```
-- Syntaxe Lingo
_movie.idleHandlerPeriod

// Syntaxe JavaScript
_movie.idleHandlerPeriod;
```

Description

Propriété d'animation ; détermine le nombre de battements maximum avant l'envoi d'un message `idle` par l'animation. Lecture/écriture.

La valeur par défaut est 1, ce qui indique à l'animation d'envoyer des messages de gestionnaires `idle` pas plus de 60 fois par seconde.

Lorsque la tête de lecture entre dans une image, Director démarre un compteur, redessine les images-objets appropriées sur la scène et émet un événement `enterFrame`. Ensuite, si le temps défini pour la cadence s'est écoulé, Director produit un événement `exitFrame` et passe à l'image suivante spécifiée ; si le temps défini pour cette image ne s'est pas écoulé, Director attend jusqu'à ce que le temps se soit écoulé et produit un message `idle` périodiquement. Le temps écoulé entre les événements `idle` est déterminé par `idleHandlerPeriod`.

Les valeurs possibles pour `idleHandlerPeriod` sont :

- `0` – Autant d'événements `idle` que possible.
- `1` – Jusqu'à 60 par seconde.
- `2` – Jusqu'à 30 par seconde.
- `3` – Jusqu'à 20 par seconde.
- *n* – Jusqu'à 60/*n* par seconde.

Le nombre d'événements `idle` par image dépend également de la cadence d'image de l'animation et d'autres activités, y compris si des scripts sont exécutés. Si la cadence est de 60 images par seconde (ips) et que la valeur de `idleHandlerPeriod` est 1, un seul événement `idle` a lieu par image. Si la cadence est de 20 ips, trois événements `idle` ont lieu par image. Un temps mort survient lorsque Director n'a pas de tâches à exécuter et ne peut pas produire d'événements.

Au contraire, si la propriété `idleHandlerPeriod` a pour valeur 0 et que la cadence est très basse, des milliers d'événements `idle` peuvent être générés.

La valeur par défaut de cette propriété est 1.

Exemple

L'instruction suivante entraîne l'animation à envoyer un message `idle` une fois par seconde, au maximum :

```
-- Syntaxe Lingo
_movie.idleHandlerPeriod = 60

// Syntaxe JavaScript
_movie.idleHandlerPeriod = 60;
```

Voir aussi

`on idle`, `idleLoadMode`, `idleLoadPeriod`, `idleLoadTag`, `idleReadChunkSize`, `Animation`

idleLoadMode

Utilisation

```
-- Syntaxe Lingo
_movie.idleLoadMode

// Syntaxe JavaScript
_movie.idleLoadMode;
```

Description

Propriété d'animation ; détermine le moment auquel les méthodes `preLoad()` et `preLoadMember()` essayent de charger les acteurs pendant les périodes d'inactivité. Lecture/écriture.

Les périodes d'inactivité peuvent avoir une des valeurs suivantes :

- `0` – Aucun chargement en période d'inactivité.
- `1` – Chargement pendant les périodes d'inactivité entre les images.
- `2` – Chargement pendant les événements `idle`.
- `3` – Chargement aussi fréquemment que possible.

La propriété `idleLoadMode` n'effectue aucune fonction et n'est exécutée qu'avec les méthodes `preLoad()` et `preLoadMember()`.

Les acteurs chargés pendant les périodes d'inactivité restent compressés jusqu'à ce que l'animation les utilise. Lors de la lecture de l'animation, des pauses importantes peuvent avoir lieu pendant la décompression des acteurs.

Exemple

L'instruction suivante ordonne à l'animation d'essayer de charger aussi fréquemment que possible les acteurs à précharger, par le biais des commandes `preLoad` et `preLoadMember` :

```
-- Syntaxe Lingo
_movie.idleLoadMode = 3

// Syntaxe JavaScript
_movie.idleLoadMode = 3;
```

Voir aussi

`on idle`, `Animation`, `preLoad() (animation)`, `preLoadMember()`

idleLoadPeriod

Utilisation

```
-- Syntaxe Lingo
_movie.idleLoadPeriod

// Syntaxe JavaScript
_movie.idleLoadPeriod;
```

Description

Propriété d'animation ; détermine le nombre de battements pendant lequel Director attend avant d'essayer de charger les acteurs en attente de chargement. Lecture/écriture.

La valeur par défaut de `idleLoadPeriod` est 0, ce qui indique à Director de s'occuper de la file de chargement aussi fréquemment que possible.

Exemple

L'instruction suivante indique à Director d'essayer de charger toutes les demi-secondes (30 battements) les acteurs en attente de chargement :

```
-- Syntaxe Lingo
_movie.idleLoadPeriod = 30

// Syntaxe JavaScript
_movie.idleLoadPeriod = 30;
```

Voir aussi

`on idle`, `Animation`

idleLoadTag

Utilisation

```
-- Syntaxe Lingo
_movie.idleLoadTag

// Syntaxe JavaScript
_movie.idleLoadTag;
```

Description

Propriété d'animation ; identifie ou affecte un numéro aux acteurs en attente de chargement pendant les périodes d'inactivité de l'ordinateur. Lecture/écriture.

La propriété `idleLoadTag` est pratique pour identifier les acteurs d'un groupe que vous souhaitez précharger ; sa valeur peut correspondre à n'importe quel numéro.

Exemple

L'instruction suivante fait du numéro 10 la balise de chargement en période d'inactivité :

```
-- Syntaxe Lingo
_movie.idleLoadTag = 10

// Syntaxe JavaScript
_movie.idleLoadTag = 10;
```

Voir aussi

`on idle`, `Animation`

idleReadChunkSize

Utilisation

```
-- Syntaxe Lingo
_movie.idleReadChunkSize

// Syntaxe JavaScript
_movie.idleReadChunkSize;
```

Description

Propriété d'animation ; détermine le nombre maximum d'octets que Director peut charger lorsqu'il essaie de charger les acteurs à partir de la file de chargement. Lecture/écriture.

La valeur par défaut de `idleReadChunkSize` est de 32 Ko.

Exemple

L'instruction suivante spécifie que 500 ko est le nombre maximum d'octets que Director peut charger lors d'un essai de chargement des acteurs de la file d'attente :

```
-- Syntaxe Lingo
_movie.idleReadChunkSize = (500 * 1024)

// Syntaxe JavaScript
_movie.idleReadChunkSize = (500 * 1024);
```

Voir aussi

`on idle`, `Animation`

image (image)

Utilisation

```
-- Syntaxe Lingo
réfObjImage.image

// Syntaxe JavaScript
réfObjImage.image;
```

Description

Propriété d'image. Fait référence à l'objet image d'un acteur bitmap ou texte, de la scène ou d'une fenêtre. Lecture/écriture pour l'image d'un acteur, lecture seule pour une image de la scène ou d'une fenêtre.

La définition de la propriété `image` d'un acteur modifie immédiatement le contenu de l'acteur. Cependant, lorsque vous obtenez l'image d'un acteur ou d'une fenêtre, Director crée une référence à l'image de l'acteur ou de la fenêtre en question. Si vous apportez des modifications aux fenêtres, le contenu de l'acteur ou de la fenêtre change immédiatement.

Si vous envisagez d'apporter de nombreuses modifications à la propriété `image` d'un élément, il est plus rapide de copier sa propriété `image` dans un nouvel objet image à l'aide de la méthode `duplicate()`, d'apporter ensuite vos modifications au nouvel objet image, puis d'appliquer l'image initiale de l'élément au nouvel objet image. Pour les acteurs autres que bitmap, il est toujours plus rapide d'utiliser la méthode `duplicate()`.

Exemple

L'instruction suivante place l'image de l'acteur fleurDorigine dans l'acteur nouvelleFleur :

```
-- Syntaxe Lingo
member("nouvelleFleur").image = member("fleurDorigine").image

// Syntaxe JavaScript
member("nouvelleFleur").image = member("fleurDorigine").image;
```

Les instructions suivantes placent une référence à l'image de la scène dans la variable `monImage` et placent ensuite cette image dans l'acteur Fleur :

```
-- Syntaxe Lingo
monImage = _movie.stage.image
member("fleur").image = monImage

// Syntaxe JavaScript
var monImage = _movie.stage.image;
member("fleur").image = monImage;
```

Voir aussi

`copyPixels()`, `draw()`, `duplicate() (image)`, `fill()`, `image()`, `setPixel()`

image (RealMedia)

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.image

// Syntaxe JavaScript
réfObjActeurOuImageObjet.image;
```

Description

Propriété d'acteur ou d'image-objet RealMedia ; renvoie un objet image Lingo contenant l'image courante du flux vidéo RealMedia. Vous pouvez utiliser cette propriété pour placer du contenu RealVideo sur un modèle 3D (voir l'exemple ci-après).

Exemple

Cette instruction copie l'image courante de l'acteur RealMedia Real sur l'acteur bitmap Insta :

```
-- Syntaxe Lingo
member("Insta").image = member("Real").image

// Syntaxe JavaScript
member("Insta").image = member("Real").image;
```

image (fenêtre)

Utilisation

```
-- Syntaxe Lingo
réfObjFenêtre.image

// Syntaxe JavaScript
réfObjFenêtre.image;
```

Description

Propriété de fenêtre ; fait référence à l'objet image d'une fenêtre. En lecture seule.

Lorsque vous obtenez l'image d'une fenêtre, Director crée une référence à l'image de la fenêtre en question. Si vous apportez des modifications à l'image, le contenu de la fenêtre change immédiatement.

Si vous envisagez d'apporter de nombreuses modifications à la propriété `image` d'un élément, il est plus rapide de copier la propriété `image` dans un nouvel objet image à l'aide de la méthode `duplicate()` de l'objet Acteur, d'apporter ensuite vos modifications au nouvel objet image, puis d'appliquer l'image initiale de l'élément au nouvel objet image. Pour les acteurs autres que bitmap, il est toujours plus rapide d'utiliser la méthode `duplicate()`.

Exemple

Les instructions suivantes placent une référence à l'image de la scène dans la variable `monImage` et placent ensuite cette image dans la fenêtre Fleur :

```
-- Syntaxe Lingo
monImage = _movie.stage.image
window("fleur").image = monImage

// Syntaxe JavaScript
var monImage = _movie.stage.image;
window("fleur").image = monImage;
```

Voir aussi

`duplicate() (acteur)`, `Fenêtre`

imageCompression

Utilisation

```
-- Syntaxe Lingo
_movie.imageCompression
réfObjActeur.imageCompression

// Syntaxe JavaScript
_movie.imageCompression;
réfObjActeur.imageCompression;
```

Description

Propriété d'acteur bitmap et d'animation ; indique le type de compression que Director applique aux acteurs bitmap internes (non liés) lors de l'enregistrement de l'animation au format Shockwave. Lecture/écriture.

Les valeurs valides de `imageCompression` sont les suivantes :

Valeur	Signification
`#standard`	Utilise le format de compression interne standard de Director.
`#movieSetting`	Utilisez les paramètres de compression de l'animation, tels qu'ils sont stockés dans la propriété `_movie.imageCompression`. Il s'agit de la valeur par défaut pour les formats d'image non limités à la compression standard.
`#jpeg`	La compression JPEG est utilisée. Pour plus d'informations, consultez `imageQuality`.

Vous devez normalement définir cette propriété dans la boîte de dialogue Paramètres de publication de Director.

Exemple

Cette instruction affiche, dans la fenêtre Messages, la valeur `imageCompression` qui s'applique à l'animation en cours de lecture :

```
-- Syntaxe Lingo
put(_movie.imageCompression)

// Syntaxe JavaScript
put(_movie.imageCompression);
```

Voir aussi

`imageQuality`, `Animation`

imageEnabled

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.imageEnabled

// Syntaxe JavaScript
réfObjActeurOuImageObjet.imageEnabled;
```

Description

Propriété d'acteur et d'image-objet ; contrôle si les graphiques d'une animation Flash ou forme vectorielle sont visibles (`TRUE`, valeur par défaut) ou invisibles (`FALSE`).

Cette propriété peut être testée et définie.

Exemple

Le script `beginSprite` suivant masque les graphiques d'une image-objet animation Flash liée lorsque qu'elle apparaît pour la première fois sur la scène et commence à arriver en mémoire, puis enregistre son numéro d'image-objet dans une variable globale intitulée `gImageObjetFluxContinu` pour l'utiliser ultérieurement dans un script d'image du scénario :

```
-- Syntaxe Lingo
global gImageObjetFluxContinu

on beginSprite me
   gImageObjetFluxContinu = me.spriteNum
   sprite(gImageObjetFluxContinu).imageEnabled = FALSE
end
```

```
// Syntaxe JavaScript
function beginSprite() {
  _global.gStreamingSprite = this.spriteNum;
  sprite(_global.gStreamingSprite).imageEnabled = 0;
}
```

Dans une image suivante de l'animation, ce script d'image vérifie si l'image-objet de l'animation Flash spécifiée par la variable globale `gImageObjetFluxContinu` a été transférée en mémoire. Si ce n'est pas le cas, le script maintient la tête de lecture en boucle dans l'image courante jusqu'à ce que l'animation ait été intégralement transférée en mémoire. Il donne ensuite à la propriété `imageEnabled` la valeur `TRUE` de façon à ce que le graphique apparaisse et laisse la tête de lecture passer à l'image suivante du scénario.

```
-- Syntaxe Lingo
global gImageObjetFluxContinu
on exitFrame me
  if sprite(gImageObjetFluxContinu).member.percentStreamed < 100 then
    _movie.go(_movie.frame)
  else
    sprite(gImageObjetFluxContinu).imageEnabled = TRUE
    _movie.updatestage()
  end if
end

// Syntaxe JavaScript
function exitFrame() {
  var stmSp = sprite(_global.gStreamingSprite).member.percentStreamed;
  if (stmSp < 100) {
    _movie.go(_movie.frame);
  } else {
    sprite(_global.gStreamingSprite).imageEnabled = 1;
    _movie.updatestage();
  }
}
```

imageQuality

Utilisation

```
-- Syntaxe Lingo
_movie.imageQuality
réfObjActeur.imageQuality

// Syntaxe JavaScript
_movie.imageQuality;
réfObjActeur.imageQuality;
```

Description

Cette propriété d'acteur bitmap et d'animation indique le niveau de compression à utiliser lorsque la propriété `imageCompression` d'une animation est définie sur `#jpeg`. Lecture/écriture pendant la programmation uniquement.

La plage de valeurs admises s'étend de 0 à 100. Zéro produit la qualité d'image la moins bonne et le plus haut degré de compression, tandis que 100 produit la meilleure qualité d'image et le degré de compression le plus bas.

Cette propriété ne peut être définie que lors de la programmation et n'a aucun effet tant que l'animation n'est pas enregistrée au format Shockwave Player.

Exemple

Cette instruction affiche, dans la fenêtre Messages, la valeur `imageQuality` qui s'applique à l'animation en cours de lecture :

```
-- Syntaxe Lingo
put(_movie.imageQuality)

// Syntaxe JavaScript
put(_movie.imageQuality);
```

Voir aussi

`imageCompression`, `Animation`

immovable

Utilisation

```
member(quelActeur).model(quelModèle).collision.immovable
```

Description

Propriété 3D de modificateur `#collision` ; indique si un modèle peut être déplacé à la suite de collisions. La valeur `TRUE` rend le modèle immuable ; la valeur `FALSE` permet le déplacement du modèle. Cette propriété est un moyen pratique d'améliorer les performances au cours de l'animation étant donné que Lingo n'a pas à vérifier les collisions pour les modèles immobiles.

Cette propriété a une valeur par défaut de `FALSE`.

Exemple

L'instruction suivante donne à la propriété `immovable` du modificateur `collision` associé au premier modèle de l'acteur Séquence la valeur `TRUE`.

```
member("Séquence").model[1].collision.immovable = TRUE
```

Voir aussi

`collision (modificateur)`

ink

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.ink

// Syntaxe JavaScript
réfObjImageObjet.ink;
```

Description

Propriété d'image-objet ; détermine l'effet d'encre appliqué à une image-objet. Lecture/écriture.

Les valeurs valides de `ink` sont les suivantes :

0 - Copie	32 - Opacité
1 - Transparente	33 - Somme limitée
2 - Inverse	34 - Somme

3 - Spectre	35 - Différence limitée
4 - Copie nég.	36 - Fond transparent
5 - Transp. nég.	37 - Plus claire
6 - Inverse nég.	38 - Différence
7 - Spectre nég.	39 - Plus foncée
8 - Dessin seul	40 - Eclaircir
9 - Masque	41 - Assombrir

Dans le cas de 36 (Fond transparent), vous sélectionnez une image-objet dans le scénario et une couleur de transparence dans la puce de couleur d'arrière-plan de la fenêtre Outils. Vous pouvez également définir la propriété `backColor`.

Si vous définissez cette propriété dans un script alors que la tête de lecture ne bouge pas, utilisez la méthode `updateStage()` de l'objet Animation pour redessiner la scène. Si vous changez plusieurs propriétés d'images-objets – ou plusieurs images-objets – utilisez une seule méthode `updateStage()` à la fin de tous les changements.

Exemple

L'instruction suivante donne à la variable *`encreCourante`* la valeur de l'effet d'encre de l'image-objet 3 :

```
-- Syntaxe Lingo
encreCourante = sprite(3).ink

// Syntaxe JavaScript
var encreCourante = sprite(3).ink;
```

L'instruction suivante donne à l'image-objet (i + 1) un effet d'encre Dessin seul en affectant à la propriété d'effet d'encre la valeur 8, qui spécifie une encre Dessin seul :

```
-- Syntaxe Lingo
sprite(i + 1).ink = 8

// Syntaxe JavaScript
sprite(i + 1).ink = 8;
```

Voir aussi

`backColor`, `Image-objet`, `updateStage()`

inker (modificateur)

Syntaxe

```
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  inker.propriétéDeModificateurInker
référenceDobjetDeRessourceDeModèle.inker.propriétéDeModificateurInker
```

Description

Modificateur 3D ; vous pouvez, après avoir ajouté le modificateur `#inker` à une ressource de modèle (avec `addModifier`), en obtenir et définir les propriétés.

Le modificateur `#inker` ajoute des silhouettes, des plis et des bords de délimitation à un modèle existant ; les propriétés du modificateur `#inker` vous permettent d'en contrôler la définition et l'ampleur.

Lorsque le modificateur #inker est utilisé en conjonction avec le modificateur #toon, le rendu est cumulatif et varie en fonction du premier modificateur appliqué. La liste des modificateurs renvoyée par la propriété modifier indiquera #inker ou #toon (en fonction du premier qui aura été ajouté), mais pas les deux. Le modificateur #inker ne peut pas être utilisé en conjonction avec le modificateur #sds.

Le modificateur #inker a les propriétés suivantes :

- lineColor permet d'obtenir ou de définir la couleur des lignes dessinées par le modificateur #inker.
- silhouettes permet de savoir ou de définir si les lignes sont dessinées avec les bords le long de la bordure d'un modèle, en soulignant la forme.
- creases permet de savoir ou de définir si les lignes sont dessinées avec des plis.
- creaseAngle permet de vérifier ou de définir la sensibilité de détection des angles des plis pour le modificateur.
- boundary permet de savoir ou de définir si les lignes sont dessinées autour de la limite de la surface.
- lineOffset permet de savoir ou de définir si les lignes sont tracées en fonction de la surface et de la caméra.
- useLineOffset permet de savoir ou de définir si lineOffset est activé ou désactivé.

Remarque : Pour plus d'informations, consultez les entrées des différentes propriétés.

Voir aussi

addModifier, modifiers, toon (modificateur), shadowPercentage

inlineImeEnabled

Utilisation

```
-- Syntaxe Lingo
_player.inlineImeEnabled

// Syntaxe JavaScript
_player.inlineImeEnabled;
```

Description

Propriété de lecteur ; détermine si la fonction IME en ligne de Director est activée. Lecture/écriture.

Lorsqu'elle est définie sur TRUE, cette propriété permet à l'utilisateur d'entrer directement des caractères double octet dans les fenêtres Texte, Champ, Script et Messages de Director, sur les systèmes japonais.

La valeur par défaut est définie par le paramètre Activer la fonction IME en ligne dans les préférences générales de Director.

Voir aussi

Lecteur

interval

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.interval

// Syntaxe JavaScript
réfObjActeur.interval;
```

Description

Propriété d'acteur curseur ; spécifie l'intervalle, en millisecondes (ms), entre chaque image de l'acteur curseur couleur animé *quelActeurCurseur*. L'intervalle par défaut est de 100 ms.

L'intervalle du curseur est indépendant de la cadence d'images définie pour l'animation dans la piste des cadences ou avec la commande Lingo `puppetTempo`.

Cette propriété peut être testée et définie.

Exemple

Dans ce script d'image-objet, lorsque le curseur animé en couleur stocké dans l'acteur Papillon pénètre dans l'image-objet, l'intervalle est réglé sur 50 ms pour accélérer l'animation. Lorsque le curseur quitte l'image-objet, l'intervalle est réglé sur 100 ms pour ralentir l'animation.

```
-- Syntaxe Lingo
on mouseEnter
  member("Papillon").interval = 50
end

on mouseLeave
  member("Papillon").interval = 100
end

// Syntaxe JavaScript
function mouseEnter() {
   member("Papillon").interval = 50;
}

function mouseLeave() {
   member("Papillon").interval = 100;
}
```

invertMask

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.invertMask

// Syntaxe JavaScript
réfObjActeur.invertMask;
```

Description

Propriété d'acteur QuickTime ; détermine si Director dessine les séquences QuickTime dans les pixels blancs du masque de la séquence (`TRUE`) ou dans ses pixels noirs (`FALSE`, valeur par défaut).

Cette propriété peut être testée et définie.

Exemple

Le gestionnaire suivant inverse le paramètre courant de la propriété `invertMask` d'une séquence QuickTime intitulée Etoile :

```
-- Syntaxe Lingo
on toggleMask
  member("Etoile").invertMask = not(member("Etoile").invertMask)
end

// Syntaxe JavaScript
function toggleMask() {
  member("Etoile").invertMask = !(member("Etoile").invertMask);
}
```

Voir aussi

`mask`

isVRMovie

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.isVRMovie

// Syntaxe JavaScript
réfObjActeurOuImageObjet.isVRMovie;
```

Description

Propriété d'image-objet et d'acteur QuickTime ; indique si un acteur ou une image-objet est une animation QuickTime VR dont le téléchargement n'a pas encore eu lieu (`TRUE`) ou si l'acteur ou l'image-objet n'est pas une animation QuickTime VR (`FALSE`).

Le test de cette propriété dans n'importe quel élément dont le type est différent de `#quickTimeMedia` produira un message d'erreur.

Cette propriété peut être testée, mais pas définie.

Exemple

Le gestionnaire suivant vérifie si l'acteur d'une image-objet est une animation QuickTime. Le cas échéant, le gestionnaire poursuit sa recherche pour vérifier s'il s'agit d'une animation QuickTime VR. Dans les deux cas, un message d'alerte est généré.

```
-- Syntaxe Lingo
on vérifDeVR(uneImageObjet)
  if sprite(uneImageObjet).member.type = #quickTimeMedia then
    if sprite(uneImageObjet).isVRMovie then
      _player.alert("Ceci est un élément QTVR.")
    else
      _player.alert("Ceci n'est pas un élément QTVR.")
    end if
  else
    _player.alert("Ceci n'est pas un élément QuickTime.")
  end if
end
```

```
// Syntaxe JavaScript
function vérifDeVR(uneImageObjet) {
  var TypeActeur = sprite(imageObjet).member.type.toString();
  if (TypeActeur == "quickTimeMedia") {
    var Type = sprite(imageObjet).isVRMovie;
    if (Type == 1) {
      _player.alert("Ceci est un élément QTVR.");
    } else {
      _player.alert("Ceci n'est pas un élément QTVR.");
    } else {
      _player.alert("Ceci n'est pas un élément QuickTime.");
    }
  }
}
```

itemDelimiter

Utilisation

```
the itemDelimiter
```

Description

Propriété de lecteur ; précise le caractère utilisé pour délimiter les éléments.

Vous pouvez utiliser la fonction `itemDelimiter` pour analyser des noms de fichiers en donnant à `itemDelimiter` la valeur d'une barre oblique inverse (\) sous Windows ou de deux-points (:) sur Macintosh. N'oubliez pas de restituer au caractère `itemDelimiter` la valeur d'une virgule (,) pour retourner à un fonctionnement normal.

Cette fonction peut être testée et définie.

Exemple

Le gestionnaire suivant détermine le dernier élément d'un chemin d'accès Macintosh. Il enregistre d'abord le séparateur courant, puis lui donne la valeur de deux-points (:). Lorsque le séparateur est un deux-points, Lingo peut utiliser `the last item of` pour déterminer le dernier élément de la sous-chaîne constituant le chemin d'accès Macintosh. Avant de quitter le gestionnaire, le séparateur reprend sa valeur d'origine.

```
on trouverDernierElément nomDuChemin
  ancien = the itemDelimiter
  the itemDelimiter = ":"
  f = the last item of nomDuChemin
  the itemDelimiter = ancien
  return f
end
```

Voir aussi

Lecteur

kerning

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.kerning

// Syntaxe JavaScript
réfObjActeur.kerning;
```

Description

Propriété d'acteur texte ; spécifie si le crénage doit être automatiquement appliqué au texte lorsque le contenu de l'acteur texte est modifié.

Lorsque cette propriété a pour valeur `TRUE`, le crénage est automatique. Lorsque sa valeur est `FALSE`, le crénage n'est pas appliqué.

La valeur par défaut de cette propriété est `TRUE`.

Voir aussi

`kerningThreshold`

kerningThreshold

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.kerningThreshold

// Syntaxe JavaScript
réfObjActeur.kerningThreshold;
```

Description

Propriété d'acteur texte ; permet de contrôler la taille à partir de laquelle le crénage est automatiquement appliqué à un acteur texte. Cette propriété n'a d'effet que lorsque la propriété `kerning` de l'acteur a pour valeur `TRUE`.

Sa valeur est un entier servant à indiquer la taille en points à partir de laquelle le crénage prend effet.

Cette propriété a une valeur par défaut de 14 points.

Voir aussi

`kerning`

key

Utilisation

```
-- Syntaxe Lingo
_key.key

// Syntaxe JavaScript
_key.key;
```

Description

Propriété de touche ; renvoie la valeur de la dernière touche sur laquelle l'utilisateur a appuyé. En lecture seule.

La valeur renvoyée correspond à la valeur ANSI affectée à la touche et non à sa valeur numérique.

Vous pouvez utiliser la fonction `key` dans les gestionnaires exécutant certaines actions lorsque l'utilisateur appuie sur des touches de raccourcis spécifiques ou d'autres formes d'interactivité. Dans le cas d'une utilisation dans un gestionnaire d'événement principal, les actions spécifiées sont les premières à être exécutées.

Remarque : La valeur de `key` n'est pas mise à jour si l'utilisateur appuie sur une touche lorsque Lingo ou une syntaxe JavaScript exécute une boucle.

Utilisez l'animation Clavier et Lingo pour tester la correspondance des caractères des différentes touches sur différents claviers.

Exemple

Les instructions suivantes font revenir l'animation au repère du menu principal lorsque l'utilisateur appuie sur la touche q. Puisque la propriété `keyDownScript` est définie sur `vérifierLaTouche`, le gestionnaire `on prepareMovie` fait que le gestionnaire d'événement `on vérifierLaTouche` est exécuté en premier lorsque l'utilisateur appuie sur une touche. Le gestionnaire `on vérifierLaTouche` vérifie si la touche q a été enfoncée et, le cas échéant, revient au repère du menu principal.

```
-- Syntaxe Lingo
on prepareMovie
  the keyDownScript = "vérifierLaTouche"
end

on vérifierLaTouche
  if (_key.key = "q") then _movie.go("Menu principal")
  end if
end

// Syntaxe JavaScript
function prepareMovie() {
  keyDownScript = vérifierLaTouche();
}

function vérifierLaTouche() {
  if (_key.key == "q") {
    _movie.go("Menu principal");
  }
}
```

Le gestionnaire `on keyDown` suivant vérifie si la dernière touche enfoncée est la touche z et, le cas échéant, appelle le gestionnaire `on addNumbers` :

```
-- Syntaxe Lingo
on keyDown
  if (_key.key = "z") then addNumbers
end

// Syntaxe JavaScript
function keyDown() {
  if (_key.key == "z") {
    addNumbers();
  }
}
```

Voir aussi

`commandDown`, `Touche`

keyboardFocusSprite

Utilisation

```
-- Syntaxe Lingo
_movie.keyboardFocusSprite

// Syntaxe JavaScript
_movie.keyboardFocusSprite;
```

Description

Propriété d'animation ; permet de concentrer les saisies de clavier (sans contrôler le point d'insertion du curseur) sur une image-objet texte spécifique affichée à l'écran. Lecture/écriture.

Elle équivaut à utiliser la touche Tab lorsque la propriété `autoTab` de l'acteur est sélectionnée.

L'affectation de la valeur -1 à `keyboardFocusSprite` redonne le contrôle des saisies clavier au scénario, tandis que l'attribution de la valeur 0 désactive toute saisie clavier dans une image-objet modifiable.

Voir aussi

`Animation`

keyCode

Utilisation

```
-- Syntaxe Lingo
_key.keyCode

// Syntaxe JavaScript
_key.keyCode;
```

Description

Propriété de touche ; renvoie le code numérique de la dernière touche sur laquelle l'utilisateur a appuyé. En lecture seule.

La valeur renvoyée correspond à la valeur numérique de la touche et non à sa valeur ANSI.

Vous pouvez utiliser `keyCode` pour détecter si l'utilisateur a appuyé sur une touche fléchée ou une touche de fonction, qui ne peut pas être spécifiée à l'aide de la propriété `key`.

Utilisez l'animation Clavier et Lingo pour tester la correspondance des caractères des différentes touches sur différents claviers.

Exemple

Le gestionnaire suivant utilise la fenêtre Messages pour afficher le code approprié chaque fois que l'utilisateur appuie sur une touche :

```
-- Syntaxe Lingo
on enterFrame
  keyDownScript = put(_key.keyCode)
end

// Syntaxe JavaScript
function enterFrame() {
  keyDownScript = put(_key.keyCode);
}
```

L'instruction suivante vérifie si la flèche vers le haut (dont le code est 126) a été enfoncée et, le cas échéant, passe au repère précédent :

```
-- Syntaxe Lingo
if (_key.keyCode = 126) then
  _movie.goPrevious()
end if

// Syntaxe JavaScript
if (_key.keyCode == 126) {
  _movie.goPrevious();
}
```

Le gestionnaire suivant vérifie si l'une des touches fléchées a été enfoncée et, le cas échéant, répond en conséquence :

```
-- Syntaxe Lingo
on keyDown
  case (_key.keyCode) of
    123: VerslaGauche
    126: EnAvant
    125: EnArrière
    124: VerslaDroite
  end case
end keyDown

// Syntaxe JavaScript
function keyDown() {
  switch (_key.keyCode) {
    cas 123 : VerslaGauche();
      break();
    cas 126 : EnAvant();
      break();
    cas 125 : EnArrière();
      break();
    cas 124 : VerslaDroite();
      break();
  }
}
```

Voir aussi

Touche, key

keyDownScript

Utilisation

```
the keyDownScript
```

Description

Propriété système ; spécifie le code Lingo exécuté lorsque l'utilisateur appuie sur une touche. Le code Lingo est rédigé sous forme d'une chaîne entourée de guillemets droits et peut être une instruction simple ou le script d'appel d'un gestionnaire.

Lorsque l'utilisateur appuie sur une touche et que la propriété `keyDownScript` est définie, Lingo exécute en premier les instructions spécifiées dans la propriété `keyDownScript`. A moins que les instructions ne contiennent la commande `pass` autorisant la transmission du message `keyDown` vers d'autres objets de l'animation, aucun autre gestionnaire `on keyDown` ne sera exécuté.

La propriété `keyDownScript` produit le même résultat que la commande `when keyDown then` utilisée dans les versions précédentes de Lingo.

Lorsque les instructions spécifiées pour la propriété `keyDownScript` ne sont plus appropriées, désactivez-les avec l'instruction `set the keyDownScript to EMPTY`.

Exemple

L'instruction suivante donne à `keyDownScript` la valeur `if the key = RETURN then go to the frame + 1`. Lorsque cette instruction est en vigueur, l'animation passe toujours à l'image suivante lorsque l'utilisateur appuie sur la touche Retour.

```
the keyDownScript = "if the key = RETURN then go to the frame + 1"
```

L'instruction suivante paramètre la propriété `keyDownScript` sur le gestionnaire personnalisé *`monGestionnairePersonnalisé`*. Un gestionnaire personnalisé doit être encadré de guillemets droits lorsque utilisé avec la propriété `keyDownScript`.

```
the keyDownScript = "monGestionnairePersonnalisé"
```

Voir aussi

`on keyDown`, `keyUpScript`, `mouseDownScript`, `mouseUpScript`

keyframePlayer (modificateur)

Syntaxe

```
member(quelActeur).model(quelModèle).\
  keyframePlayer.quellePropriétéKeyframePlayer
```

Description

Modificateur 3D ; gère l'utilisation des mouvements par les modèles. Les mouvements gérés par le modificateur `keyframePlayer` animent le modèle tout entier, alors que les mouvements `bonesPlayer` sont effectués segment par segment.

Les mouvements et les modèles qui les utilisent doivent être créés dans un programme de modélisation 3D, exportés au format *.w3d, puis importés dans une animation. Les mouvements ne peuvent pas être appliqués aux primitives de modèle créées dans Director.

L'ajout du modificateur `keyframePlayer` à un modèle à l'aide de la commande `addModifier` permet d'accéder aux propriétés suivantes du modificateur `keyframePlayer` :

- `playing` indique le mouvement d'un modèle.
- `playList` est une liste linéaire de listes de propriétés contenant les paramètres de lecture des mouvements d'un modèle en file d'attente.
- `currentTime` indique la position, en millisecondes, du mouvement en cours de lecture ou en pause.
- `playRate` est un nombre multiplié par le paramètre *`échelle`* de la commande `play()` ou `queue()` pour déterminer la cadence de lecture du mouvement.
- `playlist.count` renvoie le nombre de mouvement en file d'attente dans la liste de lecture.
- `rootLock` indique si le composant de translation du mouvement est utilisé ou ignoré.
- `currentLoopState` indique si le mouvement est lu une seule fois ou répété de façon continue.
- `blendTime` indique la durée de la transition créée par le modificateur entre les mouvements lorsque la propriété `autoblend` a pour valeur `TRUE`.
- `autoBlend` indique si le modificateur crée une transition linéaire entre le mouvement en cours de lecture et le mouvement précédent.
- `blendFactor` indique le degré de fusion entre les mouvements lorsque la propriété `autoBlend` a pour valeur `FALSE`.
- `lockTranslation` indique si le modèle peut être déplacé à partir des plans spécifiés.
- `positionReset` indique si le modèle retourne à sa position de départ à la fin du mouvement ou de chaque itération d'une boucle.
- `rotationReset` indique l'élément de rotation d'une transition d'un mouvement à un autre ou de la boucle d'un seul mouvement.

Remarque : Pour plus d'informations, consultez les entrées des différentes propriétés.

Le modificateur `keyframePlayer` utilise les commandes suivantes :

- `pause` stoppe le mouvement du modèle en cours d'exécution.
- `play()` entraîne ou reprend l'exécution d'un mouvement.
- `playNext()` entraîne la lecture du mouvement suivant de la liste de lecture.
- `queue()` ajoute un mouvement à la fin de la liste de lecture.

Le modificateur `keyframePlayer` génère les événements suivants, qui sont utilisés par les gestionnaires déclarés dans les commandes `registerForEvent()` et `registerScript()`. L'appel au gestionnaire déclaré contient trois arguments : le type d'événement (`#animationStarted` ou `#animationEnded`), le nom du mouvement, ainsi que sa position. Consultez l'entrée de `registerForEvent()` pour plus d'informations sur les événements de notification.

`#animationStarted` est envoyé au début de la lecture d'un mouvement. Si la fusion est utilisée entre des mouvements, l'événement est envoyé au début de la transition.

`#animationEnded` est envoyé à la fin d'un mouvement. Si la fusion est utilisée entre des mouvements, l'événement est envoyé à la fin de la transition.

Voir aussi

`addModifier`, `modifiers`, `bonesPlayer (modificateur)`, `motion`

keyUpScript

Utilisation

```
the keyUpScript
```

Description

Propriété système ; spécifie le code Lingo exécuté lorsque l'utilisateur relâche une touche. Le code Lingo est rédigé sous forme d'une chaîne entourée de guillemets droits et peut être une instruction simple ou le script d'appel d'un gestionnaire.

Lorsque l'utilisateur relâche une touche et que la propriété `keyUpScript` est définie, Lingo exécute en premier les instructions spécifiées dans la propriété `keyUpScript`. A moins que les instructions ne contiennent la commande `pass` autorisant la transmission du message `keyUp` vers d'autres objets de l'animation, aucun autre gestionnaire `on keyUp` ne sera exécuté.

Lorsque les instructions spécifiées dans la propriété `keyUpScript` ne sont plus appropriées, désactivez-les avec l'instruction `set the keyUpScript to EMPTY`.

Exemple

L'instruction suivante donne à `keyUpScript` la valeur `if the key = RETURN then go to the frame + 1`. Lorsque cette instruction est en vigueur, l'animation passe toujours à l'image suivante lorsque l'utilisateur appuie sur la touche Retour.

```
the keyUpScript = "if the key = RETURN then go to the frame + 1"
```

L'instruction suivante définit la propriété `keyUpScript` sur le gestionnaire personnalisé `monGestionnairePersonnalisé`. Un gestionnaire personnalisé doit être encadré de guillemets droits lorsque utilisé avec la propriété `keyUpScript`.

```
the keyUpScript = "monGestionnairePersonnalisé"
```

Voir aussi

`on keyUp`

labelList

Utilisation

```
the labelList
```

Description

Propriété système ; crée une liste des libellés d'images de l'animation courante sous forme de chaîne délimitée par des retours de chariot (et non de liste) contenant un libellé par ligne. Les libellés sont répertoriés en fonction de leur ordre dans le scénario. Les entrées de la liste étant délimitées par des retours chariot, la dernière ligne de la liste est une ligne vide. Assurez-vous de supprimer cette ligne vide si nécessaire.

Exemple

L'instruction suivante fait de la liste des libellés d'images le contenu de l'acteur champ Images-clés :

Le gestionnaire suivant détermine le libellé de l'image démarrant la scène en cours :

Voir aussi

`frameLabel`, `label()`, `marker()`

lastChannel

Utilisation

```
-- Syntaxe Lingo
_movie.lastChannel

// Syntaxe JavaScript
_movie.lastChannel;
```

Description

Propriété d'animation ; affiche le numéro de la dernière piste de l'animation, tel qu'il a été saisi dans la boîte de dialogue Propriétés de l'animation. En lecture seule.

Vous pourrez voir un exemple de `lastChannel` dans une animation en consultant l'animation QT and Flash du dossier Learning/Lingo, lui-même dans le dossier de Director.

Exemple

L'instruction suivante affiche le numéro de la dernière piste de l'animation dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(_movie.lastChannel)

// Syntaxe JavaScript
put(_movie.lastChannel);
```

Voir aussi

`Animation`

lastClick

Utilisation

```
-- Syntaxe Lingo
_player.lastClick

// Syntaxe JavaScript
_player.lastClick;
```

Description

Propriété de lecteur ; renvoie le nombre de battements (1 battement = 1/60ème de seconde) écoulés depuis le dernier clic de la souris. En lecture seule.

Exemple

L'instruction suivante vérifie si 10 secondes se sont écoulées depuis le dernier clic de la souris et, le cas échéant, fait passer la tête de lecture sur le repère Pas de clic :

```
-- Syntaxe Lingo
if (_player.lastClick > (10 * 60)) then
  _movie.go("Pas de clic")
end if

// Syntaxe JavaScript
if (_player.lastClick > (10 * 60)) {
  _movie.go("Pas de clic");
}
```

Voir aussi

lastEvent, lastKey, lastRoll, Lecteur

lastError

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.lastError

// Syntaxe JavaScript
réfObjActeurOuImageObjet.lastError;
```

Description

Propriété d'acteur ou image-objet RealMedia ; permet d'obtenir le dernier symbole d'erreur renvoyé par RealPlayer, sous la forme d'un symbole Lingo. Les symboles d'erreur renvoyés par RealPlayer sont des chaînes simples, en anglais, qui fournissent le point de départ du processus de dépannage. Cette propriété est dynamique en cours de lecture et peut être testée, mais pas définie.

La valeur #PNR_OK indique que tout fonctionne correctement.

Exemple

Les exemples suivants indiquent que la dernière erreur renvoyée par RealPlayer pour l'image-objet 2 et l'acteur Real était #PNR_OUTOFMEMORY :

```
-- Syntaxe Lingo
put(sprite(2).lastError) -- #PNR_OUTOFMEMORY
put(member("Real").lastError) -- #PNR_OUTOFMEMORY

// Syntaxe JavaScript
trace(sprite(2).lastError); // #PNR_OUTOFMEMORY
put(member("Real").lastError); // #PNR_OUTOFMEMORY
```

lastEvent

Utilisation

```
-- Syntaxe Lingo
_player.lastEvent

// Syntaxe JavaScript
_player.lastEvent;
```

Description

Propriété de lecteur ; renvoie le nombre de battements (1 battement = 1/60ème de seconde) écoulés depuis la dernière fois où l'utilisateur a appuyé sur le bouton de la souris, a survolé un élément avec la souris ou a appuyé sur une touche. En lecture seule.

Exemple

L'instruction suivante vérifie si 10 secondes se sont écoulées depuis la dernière fois où l'utilisateur a appuyé sur le bouton de la souris, a survolé un élément avec la souris ou a appuyé sur une touche. Si c'est le cas, la tête de lecture est placée sur le repère Aide :

```
-- Syntaxe Lingo
if (_player.lastEvent > (10 * 60)) then
  _movie.go("Aide")
end if

// Syntaxe JavaScript
if (_player.lastEvent > (10 * 60)) {
  _movie.go("Aide");
}
```

Voir aussi

lastClick, lastKey, lastRoll, Lecteur

lastFrame

Utilisation

```
-- Syntaxe Lingo
_movie.lastFrame

// Syntaxe JavaScript
_movie.lastFrame;
```

Description

Propriété d'animation ; affiche le numéro de la dernière image de l'animation. En lecture seule.

Exemple

L'instruction suivante affiche le numéro de la dernière image de l'animation dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(_movie.lastFrame)

// Syntaxe JavaScript
put(_movie.lastFrame);
```

Voir aussi

Animation

lastKey

Utilisation

```
-- Syntaxe Lingo
_player.lastKey

// Syntaxe JavaScript
_player.lastKey;
```

Description

Propriété de lecteur ; indique le nombre de battements (1 battement = 1/60ème de seconde) écoulés depuis que l'utilisateur a appuyé sur une touche. En lecture seule.

Exemple

L'instruction suivante vérifie si 10 secondes se sont écoulées depuis que l'utilisateur a appuyé sur une touche et, le cas échéant, fait passer la tête de lecture sur le repère Pas de touche :

```
-- Syntaxe Lingo
if (_player.lastKey > (10 * 60)) then
  _movie.go("Pas de touche")
end if

// Syntaxe JavaScript
if (_player.lastKey > (10 * 60)) {
  _movie.go("Pas de touche");
}
```

Voir aussi

lastClick, lastEvent, lastRoll, Lecteur

lastRoll

Utilisation

```
-- Syntaxe Lingo
_player.lastRoll

// Syntaxe JavaScript
_player.lastRoll;
```

Description

Propriété de lecteur ; indique le nombre de battements (1 battement = 1/60ème de seconde) écoulés depuis que l'utilisateur a déplacé la souris. En lecture seule.

Exemple

L'instruction suivante vérifie si 45 secondes se sont écoulées depuis le dernier déplacement de la souris et, le cas échéant, fait passer la tête de lecture au repère Boucle :

```
-- Syntaxe Lingo
if (_player.lastRoll > (45 * 60)) then
  _movie.go("Boucle")
end if

// Syntaxe JavaScript
if (_player.lastRoll > (45 * 60)) {
  _movie.go("Boucle");
}
```

Voir aussi

lastClick, lastEvent, lastKey, Lecteur

left

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.left

// Syntaxe JavaScript
réfObjImageObjet.left;
```

Description

Propriété d'image-objet ; identifie la coordonnée horizontale gauche du rectangle de délimitation d'une image-objet. Lecture/écriture.

Les coordonnées d'images-objets sont exprimées en pixels, (0,0) correspond au coin supérieur gauche de la scène.

Exemple

L'instruction suivante détermine si le côté gauche de l'image-objet se trouve à gauche du côté gauche de la scène. Le cas échéant, le script exécute le gestionnaire `débordementGauche` :

```
-- Syntaxe Lingo
if (sprite(3).left < 0) then
  débordementGauche()
end if

// Syntaxe JavaScript
if (sprite(3).left < 0) {
  débordementGauche();
}
```

L'instruction suivante mesure la coordonnée horizontale gauche de l'image-objet identifiée par le numéro (i + 1) et donne cette valeur à la variable `vPlusbas` :

```
-- Syntaxe Lingo
vPlusbas = sprite (i + 1).left

// Syntaxe JavaScript
var vPlusbas = sprite (i + 1).left;
```

Voir aussi

`bottom`, `height`, `locH`, `locV`, `right`, `Image-objet`, `top`, `width`

left (3D)

Utilisation

`member(`*`quelActeur`*`).modelResource(`*`quelleRessourceDeModèle`*`).left`

Description

Propriété 3D de ressource de modèle `#box` ; indique si le côté de la boîte coupé par son axe des x négatif est fermé (`TRUE`) ou ouvert (`FALSE`).

La valeur par défaut de cette propriété est `TRUE`.

Exemple

L'instruction suivante donne à la propriété `left` de la ressource de modèle Caisse la valeur `FALSE`, ce qui signifie que le côté gauche de la caisse sera ouvert.

```
member("Univers 3D").modelResource("Caisse").left = FALSE
```

Voir aussi

`back`, `front`, `bottom (3D)`, `top (3D)`, `right (3D)`

leftIndent

Utilisation

`expressionSousChaîne``.leftIndent`

Description

Propriété d'acteur texte ; contient le décalage (exprimé en pixels) entre la marge gauche de la sous-chaîne spécifiée par *`expressionSousChaîne`* et le côté gauche de l'acteur texte.

La valeur est un nombre entier supérieur ou égal à 0.

Cette propriété peut être testée et définie.

Exemple

La ligne suivante met en retrait de 10 pixels la première ligne de l'acteur texte L'histoire :

```
member("L'histoire").line[1].leftIndent = 10
```

Voir aussi

`firstIndent`, `rightIndent`

length (3D)

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).length
référenceDeVecteur.length
```

Description

Propriété 3D de ressource de modèle #box et #plane et de vecteur ; indique la longueur, en unités d'univers, de la boîte ou du plan.

La longueur d'une boîte est mesurée sur son axe des z. La longueur par défaut de la boîte est 50.

La longueur d'un plan est mesurée sur son axe des y. La longueur par défaut du plan est 1.

La longueur d'un vecteur est sa distance, en unités d'univers, à partir de `vector(0, 0, 0)`. Il s'agit de la magnitude du vecteur.

Exemple

L'instruction suivante donne à la variable `maLongueurDeBoîte` la longueur de la ressource de modèle boîteACadeau.

```
maLongueurDeBoîte = member("Univers 3D").modelResource("boîteACadeau").length
```

Voir aussi

`height (3D)`, `width (3D)`, `magnitude`

lengthVertices

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  lengthVertices
```

Description

Proprieté 3D de ressource de modèle #box et #plane ; indique le nombre de sommets de la maille le long de la longueur de la boîte ou du plan. L'augmentation de cette valeur augmente le nombre de faces et donc la précision de la maille.

La longueur d'une boîte est mesurée sur son axe des z. La longueur d'un plan est mesurée sur son axe des y.

Donnez à la propriété `renderStyle` du matériau d'un modèle la valeur #wire pour afficher toutes les faces de la maille de la ressource. Donnez à la propriété `renderStyle` la valeur #point pour n'afficher que les sommets de la maille.

La valeur de cette propriété doit être supérieure ou égale à 2. La valeur par défaut est 4.

Exemple

L'instruction suivante donne à la propriété `lengthVertices` de la ressource de modèle Tour la valeur 10. Neuf triangles seront utilisés pour définir la géométrie de la ressource de modèle le long de son axe des y ; il y aura donc dix sommets.

```
member("Univers 3D").modelResource("tour").lengthVertices = 10
```

Voir aussi

`length (3D)`

level

Utilisation

```
member(quelActeur).model(quelModèle).lod.level
```

Description

Propriété 3D de modificateur `lod` ; indique la quantité de détails supprimés par le modificateur lorsque sa propriété `auto` a pour valeur `FALSE`. La plage de cette propriété s'étend de 0.0 à 100.00.

Lorsque la propriété `auto` du modificateur a pour valeur `TRUE`, la valeur de la propriété `level` est mise à jour de façon dynamique, mais ne peut pas être définie.

Le modificateur `#lod` ne peut être ajouté qu'aux modèles créés dans un programme de modélisation 3D autre que Director. La valeur de la propriété `type` des ressources de modèle utilisées par ces modèles est `#fromFile`. Le modificateur ne peut pas être ajouté aux primitives créées dans Director.

Exemple

L'instruction suivante donne à la propriété `level` du modificateur `lod` du modèle vaisseauSpatial la valeur 50. Si la propriété `auto` du modificateur `lod` a pour valeur `FALSE`, vaisseauSpatial sera tracé avec un niveau de détail moyen. Si la propriété `auto` du modificateur `lod` a pour valeur `TRUE`, ce code n'aura aucun effet.

```
member("Univers 3D").model("vaisseauSpatial").lod.level = 50
```

Voir aussi

`lod (modificateur)`, `auto`, `bias`

lifetime

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).lifetime
```

Description

Propriété 3D de ressource de modèle `#particle` ; pour toutes les particules d'un système de particules, cette propriété indique le nombre de millisecondes entre la création d'une particule et la fin de son existence.

La valeur par défaut de cette propriété est 10 000.

Exemple

Dans l'exemple suivant, systèmeThermique est une ressource de modèle de type `#particle`. L'instruction suivante donne à la propriété `lifetime` de systèmeThermique la valeur 90.0, ce qui signifie que chaque particule de systèmeThermique existera pendant 90 millisecondes.

```
member(8,2).modelResource("systèmeThermique").lifetime = 90.0
```

Voir aussi

`emitter`

light

Utilisation

```
member(quelActeur).light(quelleLumière)
member(quelActeur).light[index]
member(quelActeur).light(quelleLumière).quellePropriétéDeLumière
member(quelActeur).light[index].quellePropriétéDeLumière
```

Description

Elément 3D ; objet à une position de vecteur à partir de laquelle la lumière émane.

Exemple

L'exemple suivant indique les deux façons de faire référence à une lumière. La première ligne utilise une chaîne entre parenthèses et la seconde utilise un nombre entre crochets. La chaîne correspond au nom de la lumière et le nombre à la position de la lumière dans la liste des lumières de l'acteur.

```
cetteLumière = member("Univers 3D").light("spot01")
cetteLumière = member("Univers 3D").light[2]
```

Voir aussi

`newLight`, `deleteLight`

lineColor

Utilisation

```
member(quelActeur).model(quelModèle).inker.lineColor
member(quelActeur).model(quelModèle).toon.lineColor
```

Description

Propriété 3D de modificateur `toon` et `inker` ; indique la couleur des lignes tracées sur le modèle par le modificateur. Pour que cette propriété ait un effet, la propriété `creases`, `silhouettes` ou `boundary`, du modificateur doit avoir pour valeur `TRUE`.

La valeur par défaut de cette propriété est `rgb(0, 0, 0)`.

Exemple

L'instruction suivante donne à la couleur de toutes les lignes tracées par le modificateur `toon` sur le modèle Théière la valeur `rgb(255, 0, 0)`, ce qui correspond à la couleur rouge.

```
member("formes").model("Théière").toon.lineColor = rgb(255, 0, 0)
```

Voir aussi

`creases`, `silhouettes`, `boundary`, `lineOffset`

lineCount

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.lineCount

// Syntaxe JavaScript
réfObjActeur.lineCount;
```

Description

Propriété d'acteur ; indique le nombre de lignes qui apparaissent dans l'acteur champ sur la scène en fonction des retours à la ligne automatiques et non du nombre de retours de chariot que contient la chaîne.

Exemple

L'instruction suivante détermine le nombre de lignes de l'acteur champ Nouvelles du jour lorsqu'il apparaît sur la scène et affecte cette valeur à la variable `nombreDeLignes` :

```
-- Syntaxe Lingo
NombreDeLignes = member("Nouvelles du jour").lineCount

// Syntaxe JavaScript
var NombreDeLignes = member("Nouvelles du jour").lineCount;
```

lineDirection

Utilisation

```
member(quelActeur).lineDirection
```

Description

Propriété d'acteur forme ; utilise 0 ou 1 pour indiquer l'inclinaison de la ligne dessinée.

Si la ligne s'élève de gauche à droite, la valeur de cette propriété est 1. Si la ligne s'abaisse de gauche à droite, sa valeur est 0.

Cette propriété peut être testée et définie.

Exemple

Le gestionnaire suivant inverse l'inclinaison de la ligne de l'acteur laLigne, produisant un effet de balancier :

```
on Balancier
  member("laLigne").lineDirection = \
  not member("laLigne").lineDirection
end
```

lineHeight

Utilisation

```
member(quelActeur).lineHeight
the lineHeight of member quelActeur
```

Description

Propriété d'acteur ; détermine l'interligne utilisé pour afficher l'acteur champ spécifié. Le paramètre *quelActeur* peut être un nom ou un numéro d'acteur.

La définition de la propriété d'acteur `lineHeight` annule temporairement le paramètre du système jusqu'à la fermeture de l'animation. Pour utiliser l'interligne souhaité dans l'animation entière, définissez la propriété `lineHeight` dans un gestionnaire `on prepareMovie`.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante donne à la variable `ancienneHauteur` la valeur `lineHeight` actuelle de l'acteur Pierre :

```
ancienneHauteur = member("Pierre").lineHeight
```

Voir aussi

`text`, `alignment`, `font`, `fontSize`, `fontStyle`

lineOffset

Utilisation

```
member(quelActeur).model(quelModèle).toon.lineOffset
member(quelActeur).model(quelModèle).inker.lineOffset
```

Description

Propriété 3D de modificateur `toon` et `inker` ; indique la distance apparente à laquelle les lignes sont tracées par le modificateur à partir de la surface du modèle. Pour que cette propriété ait un effet, la propriété `useLineOffset` du modificateur doit avoir pour valeur `TRUE` et une ou plusieurs de ses propriétés `creases`, `silhouettes` ou `boundary` doit également avoir pour valeur `TRUE`.

La plage de cette propriété s'étend de -100.00 à +100.00. Son paramètre par défaut est -2.0.

Exemple

L'instruction suivante donne à la propriété `lineOffset` du modificateur `toon` du modèle Théière la valeur 10. Les lignes tracées par le modificateur `toon` à la surface du modèle seront plus apparentes qu'avec la valeur par défaut de -2.

```
member("formes").model("Théière").toon.lineOffset = 10
```

Voir aussi

`creases`, `silhouettes`, `boundary`, `useLineOffset`, `lineColor`

lineSize

Utilisation

```
member(quelActeur).lineSize
the lineSize of member quelActeur
sprite quelleImageObjet.lineSize
the lineSize of sprite quelleImageObjet
```

Description

Propriété d'acteur forme ; détermine l'épaisseur, en pixels, de la bordure de l'acteur forme spécifié sur la scène. Pour les formes non rectangulaires, la bordure correspond au bord de la forme même et non à son rectangle de délimitation.

La propriété `lineSize` de l'image-objet annule la valeur `lineSize` de l'acteur. Si Lingo modifie la valeur de la propriété `lineSize` de l'acteur alors que son image-objet se trouve sur la scène, la valeur de la propriété `lineSize` de cette dernière reste inchangée tant que l'image-objet est présente.

Pour que la valeur définie par Lingo dure au-delà de l'image-objet courante, un script doit être affecté à l'image-objet.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante donne à l'épaisseur de l'acteur forme Champ de réponse une valeur de 5 pixels :

```
member("Champ de réponse").lineSize = 5
```

L'instruction suivante affiche l'épaisseur de la bordure de l'image-objet 4 :

```
épaisseur = sprite(4).lineSize
```

L'instruction suivante donne à l'épaisseur de la bordure de l'image-objet 4 une valeur de 3 pixels :

```
sprite(4).lineSize = 3
```

linked

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.linked

// Syntaxe JavaScript
réfObjActeur.linked;
```

Description

Propriété d'acteur ; contrôle si un script, une animation Flash ou un fichier GIF animé est stocké dans un fichier externe (`TRUE`, valeur par défaut) ou dans la bibliothèque de distribution Director (`FALSE`). Lecture/écriture pour les acteurs script, Flash et GIF animé, lecture seule pour les autres types d'acteurs.

Lorsque les données sont stockées en externe dans un fichier lié, la propriété `pathName` de l'acteur doit indiquer l'emplacement du fichier de l'animation.

Exemple

L'instruction suivante convertit l'acteur Flash Amis provenant d'un acteur lié, en un acteur stocké dans un fichier interne :

```
-- Syntaxe Lingo
member("amis").linked = 0

// Syntaxe JavaScript
member("amis").linked = 0;
```

Voir aussi

Acteur

loaded

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.loaded

// Syntaxe JavaScript
réfObjActeur.loaded;
```

Description

Propriété d'acteur ; indique si un acteur spécifié est chargé en mémoire (TRUE) ou non (FALSE). En lecture seule.

Les types d'acteurs existants se comportent de manière légèrement différente lors de leur chargement :

- Les acteurs forme et script sont toujours chargés en mémoire.
- Les acteurs animation ne sont jamais purgés de la mémoire.
- Les acteurs vidéo numérique peuvent être préchargés en mémoire et purgés de la mémoire indépendamment de leur utilisation. La lecture d'un acteur vidéo numérique chargé en mémoire est plus rapide que lorsqu'il est chargé à partir du disque dur.

Exemple

L'instruction suivante vérifie si l'acteur Démonstration est chargé en mémoire et passe à une autre image si ce n'est pas le cas :

```
-- Syntaxe Lingo
if member("Démonstration").loaded = FALSE then
  _movie.go(1, "Attente")
end if

// Syntaxe JavaScript
if (member("Démonstration").loaded == false) {
  _movie.go(1, "Attente")
}
```

Voir aussi

Acteur

loc (fond et recouvrement)

Utilisation

```
sprite(quelleImageObjet).camera{(index)}.backdrop[index].loc
member(quelActeur).camera(quelleCaméra).backdrop[index].loc
sprite(quelleImageObjet).camera{(index)}.overlay[index].loc
member(quelActeur).camera(quelleCaméra).overlay[index].loc
```

Description

Propriété 3D de fond et de recouvrement ; indique l'emplacement 2D du fond ou recouvrement, mesuré à partir du coin supérieur gauche de l'image-objet.

Cette propriété est à l'origine définie comme paramètre de la commande `addBackdrop`, `addOverlay`, `insertBackdrop` ou `insertOverlay`, qui crée le fond ou le recouvrement.

Exemple

L'instruction suivante positionne le premier fond de la caméra de l'image-objet 2.

```
sprite(2).camera.backdrop[1].loc = point(120, 120)
```

Voir aussi

`bevelDepth`, `overlay`, `regPoint (3D)`

locH

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.locH

// Syntaxe JavaScript
réfObjImageObjet.locH;
```

Description

Propriété d'image-objet ; indique la position horizontale du point d'alignement d'une image-objet. Lecture/écriture.

Les coordonnées des images-objets sont calculées par rapport au coin supérieur gauche de la scène.

Pour que la valeur dure au-delà de l'image-objet courante, un script doit être affecté à l'image-objet.

Exemple

L'instruction suivante place l'image-objet 15 à l'emplacement horizontal auquel l'utilisateur a cliqué :

```
-- Syntaxe Lingo
sprite(15).locH = _mouse.mouseH

// Syntaxe JavaScript
sprite(15).locH = _mouse.mouseH;
```

Voir aussi

`bottom`, `height`, `left`, `locV`, `point()`, `right`, `Image-objet`, `top`, `updateStage()`

lockTranslation

Utilisation

```
member(quelActeur).model(quelModèle).bonesPlayer.\
  lockTranslation
member(quelActeur).model(quelModèle).keyframePlayer.\
  lockTranslation
```

Description

Propriété 3D de modificateur `#bonesPlayer` et `#keyframePlayer` ; empêche le déplacement du ou des plans spécifiés, sauf dans le cas d'une translation absolue des données de mouvement. Toute translation supplémentaire ajoutée manuellement ou à la suite d'une accumulation d'erreurs sera supprimée. Les valeurs possibles `#none`, `#x`, `#y`, `#z`, `#xy`, `#yz`, `#xz` et `#all` indiquent, parmi les trois composants de translation, celui ou ceux qui sont contrôlés pour chaque image. Lorsqu'un verrou est activé sur un axe, le déplacement courant le long de cet axe est enregistré et utilisé ensuite comme déplacement fixé pour l'animation. Ce déplacement peut être réinitialisé en désactivant ce verrou d'axe, en déplaçant l'objet, puis en réactivant le verrou.

En d'autres termes, il définit l'axe de translation à ignorer à la lecture d'un mouvement. Pour conserver un modèle verrouillé sur un plan horizontal, avec le sommet le long de l'axe des *z*, donnez à `lockTranslation` la valeur `#z`. La valeur par défaut de cette propriété est `#none`.

Exemple

L'instruction suivante donne à la propriété `lockTranslation` du modèle Marcheur la valeur `#z`.

```
member("Parc").model("Marcheur").bonesPlayer.\
  lockTranslation = #z
```

Voir aussi

`immovable`

locV

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.locV

// Syntaxe JavaScript
réfObjImageObjet.locV;
```

Description

Propriété d'image-objet ; indique la position verticale du point d'alignement d'une image-objet. Lecture/écriture.

Les coordonnées des images-objets sont calculées par rapport au coin supérieur gauche de la scène.

Pour que la valeur dure au-delà de l'image-objet courante, un script doit être affecté à l'image-objet.

Exemple

L'instruction suivante place l'image-objet 15 à la position verticale où l'utilisateur a cliqué :

```
-- Syntaxe Lingo
sprite(15).locV = _mouse.mouseV

// Syntaxe JavaScript
sprite(15).locV = _mouse.mouseV;
```

Voir aussi

bottom, height, left, locH, point(), right, Image-objet, top, updateStage()

locZ

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.locZ

// Syntaxe JavaScript
réfObjImageObjet.locZ;
```

Description

Propriété d'image-objet ; spécifie l'ordre z dynamique d'une image-objet, permettant de contrôler les différentes couches d'images-objets sans avoir à manipuler les pistes ou les propriétés de ces images-objets. Lecture/écriture.

Cette propriété peut avoir pour valeur un nombre entier allant de - 2 milliards à + 2 milliards. Les nombres élevés font apparaître l'image-objet devant les images-objets dont la valeur est inférieure. Lorsque deux images-objets possèdent la même valeur `locZ`, le numéro de piste détermine l'ordre d'affichage de ces deux images-objets. Cela signifie que les images-objets des pistes dont les numéros sont les plus bas apparaissent derrière les images-objets possédant un numéro de piste plus élevé et ce, même lorsque leurs valeurs `locZ` sont égales.

Par défaut, la valeur `locZ` des images-objets est égale à leur numéro de piste.

Les opérations impliquant des couches, telles que la détection d'un clic ou les événements souris, étant régies par la valeur `locZ` des images-objets, la modification de la valeur `locZ` d'une image-objet peut rendre cette dernière partiellement ou complètement masquée par d'autres images-objets, ce qui empêcherait l'utilisateur de cliquer dessus.

D'autres fonctions de Director ne suivent par la valeur `locZ` des images-objets. Les événements démarrent toujours sur la piste 1 et passent par les pistes suivantes, quel que soit l'ordre Z des images-objets.

Exemple

Le gestionnaire suivant utilise une variable globale appelée gImageObjetPlusElevée qui a été initialisée dans le gestionnaire `startMovie` en fonction du nombre d'images-objets utilisées. Lorsque vous cliquez sur l'image-objet, sa valeur `locZ` est définie sur gImageObjetPlusElevée + 1, qui déplace l'image-objet au premier plan de la scène. La valeur gImageObjetPlusElevée est ensuite incrémentée de 1 pour préparer au prochain appel de `mouseUp`.

```
-- Syntaxe Lingo
on mouseUp me
  global gImageObjetPlusElevée
  sprite(me.spriteNum).locZ = gImageObjetPlusElevée + 1
  gImageObjetPlusElevée = gImageObjetPlusElevée + 1
end

// Syntaxe JavaScript
function mouseUp() {
  _global.gImageObjetPlusElevée;
  sprite(this.spriteNum).locZ = _global.gImageObjetPlusElevée + 1;
  _global.gImageObjetPlusElevée = _global.gImageObjetPlusElevée + 1;
}
```

Voir aussi

`locH`, `locV`, `Image-objet`

lod (modificateur)

Utilisation

`member(`*`quelActeur`*`).model(`*`quelModèle`*`).lod.`*`propriétéDeModificateurLod`*

Description

Modificateur 3D ; supprime de façon dynamique les détails des modèles, au fur et à mesure que ces derniers s'éloignent de la caméra.

Ce modificateur ne peut être ajouté qu'aux modèles créés dans un programme de modélisation 3D autre que Director. La valeur de la propriété `type` des ressources de modèle utilisées par ces modèles est `#fromFile`. De tels modèles utilisent tous la réduction des détails, que le modificateur `lod` y soit associé ou non. L'association du modificateur vous permet de contrôler les propriétés de réduction des détails. Le modificateur ne peut pas être ajouté aux primitives créées dans Director.

Les données du modificateur `lod` sont générées par les programmes de modélisation 3D pour tous les modèles. La définition de la propriété de `userData` à `"sw3d_no_lod = true"` vous permet de spécifier que les données du modificateur `lod` et la mémoire soit libérées lorsque la lecture en flux continu est terminée.

Faites attention lorsque vous utilisez les modificateurs `sds` et `lod` de concert étant donné qu'ils ont des fonctions opposées (le modificateur `sds` ajoute des détails géométriques alors que le modificateur `lod` les supprime). Il est recommandé, avant d'ajouter le modificateur `sds`, de désactiver la propriété `lod.auto` et de choisir la résolution maximum pour la propriété `lod.level`, comme suit :

```
member("monActeur").model("monModèle").lod.auto = 0
member("monActeur").model("monModèle").lod.level = 100
member("monActeur").model("monModèle").addmodifier(#sds)
```

Le modificateur `lod` a les propriétés suivantes :

- `auto` permet au modificateur de définir le niveau de réduction des détails au fur et à mesure que la distance entre le modèle et la caméra change. La valeur de la propriété `level` du modificateur est mise à jour, mais la définition de la propriété `level` n'a aucun effet lorsque la propriété `auto` a pour valeur `TRUE`.
- `bias` indique dans quelle mesure le modificateur supprime les détails du modèle lorsque sa propriété `auto` a pour valeur `TRUE`. La plage de cette propriété s'étend de 0.0 (qui supprime tous les polygones) à 100.0 (qui ne supprime aucun polygone). Le paramètre par défaut de cette propriété est 100,0.
- `level` indique le niveau de réduction des détails lorsque la propriété `auto` du modificateur a pour valeur `FALSE`. La plage de cette propriété s'étend de 0.0 à 100.00.

Remarque : Pour plus d'informations, consultez les entrées des différentes propriétés.

Voir aussi

`sds (modificateur)`, `auto`, `bias`, `level`, `addModifier`

loop (3D)

Utilisation

```
member(quelActeur).loop
```

Description

Propriété 3D d'acteur ; indique si la lecture des mouvements appliqués au premier modèle de l'acteur est répétée (`TRUE`) ou si elle s'arrête après avoir eu lieu une fois (`FALSE`).

Le paramètre par défaut de cette propriété est `TRUE`.

Exemple

L'instruction suivante donne à la propriété `loop` de l'acteur Marcheurs la valeur `TRUE`. Les mouvements exécutés par le premier modèle de Marcheurs seront lus de façon répétée.

```
member("Marcheurs").loop = TRUE
```

Voir aussi

`motion`, `play() (3D)`, `queue() (3D)`, `animationEnabled`

loop (émetteur)

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  emitter.loop
```

Description

Propriété 3D ; utilisée avec une ressource de modèle de type `#particle`, cette propriété permet d'obtenir et de définir ce qu'il advient des particules à la fin de leur vie. Une valeur de boucle de `TRUE` entraîne la « résurrection » des particules, à l'emplacement défini par la propriété `region` de l'émetteur. Une valeur de `FALSE` entraîne la mort des particules à la fin de la durée prévue. Le paramètre par défaut de cette propriété est `TRUE`.

Exemple

Dans l'exemple suivant, systèmeThermique est une ressource de modèle de type `#particle`. L'instruction suivante donne à la propriété `emitter.loop` de systèmeThermique la valeur 1, ce qui entraîne l'émission continue des particules de systèmeThermique.

```
member("Feux").modelResource("systèmeThermique").emitter.loop = 1
```

Voir aussi

`emitter`

loop (acteur)

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.loop

// Syntaxe JavaScript
réfObjActeur.loop;
```

Description

Propriété d'acteur ; détermine si l'acteur vidéo numérique, audio ou animation Flash spécifié doit être exécuté en boucle (`TRUE`) ou non (`FALSE`).

Exemple

L'instruction suivante fait boucler l'acteur séquence QuickTime Démo :

```
-- Syntaxe Lingo
member("Démo").loop = 1

// Syntaxe JavaScript
member("Démo").loop = 1;
```

loop (Flash)

Utilisation

```
sprite(quelleImageObjetFlash).loop
the loop of sprite quelleImageObjetFlash
member(quelActeurFlash).loop
the loop of member quelActeurFlash
```

Description

Propriété d'image-objet et d'acteur Flash ; contrôle si une animation Flash sera lue en boucle continue (`TRUE`) ou ne sera lue qu'une fois avant de s'arrêter (`FALSE`).

Cette propriété peut être testée et définie.

Exemple

Le script d'image suivant vérifie la progression du téléchargement d'un acteur animation Flash appelé NetFlash dans la piste 5 en utilisant la propriété `percentStreamed`. Pendant le téléchargement de NetFlash, l'animation exécute une lecture en boucle de l'image courante. Une fois le téléchargement de NetFlash terminé, l'animation passe à l'image suivante et la propriété `loop` de l'animation Flash de la piste 6 reçoit la valeur `FALSE` pour permettre la lecture de l'animation jusqu'à la fin avant son arrêt (l'image-objet est lue en boucle pendant le téléchargement de NetFlash).

```
on exitFrame
  if member("NetFlash").percentStreamed = 100 then
    sprite(6).loop = FALSE
    go the frame + 1
  end if
  go the frame
end
```

loop (Windows Media)

Utilisation

```
-- Syntaxe Lingo
réfObjWindowsMedia.loop

// Syntaxe JavaScript
réfObjWindowsMedia.loop;
```

Description

Propriété Windows Media. Détermine si une animation est lue en boucle (`TRUE`, par défaut) ou pas (`FALSE`). Lecture/écriture.

Exemple

Cette instruction spécifie que l'acteur Classique doit effectuer une boucle une fois la lecture terminée :

```
-- Syntaxe Lingo
member("Classique").loop = TRUE

// Syntaxe JavaScript
member("Classique").loop = true;
```

Voir aussi

Windows Media

loopBounds

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.loopBounds

// Syntaxe JavaScript
réfObjImageObjet.loopBounds;
```

Description

Propriété d'image-objet QuickTime ; définit les points de boucle internes d'un acteur ou d'une image-objet QuickTime. Les points de la boucle sont spécifiés sous forme de liste Director : [*posInitiale*, *posFinale*].

Les paramètres *posInitiale* et *posFinale* doivent répondre aux exigences suivantes :

- Ils doivent tous deux être des nombres entiers spécifiant le temps en nombre de battements Director.
- Leurs valeurs doivent être comprises entre 0 et la durée de l'acteur QuickTime.
- La position de départ doit être inférieure à la position de fin.

Si les exigences ci-dessus ne sont pas satisfaites, l'animation QuickTime est lue en boucle pendant sa durée complète.

La propriété `loopBounds` n'a aucun effet lorsque la propriété `loop` de l'animation a la valeur `FALSE`. Si la propriété `loop` est définie sur `TRUE` pendant la lecture de l'animation, la lecture se poursuit. Director utilise les règles suivantes pour décider de la lecture en boucle de l'animation :

- Lorsque la position de fin spécifiée par `loopBounds` est atteinte, l'animation est relue en boucle à partir de la position de départ.
- Lorsque la fin de l'animation est atteinte, l'animation est relue en boucle à partir de son début.

Si la propriété `loop` est désactivée pendant la lecture de l'animation, la lecture se poursuit. Director arrête la lecture lorsqu'il atteint la fin de l'animation.

Cette propriété peut être testée et définie. La valeur par défaut est `[0,0]`.

Exemple

Le script d'image-objet suivant définit les positions de départ et de fin de la boucle à l'intérieur d'une image-objet QuickTime. Ces positions sont exprimées en secondes et sont ensuite converties en battements (multiplication par 60).

```
-- Syntaxe Lingo
on beginSprite me
  sprite(me.spriteNum).loopBounds = [(16 * 60),(32 * 60)]
end

// Syntaxe JavaScript
function beginSprite() {
  sprite(me.spriteNum).loopBounds = list((16 * 60),(32 * 60));
}
```

loopCount

Utilisation

```
-- Syntaxe Lingo
réfObjPisteAudio.loopCount

// Syntaxe JavaScript
réfObjPisteAudio.loopCount;
```

Description

Propriété de piste audio ; spécifie le nombre d'occurrences de lecture en boucle du son courant d'une piste audio. En lecture seule.

La valeur par défaut de cette propriété est 1 pour les sons simplement placés en file d'attente et ne contenant pas de boucle interne.

Vous pouvez définir la lecture en boucle d'une partie d'un son en passant les paramètres `loopStartTime`, `loopEndTime` et `loopCount` avec la méthode `queue()` ou `setPlayList()`. Il s'agit là des seules méthodes permettant de définir cette propriété.

Si la valeur `loopCount` est définie sur 0, la boucle se répète à l'infini. Si la propriété `loop` de l'acteur son est définie sur `TRUE`, la valeur `loopCount` revient à 0.

Exemple

Le gestionnaire suivant place en file d'attente et lit deux sons dans la piste audio 2. Le premier son, l'acteur intro, est exécuté cinq fois sur une durée comprise entre 8 et 8,9 secondes. Le second son, l'acteur Crédits, est exécuté trois fois en boucle. Toutefois, aucune valeur `#loopStartTime` ni `#loopEndTime` n'étant définie, ces valeurs passent respectivement par défaut à `#startTime` et `#endTime`.

```
-- Syntaxe Lingo
on lireLaMusique
  sound(2).queue([#member:member("Intro"), #startTime:3000, \
  #loopCount:5, #loopStartTime:8000, #loopEndTime:8900])
  sound(2).queue([#member:member("Crédits"), #startTime:3000, \
  #endTime:3000, #loopCount:3])
  sound(2).play()
end lireLaMusique

// Syntaxe JavaScript
function lireLaMusique() {
  sound(2).queue(propList("member",member("Intro"), symbol("startTime"),3000,
  symbol("loopCount"),5, symbol("loopStartTime"),8000,
  symbol("loopEndTime"),8900));
  sound(2).queue(propList("member",member("Crédits"),
  symbol("startTime)",3000,
  symbol("endTime"),3000, symbol("loopCount"),3]);
  sound(2).play();
}
```

Voir aussi

`loopEndTime`, `loopStartTime`, `queue()`, `setPlayList()`, `Piste audio`

loopEndTime

Utilisation

```
-- Syntaxe Lingo
réfObjPisteAudio.loopEndTime

// Syntaxe JavaScript
réfObjPisteAudio.loopEndTime;
```

Description

Propriété de piste audio ; spécifie la position temporelle de fin, en millisecondes, de la boucle définie dans le son courant d'une piste audio. En lecture seule.

Cette propriété dispose d'une valeur à virgule flottante, permettant de mesurer et de contrôler la lecture du son en fraction de millisecondes.

Cette propriété ne peut être définie que si elle est passée comme propriété dans une commande `queue()` ou `setPlaylist()`.

Exemple

Le gestionnaire suivant lit l'acteur son Intro sur la piste audio 2. La lecture est exécutée cinq fois en boucle entre le point 8 secondes et le point 8,9 secondes du son.

```
-- Syntaxe Lingo
on lireLaMusique
  sound(2).play([#member:member("Intro"), #startTime:3000, #loopCount:5 \
  #loopStartTime:8000, #loopEndTime:8900])
end lireLaMusique

// Syntaxe JavaScript
function lireLaMusique() {
  sound(2).play(propList("member",member("Intro"), symbol("startTime"),3000,
  symbol("loopCount"),5, symbol("loopStartTime"),8000,
  symbol("loopEndTime"),8900));
}
```

Voir aussi

`queue()`, `setPlayList()`, Piste audio

loopsRemaining

Utilisation

```
-- Syntaxe Lingo
réfObjPisteAudio.loopsRemaining

// Syntaxe JavaScript
réfObjPisteAudio.loopsRemaining;
```

Description

Propriété de piste audio ; spécifie le nombre d'occurrences restantes de lecture en boucle du son en cours sur une piste audio. En lecture seule.

Si aucune lecture en boucle n'a été définie lors de son placement en file d'attente, cette propriété a une valeur de 0. Si cette propriété est testée immédiatement après le départ de la lecture d'un son, elle renvoie un nombre inférieur de 1 par rapport à celui défini dans la propriété `#loopCount` de la méthode `queue()` ou `setPlayList()`.

Voir aussi

`loopCount`, `queue()`, `setPlayList()`, `Piste audio`

loopStartTime

Utilisation

```
-- Syntaxe Lingo
réfObjPisteAudio.loopStartTime

// Syntaxe JavaScript
réfObjPisteAudio.loopStartTime;
```

Description

Propriété de piste audio ; spécifie la position temporelle de début, en millisecondes, de la boucle définie pour le son courant d'une piste audio. En lecture seule.

Il s'agit d'une valeur à virgule flottante, permettant de mesurer et de contrôler la lecture du son en fraction de millisecondes. La valeur par défaut est la position de départ `startTime` du son si aucune boucle n'est définie.

Cette propriété ne peut être définie que si elle est passée comme propriété dans une méthode `queue()` ou `setPlaylist()`.

Exemple

Le gestionnaire suivant lit l'acteur son Intro sur la piste audio 2. La lecture est exécutée cinq fois en boucle entre le point 8 secondes et le point 8,9 secondes du son.

```
-- Syntaxe Lingo
on lireLaMusique
  sound(2).play([#member:member("Intro"), #startTime:3000, #loopCount:5 \
  #loopStartTime:8000, #loopEndTime:8900])
end lireLaMusique

// Syntaxe JavaScript
function lireLaMusique() {
  sound(2).play(propList("member",member("Intro"), symbol("startTime"),3000,
  symbol("loopCount"),5, symbol("loopStartTime"),8000,
  symbol("loopEndTime"),8900));
}
```

Voir aussi

`queue()`, `setPlayList()`, `Piste audio`, `startTime`

magnitude

Utilisation

```
quelVecteur.magnitude
```

Description

Propriété 3D ; renvoie la magnitude d'un vecteur. La valeur est un nombre à virgule flottante. L'amplitude est la longueur d'un vecteur et est toujours supérieure ou égale à 0.0 (`vector (0, 0, 0)` est égal à 0).

Exemple

L'instruction suivante indique que l'amplitude de monVecteur1 est 100.0000 et celle de monVecteur2 est 141.4214.

```
monVecteur1 = vector(100, 0, 0)
put monVecteur1.magnitude
-- 100.0000
monVecteur2 = vector(100, 100, 0)
put monVecteur2.magnitude
-- 141.4214
```

Voir aussi

`length (3D)`, `identity()`

margin

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.margin

// Syntaxe JavaScript
réfObjActeur.margin;
```

Description

Propriété d'acteur champ : détermine la taille, en pixels, de la marge à l'intérieur de la case du champ.

Exemple

L'instruction suivante définit la marge de la case de l'acteur champ Nouvelles du Jour sur 15 pixels :

```
-- Syntaxe Lingo
member("Nouvelles du Jour").margin = 15

// Syntaxe JavaScript
member("Nouvelles du Jour").margin = 15;
```

markerList

Utilisation

```
-- Syntaxe Lingo
_movie.markerList

// Syntaxe JavaScript
_movie.markerList;
```

Description

Propriété d'animation ; contient une liste de propriétés de script des repères du scénario. En lecture seule.

La liste est au format :

numéroDimage : "numéroDeRepère"

Exemple

L'instruction suivante affiche la liste des repères dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(_movie.markerList)

// Syntaxe JavaScript
put(_movie.markerList);
```

Voir aussi

Animation

mask

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.mask

// Syntaxe JavaScript
réfObjActeur.mask;
```

Description

Propriété d'acteur ; spécifie l'acteur noir et blanc (1 bit) qui servira à masquer des médias rendus au premier plan avec les médias qui se trouvent dans des zones dans lesquelles les pixels du masque sont noirs. La propriété `mask` vous permet de bénéficier des performances d'une vidéo numérique Premier plan pendant la lecture d'une séquence QuickTime dans une zone non rectangulaire. La propriété `mask` n'a aucun effet sur les acteurs qui ne sont pas rendus au premier plan.

Director aligne toujours le point d'alignement de l'acteur masque avec le coin supérieur gauche de l'image-objet séquence QuickTime. N'oubliez pas de définir le point d'alignement d'un bitmap sur le coin supérieur gauche, car il est défini sur le centre par défaut. Le point d'alignement de l'acteur QuickTime ne peut pas être défini ailleurs que sur le coin supérieur gauche. L'acteur masque ne peut pas être déplacé et n'est pas affecté par les propriétés `center` et `crop` de l'acteur qui lui est associé.

Pour obtenir des résultats optimaux, définissez la propriété `mask` d'un acteur QuickTime avant l'apparition de ses images-objets sur la scène dans le gestionnaire d'événement `on beginSprite`. En effet, la définition ou la modification de la propriété `mask` d'un acteur alors que celui-ci se trouve déjà sur la scène peut produire des résultats inattendus (par exemple, le masque peut apparaître sous forme d'une image figée de l'animation numérique au moment où la propriété `mask` a pris effet).

L'utilisation des masques est une fonction avancée, qui exigera vraisemblablement plusieurs essais avant d'être maîtrisée.

Cette propriété peut être testée et définie. Pour supprimer un masque, donnez à la propriété `mask` une valeur de `0`.

Exemple

Le script d'image suivant définit le masque d'une image-objet QuickTime avant que Director ne commence à dessiner l'image :

```
-- Syntaxe Lingo
on prepareFrame
  member("Voyeur").mask = member("Serrure")
end

// Syntaxe JavaScript
function prepareFrame() {
  member("Voyeur").mask = member("Serrure");
}
```

Voir aussi

`invertMask`

maxInteger

Utilisation

```
the maxInteger
```

Description

Propriété système ; renvoie le nombre entier le plus élevé supporté par le système. Sur la plupart des ordinateurs personnels, ce nombre est 2 147 483 647 (2 à la puissance 31, moins 1).

Cette propriété peut servir à initialiser des variables utilisées dans des boucles ou pour limiter certains tests.

Pour utiliser des nombres supérieurs à la plage d'entiers utilisables, utilisez des nombres à virgule flottante. Ces nombres ne sont pas traités aussi rapidement que les nombres entiers, mais permettent d'utiliser une plus grande plage de valeurs.

Exemple

L'instruction suivante crée un tableau dans la fenêtre Messages contenant les valeurs décimales maximales pouvant être représentées par un certain nombre de chiffres binaires :

```
on afficherLesValeursMax
  b = 31
  v = the maxInteger
  repeat while v > 0
    put b && "-" && v
    b = b-1
    v = v/2
  end repeat
end
```

maxSpeed

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  emitter.maxSpeed
```

Description

Propriété 3D ; utilisée avec une ressource de modèle de type `#particle`, cette propriété permet d'obtenir et de définir la vitesse maximum à laquelle les particules sont émises. La vitesse initiale de chaque particule est sélectionnée de façon aléatoire et est comprise entre les propriétés `minSpeed` et `maxSpeed` de l'émetteur.

Cette valeur est un nombre à virgule flottante et doit être supérieure à 0.0.

Exemple

Dans l'exemple suivant, systèmeThermique est une ressource de modèle de type `#particle`. L'instruction suivante donne à la propriété `maxSpeed` de systèmeThermique la valeur 15, qui entraîne les particules les plus rapides de systèmeThermique à se déplacer relativement rapidement. Dans un système de particules donné, plus une particule se déplace rapidement, plus elle couvre de distance.

```
member("Feux").modelResource("systèmeThermique").emitter.maxSpeed=15
```

Voir aussi

`minSpeed`, `emitter`

media

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.media

// Syntaxe JavaScript
réfObjActeur.media;
```

Description

Propriété d'acteur ; identifie l'acteur spécifié comme un jeu de numéros. Lecture/écriture.

La définition de cette propriété nécessitant une grande quantité de mémoire, il est préférable de ne l'utiliser que pendant la programmation.

Vous pouvez utiliser la propriété `media` pour copier le contenu d'un acteur dans un autre en donnant à la valeur `media` du deuxième acteur une valeur identique à la valeur `media` du premier acteur.

Pour un acteur boucle, la propriété `media` spécifie une sélection d'images et de pistes du scénario.

Pour permuter des médias dans une projection, il est plus efficace de définir la propriété d'image-objet `member`.

Exemple

L'instruction suivante copie le contenu de l'acteur leverDeSoleil dans l'acteur Aube en définissant la valeur de la propriété d'acteur `media` de Aube sur la même valeur que la propriété d'acteur `media` de leverDeSoleil :

```
-- Syntaxe Lingo
member("Aube").media = member("leverDeSoleil").media

// Syntaxe JavaScript
member("Aube").media = member("leverDeSoleil").media;
```

Voir aussi

Acteur

mediaReady

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.mediaReady

// Syntaxe JavaScript
réfObjActeur.mediaReady;
```

Description

Propriété d'acteur ; détermine si le contenu d'un acteur, d'un fichier d'animation ou de bibliothèque de distribution, ou un acteur lié a été complètement téléchargé depuis Internet et est disponible sur le disque dur local (`TRUE`) ou non (`FALSE`). En lecture seule.

Cette propriété n'est utile que lors du transfert en flux continu d'un fichier d'animation ou de bibliothèque de distribution. Pour activer le transfert d'une animation, choisissez l'option Lire pendant le téléchargement (option par défaut) dans la boîte de dialogue Propriétés de lecture de l'animation du menu Modification.

Pour une démonstration de la propriété `mediaReady`, consultez l'animation Shockwave en flux continu dans l'Aide de Director.

Exemple

L'instruction suivante change les acteurs lorsque l'acteur souhaité est récupéré et disponible localement :

```
-- Syntaxe Lingo
if member("fond").mediaReady = TRUE then
  sprite(2).member = member("fond").number
end if

// Syntaxe JavaScript
if (member("fond").mediaReady == true) {
  sprite(2).member = member("fond").number;
}
```

Voir aussi

Acteur

mediaStatus (DVD)

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.mediaStatus

// Syntaxe JavaScript
réfObjDvd.mediaStatus;
```

Description

Propriété DVD ; renvoie un symbole qui indique l'état courant du lecteur DVD. En lecture seule.

Les symboles possibles sont les suivants :

Symbole	Description
#stopped	Le DVD est arrêté.
#playing	Le DVD est en cours de lecture.
#paused	Le DVD est en pause.
#scanning	Le DVD effectue un balayage.
#uninitialized	Le DVD n'est pas initialisé.
#volumeInvalid	Le DVD spécifié n'est pas valide.
#volumeUknown	Le DVD n'existe pas ou il n'y a pas de disque dans le lecteur.
#systemSoftwareMissing	Les décodeurs DVD ne sont pas installés.
#systemSoftwareError	Le logiciel système requis pour la lecture du DVD a renvoyé une erreur.

Voir aussi

DVD

mediaStatus (RealMedia, Windows Media)

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.mediaStatus

// Syntaxe JavaScript
réfObjActeurOuImageObjet.mediaStatus;
```

Description

Propriété d'acteur et d'image-objet RealMedia et Windows Media ; permet d'obtenir un symbole représentant l'état du flux RealMedia ou Windows Media. En lecture seule.

La valeur de cette propriété peut changer pendant la lecture.

Les valeurs correctes de cette propriété sont les suivantes :

- `#closed` indique que l'acteur RealMedia ou Windows Media n'est pas actif. La valeur de `mediaStatus` reste `#closed` jusqu'au début de la lecture.
- `#connecting` indique qu'une connexion au train RealMedia ou Windows Media est en cours d'établissement.
- `#opened` indique qu'une connexion au flux RealMedia ou Windows Media a été établie et est ouverte. Il s'agit d'un état transitoire, rapidement suivi par `#buffering`.
- `#buffering` indique que le flux RealMedia ou Windows Media est en cours de chargement dans le tampon de lecture. Une fois la mise en tampon terminée (`percentBuffered` est égal à 100), la lecture du flux démarre si la propriété `pausedAtStart` a pour valeur `FALSE`. Pour plus d'informations, consultez `percentBuffered`.
- `#playing` indique que le flux RealMedia ou Windows Media est en cours de lecture.
- `#seeking` indique que la lecture a été interrompue par la commande `seek`.
- `#paused` indique que la lecture a été interrompue, par le bouton Arrêter de la fenêtre RealMedia ou Windows Media, ou par l'invocation de la méthode `pause()` dans un script.
- `#error` indique que le flux n'a pas pu être connecté, mis en tampon ou lu. La propriété `lastError` indique l'erreur.

En fonction de la valeur de l'acteur `state (RealMedia)`, une autre valeur de la propriété `mediaStatus` est renvoyée. Chaque valeur `mediaStatus` correspond à une seule valeur `state`.

Exemple

Les exemples suivants indiquent que l'élément RealMedia de l'image-objet 2 et l'acteur Real sont en cours de lecture.

```
-- Syntaxe Lingo
put(sprite(2).mediaStatus) -- #playing
put(member("Real").mediaStatus) -- #playing

// Syntaxe JavaScript
put(sprite(2).mediaStatus); // #playing
put(member("Real").mediaStatus); // #playing
```

Voir aussi

`state (RealMedia)`, `percentBuffered`, `lastError`

mediaXtraList

Utilisation

```
-- Syntaxe Lingo
_player.mediaXtraList

// Syntaxe JavaScript
_player.mediaXtraList;
```

Description

Propriété de lecteur ; renvoie une liste linéaire de tous les Xtras de médias disponibles sur le lecteur Director. En lecture seule.

Exemple

Cette instruction affiche, dans la fenêtre Messages, tous les Xtras de médias disponibles sur le lecteur Director.

```
-- Syntaxe Lingo
put(_player.mediaXtraList)

// Syntaxe JavaScript
put(_player.mediaXtraList);
```

Voir aussi

Types de médias, Lecteur, scriptingXtraList, toolXtraList, transitionXtraList, xtraList (lecteur)

member

Utilisation

```
member(quelActeur).texture(quelleTexture).member
member(quelActeur).model(quelModèle).shader.texture.member
member(quelActeur).model(quelModèle).shaderList\
  [indexDeListeDeMatériaux].textureList[indexDeListeDeTextures].member
```

Description

Propriété de texture 3D ; si la texture est de type #fromCastMember, cette propriété indique l'acteur utilisé comme source de texture.

Cette propriété peut être testée et définie.

Si la texture est de type #importedFromFile, la valeur de cette propriété est void et ne peut pas être définie. Si la texture est de type #fromImageObject, la valeur de cette propriété est void, mais peut néanmoins être définie.

Exemple

Le code Lingo suivant ajoute une nouvelle texture. La seconde instruction indique que l'acteur utilisé pour créer la texture gbTexture est l'acteur 16 de la distribution 1.

```
member("séquence").newTexture("gbTexture", #fromCastmember, \
  member(16, 1))
put member("séquence").texture("Texture").member
-- (member 16 of castLib 1)
```

member (distribution)

Utilisation

```
-- Syntaxe Lingo
réfObjDistribution.member[nomOuNumDActeur]

// Syntaxe JavaScript
réfObjDistribution.member[nomOuNumDActeur]
```

Description

Propriété de distribution ; fournit un accès nommé ou indexé aux acteurs d'une bibliothèque de distribution. En lecture seule.

L'argument *nomOuNumDActeur* peut être une chaîne désignant l'acteur par son nom ou un entier spécifiant l'acteur par un numéro.

Exemple

L'exemple suivant donne accès au deuxième acteur de la bibliothèque de distribution Interne.

```
-- Syntaxe Lingo
monActeur = castLib("Interne").member[2]

// Syntaxe JavaScript
var monActeur = castLib("Interne").member[2];
```

Voir aussi

Bibliothèque de distribution

member (animation)

Utilisation

```
-- Syntaxe Lingo
_movie.member[nomOuNumDActeur]

// Syntaxe JavaScript
_movie.member[nomOuNumDActeur];
```

Description

Propriété d'animation ; fournit un accès nommé ou indexé aux acteurs de la bibliothèque de distribution d'une animation. En lecture seule.

L'argument *nomOuNumDActeur* peut être une chaîne désignant l'acteur par son nom ou un entier spécifiant l'acteur par un numéro.

Exemple

L'instruction suivante accède à un acteur par nom et par numéro, puis affecte au résultat la variable `monActeur`.

```
-- Syntaxe Lingo
monActeur = _movie.member[2] - accès par numéro
myMember = _movie.member["Athlète"] - accès par nom

// Syntaxe JavaScript
var monActeur = _movie.member[2]; // accès par numéro;
var myMember = _movie.member["Athlète"]; // accès par nom;
```

Voir aussi

`Animation`

member (piste audio)

Utilisation

```
-- Syntaxe Lingo
réfObjPisteAudio.member

// Syntaxe JavaScript
réfObjPisteAudio.member;
```

Description

Propriété de piste audio ; spécifie l'acteur audio en cours de lecture sur une piste audio. En lecture seule.

Cette propriété renvoie `VOID` (Lingo) ou `null` (syntaxe JavaScript) si aucun son n'est en cours de lecture.

Exemple

L'instruction suivante affiche le nom de l'acteur correspondant au son lu sur la piste2 dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(sound(2).member)

// Syntaxe JavaScript
put(sound(2).member);
```

Voir aussi

`Piste audio`

member (image-objet)

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.member

// Syntaxe JavaScript
réfObjImageObjet.member;
```

Description

Propriété d'image-objet ; spécifie l'acteur et la bibliothèque de distribution d'une image-objet. Lecture/écriture.

La propriété d'image-objet `member` est différente de la propriété d'image-objet `spriteNum`, qui ne spécifie que le numéro de l'image-objet pour identifier sa position dans la bibliothèque de distribution, mais ne spécifie pas la bibliothèque de distribution elle-même. La propriété `member` est également différente de la propriété `mouseMember` de l'objet Mouse, qui ne spécifie pas la bibliothèque de distribution d'une image-objet.

Lorsque vous affectez la propriété d'image-objet `member`, utilisez l'un des formats suivants :

- Spécifiez la description complète de l'acteur et de la bibliothèque de distribution (*réfObjImageObjet*`.member = member(`*numActeurInt* `{`, *nomOuNumBibliothèqueDeDistribution*`)`).
- Spécifiez le nom de l'acteur (*réfObjImageObjet*`.member = member("`*nomActeurChaîne*").
- Spécifiez le nombre entier unique qui inclut toutes les bibliothèques de distributions et correspond à la propriété `mouseMember` (*réfObjImageObjet*`.member = 132`) .

Si vous n'utilisez que le nom de l'acteur, Director trouve le premier acteur portant ce nom dans toutes les bibliothèques de distributions courantes. Si ce nom se répète dans deux bibliothèques de distributions, seul le premier nom est utilisé.

Pour spécifier un acteur par son numéro uniquement lorsqu'il existe plusieurs distributions, utilisez la propriété d'image-objet `memberNum`, qui change la position de l'acteur dans sa bibliothèque de distribution sans affecter la bibliothèque de distribution de l'image-objet (*réfObjImageObjet*`.memberNum = 10`).

L'acteur affecté à une piste d'image-objet n'est qu'une des propriétés de cette image-objet. Celle-ci possède d'autres propriétés, qui varient selon le type de médias de cette piste du scénario. Par exemple, si vous remplacez un bitmap par une forme vide en définissant la propriété d'image-objet `member`, la propriété `lineSize` de l'image-objet forme ne change pas automatiquement. Vous ne verrez probablement pas la forme.

Des problèmes de correspondance semblables peuvent se produire si vous changez l'acteur d'une image-objet champ en acteur vidéo. Il est généralement plus utile et plus sûr de remplacer les acteurs avec des acteurs du même type. Par exemple, remplacer des images-objets bitmap par des acteurs bitmap.

Exemple

L'instruction suivante affecte l'acteur 3 de la distribution 4 à l'image-objet 15 :

```
-- Syntaxe Lingo
sprite(15).member = member(3, 4)

// Syntaxe JavaScript
sprite(15).member = member(3, 4);
```

Le gestionnaire suivant utilise la fonction `mouseMember` dans la propriété `sprite.member` afin de découvrir si la souris est positionnée sur une image-objet spécifique :

```
-- Syntaxe Lingo
on exitFrame
  mm = _mouse.mouseMember
  cible = sprite(1).member
  if (cible = mm) then
    put("au-dessus de la zone référencée.")
    _movie.go(_movie.frame)
  end if
end

// Syntaxe JavaScript
function exitFrame() {
  var mm = _mouse.mouseMember;
  var cible = sprite(1).member;
  if (cible == mm) then
    put("au-dessus de la zone référencée.");
    _movie.go(_movie.frame);
  }
}
```

Voir aussi

`lineSize`, `mouseMember`, `Image-objet`, `spriteNum`

memorySize

Utilisation

```
the memorySize
```

Description

Propriété système ; renvoie la quantité totale de mémoire affectée au programme, qu'elle soit disponible ou non. Elle est utile pour vérifier la quantité de mémoire minimale requise. La valeur est exprimée en octets.

Sous Windows, cette valeur correspond à la mémoire physique disponible ; sur Macintosh, elle représente la partition complète allouée à l'application.

Exemple

L'instruction suivante vérifie si l'ordinateur alloue moins de 500 Ko de mémoire et, le cas échéant, affiche un message d'alerte :

```
if the memorySize < 500 * 1024 then alert "Mémoire insuffisante."
```

Voir aussi

`freeBlock()`, `freeBytes()`, `ramNeeded()`, `size`

meshDeform (modificateur)

Utilisation

```
member(quelActeur).model(quelModèle).meshDeform.nomDePropriété
```

Description

Modificateur 3D ; permet de contrôler les différents aspects de la structure de maille du modèle référencé. Lorsque vous ajoutez le modificateur `#meshDeform` (à l'aide de la commande `addModifier`) à un modèle, vous avez accès aux propriétés suivantes du modificateur `#meshDeform` :

Remarque : Pour plus d'informations sur ces propriétés, consultez les entrées correspondantes (et qui apparaissent sous la section Voir aussi de cette entrée).

- `face.count` renvoie le nombre total de faces du modèle référencé.
- `mesh.count` renvoie le nombre de mailles du modèle référencé.
- `mesh[index]` permet d'accéder aux propriétés de la maille spécifiée.

Exemple

L'instruction suivante affiche le nombre de faces du modèle gbFace.

```
put member("Univers 3D").model("gbFace").meshDeform.face.count
-- 432
```

L'instruction suivante affiche le nombre de mailles du modèle gbFace.

```
put member("Univers 3D").model("gbFace").meshDeform.mesh.count
-- 2
```

L'instruction suivante affiche le nombre de faces de la deuxième maille du modèle gbFace :

```
put member("Univers 3D").model("gbFace").meshDeform.mesh[2].face.count
-- 204
```

Voir aussi

`mesh (propriété)`, `addModifier`

milliseconds

Utilisation

```
-- Syntaxe Lingo
_system.milliseconds

// Syntaxe JavaScript
_system.milliseconds;
```

Description

Propriété système ; renvoie l'heure courante en millisecondes (1/1000ème de seconde). En lecture seule.

Le compte commence à partir du démarrage de l'ordinateur.

Exemple

L'instruction suivante convertit des millisecondes en secondes et minutes en divisant le nombre de millisecondes par 1 000, puis le résultat par 60, et affecte la variable `minutesCourantes` au résultat de l'opération :

```
-- Syntaxe Lingo
secondesCourantes = _system.milliseconds/1000
minutesCourantes = secondesCourantes/60

// Syntaxe JavaScript
var secondesCourantes = (_system.milliseconds / 1000);
var minutesCourantes = secondesCourantes/60;
```

La précision du calcul dépend de l'ordinateur et du système d'exploitation.

Le gestionnaire suivant compte les millisecondes et affiche un message d'alerte si vous travaillez depuis trop longtemps.

```
-- Syntaxe Lingo
on idle
  if (_system.milliseconds > (1000 * 60 * 60 * 4)) then
    _player.alert("Faites une pause")
  end if
end

// Syntaxe JavaScript
function idle() {
  if (_system.milliseconds > (1000 * 60 * 60 * 4)) {
    _player.alert("Faites une pause");
  }
}
```

Voir aussi

Système

minSpeed

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).
  emitter.minSpeed
```

Description

Propriété 3D ; utilisée avec une ressource de modèle de type `#particle`, cette propriété permet d'obtenir et de définir la vitesse minimum à laquelle les particules sont émises. La vitesse initiale de chaque particule est sélectionnée de façon aléatoire et est comprise entre les propriétés `minSpeed` et `maxSpeed` de l'émetteur.

Cette valeur est un nombre à virgule flottante et doit être supérieure à 0.0.

Exemple

Dans l'exemple suivant, systèmeThermique est une ressource de modèle de type #particle. L'instruction suivante donne à la propriété minSpeed de systèmeThermique la valeur 5, qui entraîne les particules les plus lentes de systèmeThermique à se déplacer relativement lentement. Dans un système de particules donné, plus une particule se déplace lentement, moins elle couvre de distance.

```
member("Chauffage").modelResource("systèmeThermique").emitter.\
  minSpeed = 5
```

Voir aussi

maxSpeed, emitter

missingFonts

Utilisation

```
member(acteurTexte).missingFonts
```

Description

Propriété d'acteur texte ; contient une liste des noms de polices utilisées dans le texte mais non disponibles sur le système.

Cette propriété permet aux développeurs de vérifier si une police spécifique est disponible ou non pendant l'exécution.

Cette propriété peut être testée, mais pas définie.

Voir aussi

substituteFont

mode (émetteur)

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  emitter.mode
```

Description

Propriété 3D ; utilisée avec une ressource de modèle de type #particle, cette propriété permet d'obtenir et de définir la propriété mode de l'émetteur de particules de la ressource.

Cette propriété peut avoir la valeur #burst ou #stream (valeur par défaut). Lorsque mode a pour valeur #burst, toutes les particules sont émises en même temps, alors qu'avec la valeur #stream, un groupe de particules est émis à chaque image. Le nombre de particules émises à chaque image est déterminé au moyen de l'équation suivante :

```
particulesParImage = objetRessource.émetteur.nombreDeParticules \
  (objetRessource.duréeDeVie x millisecondesParImageRendue)
```

Exemple

Dans l'exemple suivant, systèmeThermique est une ressource de modèle de type `#particle`. L'instruction suivante donne à la propriété `emitter.mode` de systèmeThermique la valeur `#burst`, ce qui entraîne l'émission en éclats des particules de systèmeThermique. Pour créer une seule explosion de particules, choisissez `emitter.mode = #burst` et `emitter.loop = 0`.

```
member("Feux").modelResource("systèmeThermique").emitter.mode = #burst
```

Voir aussi

`emitter`

mode (collision)

Utilisation

```
member(quelActeur).model(quelModèle).collision.mode
```

Description

Propriété 3D de modificateur de collision ; indique la géométrie à utiliser dans l'algorithme de détection de collision. L'utilisation d'une géométrie plus simple, telle qu'une sphère de délimitation, permet une meilleure performance. Les valeurs possibles de cette propriété sont :

- `#mesh` utilise la géométrie de maille réelle de la ressource de modèle. Cette valeur offre une précision unitriangulaire et est généralement plus lente que `#box` ou `#sphere`.
- `#box` utilise la boîte de délimitation du modèle. Cette valeur est utile pour les objets, tels qu'un mur, qui logent plus facilement dans une boîte que dans une sphère.
- `#sphere` est le mode le plus rapide, car il utilise la sphère de délimitation du modèle. Il s'agit de la valeur par défaut de cette propriété.

Exemple

Les instructions suivantes ajoutent le modificateur de collision au modèle `votreModèle` et donnent à la propriété le mode `#mesh` :

```
member("3d").model("votreModèle").addModifier(#collision)
member("3d").model("votreModèle").collision.mode = #mesh
```

model

Utilisation

```
member(quelActeur).model(quelModèle)
member(quelActeur).model[index]
member(quelActeur).model.count
member(quelActeur).model(quelModèle).nomDePropriété
member(quelActeur).model[index].nomDePropriété
```

Description

Commande 3D ; renvoie le modèle trouvé dans l'acteur référencé dont le nom est spécifié par *quelModèle*, ou trouvé à la position d'index spécifiée par *index*. Si aucun modèle n'existe pour le paramètre spécifié, la commande renvoie `void`. Tout comme `model.count`, la commande renvoie le nombre de modèles détectés dans l'acteur référencé. Cette commande permet également d'accéder aux propriétés du modèle spécifié.

Les comparaisons de noms de modèle ne sont pas sensibles à la hauteur de casse. La position d'index d'un modèle particulier peut changer lorsque des objets dans des positions inférieures sont supprimés.

Si aucun modèle n'utilise le nom spécifié ou n'est trouvé à la position d'index spécifiée, cette commande renvoie `void`.

Exemple

L'instruction suivante enregistre une référence au modèle Lecteur dans la variable ceModèle :

```
ceModèle = member("Univers3D").model("Lecteur")
```

L'instruction suivante enregistre une référence au huitième modèle de l'acteur Univers3D dans la variable ceModèle.

```
ceModèle = member("Univers3D").model[8]
```

L'instruction suivante indique que quatre modèles se trouvent dans l'acteur de l'image-objet 1.

```
put sprite(1).member.model.count
-- 4
```

modelA

Utilisation

```
collisionData.modelA
```

Description

Propriété 3D `collisionData` ; indique un des modèles impliqués dans une collision, l'autre modèle étant `modelB`.

L'objet `collisionData` est envoyé comme argument avec les événements `#collideWith` et `#collideAny` au gestionnaire spécifié dans les commandes `registerForEvent`, `registerScript` et `setCollisionCallback`.

Les événements `#collideWith` et `#collideAny` sont envoyés lors d'une collision entre deux modèles associés à des modificateur de collision. La propriété `resolve` des modificateurs de modèles doit être `TRUE`.

Cette propriété peut être testée, mais pas définie.

Exemple

L'exemple suivant est constitué de trois parties. La première partie est la première ligne de code, qui enregistre le gestionnaire `#placerDétails` de l'événement `#collideAny`. La deuxième partie est le gestionnaire `#placerDétails`. Lorsque les deux modèles de l'acteur maSéquence entrent en collision, le gestionnaire `#placerDétails` est appelé, et l'argument `collisionData` lui est envoyé. Ce gestionnaire affiche les propriétés de `modelA` et `modelB` de l'objet `collisionData` dans la fenêtre Messages. La troisième partie de l'exemple affiche les résultats de la fenêtre Messages. Le modèle nommé balleVerte est `modelA` et balleJaune est `modelB`.

```
member("maSéquence").registerForEvent(#collideAny, #placerDétails, 0)
on placerDétails me, collisionData
  put collisionData.modelA
  put collisionData.modelB
end
-- model("balleVerte")
-- model("balleJaune")
```

Voir aussi

registerScript(), registerForEvent(), sendEvent, modelB, setCollisionCallback()

modelB

Utilisation

```
collisionData.modelB
```

Description

Propriété 3D `collisionData` ; indique un des modèles impliqués dans une collision, l'autre modèle étant `modelA`.

L'objet `collisionData` est envoyé comme argument avec les événements `#collideWith` et `#collideAny` au gestionnaire spécifié dans les commandes `registerForEvent`, `registerScript` et `setCollisionCallback`.

Les événements `#collideWith` et `#collideAny` sont envoyés lors d'une collision entre deux modèles associés à des modificateur de collision. La propriété `resolve` des modificateurs des modèles doit être `TRUE`.

Cette propriété peut être testée, mais pas définie.

Exemple

L'exemple suivant est constitué de trois parties. La première partie est la première ligne de code, qui enregistre le gestionnaire `#placerDétails` de l'événement `#collideAny`. La deuxième partie est le gestionnaire `#placerDétails`. Lorsque les deux modèles de l'acteur maSéquence entrent en collision, le gestionnaire `#placerDétails` est appelé, et l'argument `collisionData` lui est envoyé. Ce gestionnaire affiche les propriétés de `modelA` et `modelB` de l'objet `collisionData` dans la fenêtre Messages. La troisième partie de l'exemple affiche les résultats de la fenêtre Messages. Le modèle nommé balleVerte est `modelA` et balleJaune est `modelB`.

```
member("maSéquence").registerForEvent(#collideAny, #placerDétails, 0)
on placerDétails me, collisionData
  put collisionData.modelA
  put collisionData.modelB
end
-- model("balleVerte")
-- model("balleJaune")
```

Voir aussi

registerScript(), registerForEvent(), sendEvent, modelA, collisionNormal, setCollisionCallback()

modelResource

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle)
member(quelActeur).modelResource[index]
member(quelActeur).modelResource.count
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  nomDePropriété
member(quelActeur).modelResource[index].nomDePropriété
```

Description

Commande 3D ; renvoie la ressource de modèle trouvée dans l'acteur référencé dont le nom est spécifié par *quelleRessourceDeModèle*, ou trouvée à la position d'index spécifiée par le paramètre *index*. Si aucune ressource de modèle n'existe pour le paramètre spécifié, la commande renvoie `void`. Tout comme `modelResource.count`, la commande renvoie le nombre de ressources de modèle détectées dans l'acteur référencé. Cette commande permet également d'accéder aux propriétés du modèle spécifié.

Les comparaisons de chaînes de noms de ressources de modèle ne sont pas sensibles à la hauteur de casse. La position d'index d'une ressource de modèle particulière peut changer lorsque des objets dans des positions inférieures sont supprimés.

Exemple

L'instruction suivante enregistre une référence à la ressource de modèle MaisonA dans la variable `cetteRessourceDeModèle`.

```
cetteRessourceDeModèle = member("Univers3D").modelResource("MaisonA")
```

L'instruction suivante enregistre une référence à la quatorzième ressource de modèle de l'acteur Univers3D dans la variable `cetteRessourceDeModèle`.

```
cetteRessourceDeModèle = member("Univers3D").modelResource[14]
```

L'instruction suivante indique que dix ressources de modèle se trouvent dans l'acteur de l'image-objet 1.

```
put sprite(1).member.modelResource.count
--10
```

modified

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.modified

// Syntaxe JavaScript
réfObjActeur.modified;
```

Description

Propriété d'acteur ; indique si un acteur a été modifié depuis sa lecture à partir d'un fichier d'animation. En lecture seule.

- Lorsque la propriété `modified` a pour valeur `TRUE` (`1`), l'acteur a été modifié depuis sa lecture à partir du fichier de l'animation.

- Lorsque la propriété `modified` a pour valeur `FALSE` (`0`), l'acteur n'a pas été modifié depuis sa lecture à partir du fichier de l'animation.

Exemple

L'instruction suivante vérifie si l'acteur Introduction a été modifié depuis sa lecture à partir du fichier de l'animation :

```
-- Syntaxe Lingo
if (member("Introduction").modified) then
  _player.alert("L'introduction a été modifiée")
else
  _player.alert("L'introduction n'a pas été modifiée")
end if

// Syntaxe JavaScript
if (member("Introduction").modified) {
  _player.alert("L'introduction a été modifiée");
}
else {
  _player.alert("L'introduction n'a pas été modifiée");
}
```

Voir aussi

Acteur

modifiedBy

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.modifiedBy

// Syntaxe JavaScript
réfObjActeur.modifiedBy;
```

Description

Propriété d'acteur ; enregistre le nom de la personne qui a effectué la modification la plus récente de cet acteur. En lecture seule.

Le nom provient des coordonnées fournies au cours de l'installation de Director. Vous pouvez modifier cette information dans la boîte de dialogue Préférences générales de Director.

Cette propriété s'avère utile pour assurer le suivi et la coordination des projets Director à plusieurs auteurs et peut également être affichée dans le volet Acteur de l'inspecteur des propriétés.

Exemple

L'instruction suivante affiche le nom de la personne qui a effectué la modification la plus récente de l'acteur 1 :

```
-- Syntaxe Lingo
put(member(1).modifiedBy)

// Syntaxe JavaScript
put(member(1).modifiedBy);
```

Voir aussi

Acteur

modifiedDate

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.modifiedDate

// Syntaxe JavaScript
réfObjActeur.modifiedDate;
```

Description

Propriété d'acteur ; indique la date et l'heure de la dernière modification de l'acteur, à l'aide de l'heure système de l'ordinateur. En lecture seule.

Cette propriété s'avère utile pour assurer le suivi et la coordination des projets Director. Elle peut également être affichée dans le volet Acteur de l'inspecteur des propriétés ou dans la fenêtre Distribution en mode d'affichage sous forme de liste.

Exemple

L'instruction suivante affiche la date de la modification la plus récente de l'acteur 1 :

```
-- Syntaxe Lingo
put(member(1).modifiedDate)

// Syntaxe JavaScript
put(member(1).modifiedDate);
```

Voir aussi

Acteur

modifier

Utilisation

```
member(quelActeur).model(quelModèle).modifier
member(quelActeur).model(quelModèle).modifier.count
```

Description

Propriété 3D de modèle ; renvoie une liste des modificateurs associés au modèle spécifié. Tout comme `modifier.count`, la commande renvoie le nombre de modificateurs associés au modèle.

Si les modificateurs `toon` et `inker` sont tous les deux appliqués à un modèle, seul celui qui a été ajouté en premier est renvoyé.

Cette propriété peut être testée, mais pas définie. Utilisez les commandes `addModifier` et `removeModifier` pour ajouter des modificateurs aux modèles ou en supprimer.

Exemple

L'instruction suivante indique les modificateurs qui sont associés au modèle Jongleur.

```
put member("Parc").model("Jongleur").modifier
-- [#bonesPlayer, #lod]
```

Voir aussi

modifier[], modifiers, addModifier, removeModifier

modifier[]

Utilisation

```
member(quelActeur).model(quelModèle).modifier[index]
```

Description

Propriété 3D de modèle ; renvoie le type du modificateur situé à la position spécifiée par *index* dans la liste des modificateurs associés au modèle. La valeur renvoyée est un symbole.

Si aucun modificateur n'est situé à la position spécifiée, la valeur de cette propriété est `void`.

Pour plus d'informations sur la liste des modificateurs associés à un modèle, utilisez la propriété `modifier`.

L'accès direct aux propriétés d'un modificateur associé n'est pas supporté par cette commande.

Exemple

```
put member("Univers 3D").model("boîte").modifier[1]
-- #lod
```

Voir aussi

`modifier`, `modifiers`, `addModifier`, `removeModifier`

modifiers

Utilisation

```
getRendererServices().modifiers
```

Description

Propriété 3D globale ; renvoie une liste des modificateurs disponibles pour tous les modèles d'acteurs 3D.

Exemple

L'instruction suivante renvoie la liste de tous les modificateurs actuellement disponibles.

```
put getRendererServices().modifiers
-- [#collision, #bonesPlayer, #keyFramePlayer, #toon, #lod, \
  #meshDeform, #sds, #inker]
```

Voir aussi

`getRendererServices()`, `addModifier`

mostRecentCuePoint

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.mostRecentCuePoint

// Syntaxe JavaScript
réfObjImageObjet.mostRecentCuePoint;
```

Description

Propriété d'acteur et d'image-objet ; pour les images-objets audio, la vidéo numérique QuickTime et les Xtras supportant les points de repère, indique le numéro du point de repère de l'image-objet ou du son rencontré en dernier. Sa valeur correspond au nombre ordinal du point de repère. Si aucun point de repère n'a été passé, sa valeur est de 0.

Les sons Shockwave Audio (SWA) peuvent apparaître sous forme d'images-objets dans les pistes d'images-objets, mais sont lus dans les pistes audio. Il est conseillé de faire référence aux images-objets audio SWA par le numéro de leur piste d'image-objet plutôt que par celui de leur piste audio.

Exemple

L'instruction suivante affiche dans la fenêtre Messages le numéro du point de repère passé en dernier dans l'image-objet de la piste d'image-objet 1 :

```
-- Syntaxe Lingo
put sprite(1).mostRecentCuePoint

// Syntaxe JavaScript
put(sprite(1).mostRecentCuePoint);
```

L'instruction suivante renvoie le nombre ordinal du point de repère le plus récent dans le son en cours de lecture sur la piste audio 2 :

```
-- Syntaxe Lingo
put sound(2).mostRecentCuePoint

// Syntaxe JavaScript
put(sound(2).mostRecentCuePoint);
```

Voir aussi

`cuePointNames`, `isPastCuePoint()`, `cuePointTimes`, `on cuePassed`

motion

Utilisation

```
member(quelActeur).motion(quelMouvement)
member(quelActeur).motion[index]
member(quelActeur).motion.count
```

Description

Commande 3D ; renvoie le mouvement trouvé dans l'acteur référencé dont le nom est spécifié par *quelMouvement*, ou trouvé à la position d'index spécifiée par *index*. Tout comme `motion.count`, cette propriété renvoie le nombre total de mouvements détectés dans l'acteur.

Les comparaisons de chaînes de nom d'objet ne sont pas sensibles à la hauteur de casse. La position d'index d'un mouvement particulier peut changer lorsque des objets dans des positions inférieures sont supprimés.

Si aucun mouvement n'utilise le nom spécifié ou n'est trouvé à la position d'index spécifiée, cette commande renvoie `void`.

Exemple

```
ceMouvement = member("Univers 3D").motion("Aile")
ceMouvement = member("Univers 3D").motion[7]
put member("séquence").motion.count
-- 2
```

Voir aussi

duration (3D), map (3D)

motionQuality

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.motionQuality

// Syntaxe JavaScript
réfObjImageObjet.motionQuality;
```

Description

Propriété d'image-objet QuickTime VR ; qualité de codec utilisée lorsque l'utilisateur clique sur une image-objet QuickTime VR et la fait glisser. La valeur de cette propriété peut être `#minQuality`, `#maxQuality` ou `#normalQuality`.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante donne à `motionQuality` de l'image-objet 1 la valeur `#minQuality`.

```
-- Syntaxe Lingo
sprite(1).motionQuality = #minQuality

// Syntaxe JavaScript
sprite(1).motionQuality = symbol("minQuality");
```

mouseChar

Utilisation

```
-- Syntaxe Lingo
_mouse.mouseChar

// Syntaxe JavaScript
_mouse.mouseChar;
```

Description

Propriété de souris ; pour les images-objets champ, contient le numéro du caractère se trouvant sous le curseur lorsqu'elle est appelée. En lecture seule.

Le compte démarre au début du champ. Si la souris ne se trouve pas sur un champ ou se trouve sur la bordure d'un champ, le résultat est -1.

La valeur de `mouseChar` peut changer dans un gestionnaire ou une boucle. Lorsqu'un gestionnaire ou une boucle utilise cette propriété plusieurs fois, il est d'usage d'appeler cette dernière une seule fois et d'affecter sa valeur à une variable locale.

Exemple

L'instruction suivante vérifie si le pointeur se trouve au-dessus d'une image-objet champ et, si ce n'est pas le cas, remplace le contenu de l'acteur champ Instructions par Veuillez pointer sur un caractère :

```
-- Syntaxe Lingo
if (_mouse.mouseChar = -1) then
  member("Instructions").text = "Veuillez pointer sur un caractère."
end if

// Syntaxe JavaScript
if (_mouse.mouseChar == -1) {
  member("Instructions").text = "Veuillez pointer sur un caractère.";
}
```

L'instruction suivante affecte à la variable `caractèreActuel` le caractère du champ spécifié qui se trouve sous le curseur :

```
-- Syntaxe Lingo
caractèreActuel = member(_mouse.mouseMember).char[_mouse.mouseChar]

// Syntaxe JavaScript
var caractèreActuel = member(_mouse.mouseMember).getProp("char",
  _mouse.mouseChar);
```

Voir aussi

`Souris`, `mouseItem`, `mouseLine`

mouseDown

Utilisation

```
-- Syntaxe Lingo
_mouse.mouseDown

// Syntaxe JavaScript
_mouse.mouseDown;
```

Description

Propriété de souris ; indique si l'utilisateur est en train d'appuyer sur le bouton de la souris (`TRUE`) ou non (`FALSE`). En lecture seule.

Exemple

Le gestionnaire `mouseEnter` suivant lié à une image-objet appelle un gestionnaire si le bouton de la souris est enfoncé lorsque le pointeur entre dans l'image-objet et un autre gestionnaire si le bouton de la souris n'est pas enfoncé lorsque le pointeur entre dans l'image-objet.

```
-- Syntaxe Lingo
on mouseEnter
  if (_mouse.mouseDown) then
    runMouseDownScript
  else
    runMouseUpScript
  end if
end

// Syntaxe JavaScript
function mouseEnter() {
  if (_mouse.mouseDown) {
    runMouseDownScript();
  }
  else {
    runMouseUpScript();
  }

}
```

Voir aussi

Souris, on mouseDown (gestionnaire d'événement), mouseH, mouseUp, on mouseUp (gestionnaire d'événement), mouseV

mouseDownScript

Utilisation

```
the mouseDownScript
```

Description

Propriété système ; spécifie le code Lingo exécuté lorsque l'utilisateur appuie sur le bouton de la souris. Ce code est rédigé sous forme d'une chaîne encadrée de guillemets droits et peut être une simple instruction ou le script d'appel d'un gestionnaire. Sa valeur par défaut est `EMPTY`, ce qui signifie que la propriété `mouseDownScript` n'est associée à aucune expression Lingo.

Lorsque l'utilisateur appuie sur le bouton de la souris et que la propriété `mouseDownScript` est définie, Lingo exécute en premier les instructions spécifiées dans la propriété `mouseDownScript`. A moins que les instructions ne contiennent la commande `pass` autorisant la transmission du message `mouseDown` vers d'autres objets de l'animation, aucun autre gestionnaire `on mouseDown` ne sera exécuté.

L'utilisation de la propriété `mouseDownScript` équivaut à utiliser la commande `when mouseDown then` des versions précédentes de Director.

Pour désactiver les instructions spécifiées pour la propriété `mouseDownScript`, utilisez l'instruction `set the mouseDownScript to empty`.

Cette propriété peut être testée et définie.

Exemple

Dans l'instruction suivante, la tête de lecture passe au repère suivant de l'animation à chaque fois que l'utilisateur appuie sur le bouton de la souris :

```
the mouseDownScript = "go next"
```

Dans l'instruction suivante, l'ordinateur émet un bip sonore à chaque fois que l'utilisateur clique à un endroit quelconque de la scène :

```
the mouseDownScript = "if the clickOn = 0 then beep"
```

L'instruction suivante paramètre la propriété `mouseDownScript` sur le gestionnaire personnalisé *`monGestionnairePersonnalisé`*. Un gestionnaire Lingo personnalisé doit toujours être encadré de guillemets droits lorsqu'il est utilisé avec la propriété `mouseDownScript`.

```
the mouseDownScript = "monGestionnairePersonnalisé"
```

Voir aussi

`stopEvent()`, `mouseUpScript`, `on mouseDown (gestionnaire d'événement)`, `on mouseUp (gestionnaire d'événement)`

mouseH

Utilisation

```
-- Syntaxe Lingo
_mouse.mouseH

// Syntaxe JavaScript
_mouse.mouseH;
```

Description

Propriété de souris ; indique la position horizontale du pointeur de la souris. En lecture seule.

La valeur de `mouseH` correspond au nombre de pixels entre le curseur et le bord gauche de la scène.

La propriété `mouseH` est pratique pour déplacer des images-objets au même niveau horizontal que le pointeur de la souris et pour vérifier si celui-ci se trouve sur la scène. L'utilisation conjointe des propriétés `mouseH` et `mouseV` permet de déterminer la position exacte du curseur.

Exemple

Le gestionnaire suivant place l'image-objet 10 à la position du pointeur et met la scène à jour chaque fois que l'utilisateur appuie sur le bouton de la souris :

```
-- Syntaxe Lingo
on mouseDown
  sprite(10).locH = _mouse.mouseH
  sprite(10).locV = _mouse.mouseV
end

// Syntaxe JavaScript
function mouseDown() {
  sprite(10).locH = _mouse.mouseH;
  sprite(10).locV = _mouse.mouseV;
}
```

L'instruction suivante vérifie si le curseur se trouve à plus de 10 pixels à droite ou à gauche d'un point de départ et, le cas échéant, donne à la variable `glissementSuffisant` la valeur `TRUE` :

```
-- Syntaxe Lingo
startH = 7

if (abs(_mouse.mouseH - startH) > 10) then
  glissementSuffisant = TRUE
end if

// Syntaxe JavaScript
var startH = 7;

if (Math.abs(_mouse.mouseH - startH) > 10) {
  var glissementSuffisant = true;
}
```

Voir aussi

`locH`, `locV`, `Souris`, `mouseLoc`, `mouseV`

mouseItem

Utilisation

```
-- Syntaxe Lingo
_mouse.mouseItem

// Syntaxe JavaScript
_mouse.mouseItem;
```

Description

Propriété de souris ; lorsque appelée et que le curseur se trouve au-dessus d'une image-objet champ, renvoie le numéro de l'élément situé sous le curseur. En lecture seule.

Un élément est toute séquence de caractères délimitée par le séparateur courant défini dans la propriété `the itemDelimiter`. Le compte démarre au début du champ. Si la souris ne se trouve pas sur un champ, le résultat est -1.

La valeur de la propriété `mouseItem` peut changer dans un gestionnaire ou une boucle. Si le gestionnaire ou la boucle utilise la valeur initiale de `mouseItem` lorsqu'il ou elle démarre, appelez cette propriété une seule fois et affectez sa valeur à une variable locale.

Exemple

L'instruction suivante détermine si le curseur se trouve au-dessus d'une image-objet champ. Si ce n'est pas le cas, elle remplace le contenu de l'acteur champ Instructions par Veuillez pointer sur un élément :

```
-- Syntaxe Lingo
if (mouse.mouseItem = -1) then
  member("Instructions").text = "Veuillez pointer sur un élément."
end if

// Syntaxe JavaScript
if (_mouse.mouseItem == -1) {
  member("Instructions").text = "Veuillez pointer sur un élément.";
}
```

L'instruction suivante affecte à la variable `élémentCourant` l'élément situé en-dessous du curseur dans le champ spécifié :

```
-- Syntaxe Lingo
élémentCourant = member(_mouse.mouseMember).item[_mouse.mouseItem]

// Syntaxe JavaScript
var élémentCourant = member(_mouse.mouseMember).getProp("item",
  _mouse.mouseItem);
```

Voir aussi

itemDelimiter, Souris, mouseChar, mouseLine, mouseWord

mouseLevel

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.mouseLevel

// Syntaxe JavaScript
réfObjImageObjet.mouseLevel;
```

Description

Propriété d'image-objet QuickTime ; contrôle la manière dont Director passe les clics de la souris sur une image-objet QuickTime à QuickTime. La possibilité de passer les clics de la souris sur le rectangle de délimitation de l'image-objet peut se révéler pratique avec les médias interactifs comme QuickTime VR. La propriété d'image-objet `mouseLevel` peut avoir les valeurs suivantes :

- `#controller` – Ne passe que les clics de la souris sur le contrôleur de l'animation à QuickTime. Director ne répond qu'aux clics de la souris qui se produisent à l'extérieur du contrôleur. Il s'agit du comportement normal des images-objets QuickTime autres que QuickTime VR.
- `#all` – Passe tous les clics de la souris se produisant à l'intérieur du rectangle de délimitation de l'image-objet à QuickTime. Aucun clic n'est passé aux autres gestionnaires.
- `#none` – Ne passe aucun clic de la souris à QuickTime. Director répond à tous les clics de la souris.
- `#shared` – Passe tous les clics de la souris sur le rectangle de délimitation d'une image-objet QuickTime VR à QuickTime, puis passe ces événements aux gestionnaires. Il s'agit de la valeur par défaut pour QuickTime VR.

Cette propriété peut être testée et définie.

Exemple

Le script d'image suivant vérifie si le nom de l'image-objet QuickTime de la piste 5 contient la chaîne QTVR. Le cas échéant, le script donne à `mouseLevel` la valeur `#all`. Sinon, il donne à `mouseLevel` la valeur `#none`.

```
-- Syntaxe Lingo
on prepareFrame
  if sprite(5).member.name contains "QTVR" then
    sprite(5).mouseLevel = #all
  else
    sprite(5).mouseLevel = #none
  end if
end

// Syntaxe JavaScript
function prepareFrame() {
  var nm = sprite(5).member.name;
  var nmStr = nm.indexOf("QTVR");
  if (nmStr != -1) {
    sprite(5).mouseLevel = symbol("all");
  } else {
    sprite(5).mouseLevel = symbol("none");
  }
}
```

mouseLine

Utilisation

```
-- Syntaxe Lingo
_mouse.mouseLine

// Syntaxe JavaScript
_mouse.mouseLine;
```

Description

Propriété de souris ; lorsque appelée et que le curseur se trouve au-dessus d'une image-objet champ, renvoie le numéro de la ligne située sous le curseur. En lecture seule.

Le compte commence au début du champ. Une ligne est définie par un retour chariot et non par un retour à la ligne automatique. Si la souris ne se trouve pas au-dessus d'une image-objet champ, le résultat est -1.

La valeur de la propriété `mouseLine` peut changer dans un gestionnaire ou une boucle. Lorsqu'un gestionnaire ou une boucle utilise cette propriété plusieurs fois, il est d'usage d'appeler cette dernière une seule fois et d'affecter sa valeur à une variable locale.

Exemple

L'instruction suivante détermine si le curseur se trouve au-dessus d'une image-objet champ et remplace le contenu de l'acteur champ Instructions par Veuillez pointer sur une ligne si ce n'est pas le cas :

```
-- Syntaxe Lingo
if (_mouse.mouseLine = -1) then
  member("Instructions").text = "Veuillez pointer sur une ligne"
end if

// Syntaxe JavaScript
if (_mouse.mouseLine == -1) {
  member("Instructions").text = "Veuillez pointer sur une ligne";
}
```

L'instruction suivante affecte à la variable `ligneCourante` le contenu de la ligne placée sous le curseur dans le champ spécifié :

```
-- Syntaxe Lingo
ligneCourante = member(_mouse.mouseMember).line[_mouse.mouseLine]

// Syntaxe JavaScript
var ligneCourante = member(_mouse.mouseMember).getProp("line",
  _mouse.mouseLine);
```

Voir aussi

Souris, mouseChar, mouseItem, mouseWord

mouseLoc

Utilisation

```
-- Syntaxe Lingo
_mouse.mouseLoc

// Syntaxe JavaScript
_mouse.mouseLoc;
```

Description

Propriété de souris ; renvoie la position actuelle de la souris sous forme d'un point(). En lecture seule.

L'emplacement du point est indiqué sous forme de deux coordonnées, d'abord l'emplacement horizontal, puis l'emplacement vertical.

Exemple

L'instruction suivante affiche la position courante de la souris.

```
-- Syntaxe Lingo
trace(_mouse.mouseLoc)

// Syntaxe JavaScript
trace(_mouse.mouseLoc);
```

Voir aussi

Souris, mouseH, mouseV

mouseMember

Utilisation

```
-- Syntaxe Lingo
_mouse.mouseMember

// Syntaxe JavaScript
_mouse.mouseMember;
```

Description

Propriété de souris ; lorsque appelée, renvoie l'acteur associé à l'image-objet qui se trouve sous le pointeur. En lecture seule.

Si le pointeur ne se trouve pas au-dessus d'une image-objet, cette propriété renvoie le résultat `VOID` (Lingo) ou `null` (syntaxe JavaScript).

Vous pouvez utiliser cette propriété pour que l'animation effectue des actions spécifiques lorsque le pointeur survole une image-objet et que celle-ci utilise un acteur particulier.

La valeur de la propriété `mouseMember` peut changer fréquemment. Pour utiliser cette propriété plusieurs fois dans un gestionnaire avec une valeur constante, placez la valeur de `mouseMember` dans une variable locale lorsque le gestionnaire démarre et utilisez cette variable à la place.

Exemple

L'instruction suivante vérifie si l'acteur Hors limites est l'acteur associé à l'image-objet qui se trouve sous le curseur et affiche un message d'alerte si c'est le cas. Cet exemple illustre comment vous pouvez spécifier une action en fonction de l'acteur associé à une image-objet.

```
-- Syntaxe Lingo
if (_mouse.mouseMember == member("Hors limites")) then
  _player.alert("Gardez vos distances !")
end if

// Syntaxe JavaScript
if (_mouse.mouseMember = member("Hors limites")) {
  _player.alert("Gardez vos distances !");
}
```

L'instruction suivante affecte l'acteur de l'image-objet qui se trouve sous le curseur à la variable `dernierActeur` :

```
-- Syntaxe Lingo
dernierActeur = _mouse.mouseMember

// Syntaxe JavaScript
dernierActeur = _mouse.mouseMember;
```

Voir aussi

`castLibNum`, `Souris`

mouseOverButton

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.mouseOverButton

// Syntaxe JavaScript
réfObjImageObjet.mouseOverButton;
```

Description

Propriété d'image-objet Flash ; indique si le curseur de la souris se trouve sur un bouton dans une image-objet d'animation Flash spécifiée par le paramètre *quelleImageObjetFlash* (TRUE) ou s'il se trouve en-dehors de l'image-objet ou encore à l'intérieur de l'image-objet, mais sur un élément autre qu'un bouton, comme un arrière-plan par exemple (FALSE).

Cette propriété peut être testée, mais pas définie.

Exemple

Le script d'image suivant vérifie si le curseur de la souris se trouve sur un bouton de navigation de l'animation Flash dans l'image-objet 3. Le cas échéant, le script met à jour un champ de texte en affichant un message approprié. Sinon, le script efface le message.

```
-- Syntaxe Lingo
on enterFrame
  case sprite(3).mouseOverButton of
    TRUE:
      member("message").text = "Cliquez ici pour passer à la page suivante."
    FALSE:
      member("message").text = " "
  end case
  _movie.updatestage()
end

// Syntaxe JavaScript
function enterFrame() {
  switch(sprite(3).mouseOverButton)
    cas 1 :
      member("message").text = "Cliquez ici pour passer à la page suivante.";
      break();
    cas 0 :
      member("message").text = " ";
      break();
  }
  _movie.updatestage();
}
```

mouseUp

Utilisation

```
-- Syntaxe Lingo
_mouse.mouseUp

// Syntaxe JavaScript
_mouse.mouseUp;
```

Description

Propriété de souris ; indique si le bouton de la souris est relâché (TRUE) ou enfoncé (FALSE). En lecture seule.

Exemple

Le gestionnaire suivant entraîne la lecture de l'animation jusqu'à ce que l'utilisateur clique sur la souris. La tête de lecture s'arrête lorsque l'utilisateur relâche le bouton de la souris.

```
-- Syntaxe Lingo
on exitFrame me
  if (_mouse.mouseUp) then
    _movie.go(_movie.frame)
  end if
end

// Syntaxe JavaScript
function exitFrame() {
  if (_mouse.mouseUp) {
    _movie.go(_movie.frame);
  }
}
```

L'instruction suivante fait sortir le code de la boucle de répétition ou du gestionnaire qu'il est en train d'exécuter lorsque l'utilisateur relâche le bouton de la souris :

```
-- Syntaxe Lingo
if (_mouse.mouseUp) then exit

// Syntaxe JavaScript
if (_mouse.mouseUp) {
  return;
}
```

Voir aussi

Souris, mouseDown, mouseH, mouseV

mouseUpScript

Utilisation

```
the mouseUpScript
```

Description

Propriété système ; spécifie le code Lingo exécuté lorsque l'utilisateur relâche le bouton de la souris. Le code Lingo est rédigé sous forme d'une chaîne encadrée de guillemets droits et peut être une instruction simple ou un script d'appel d'un gestionnaire.

Lorsque l'utilisateur relâche le bouton de la souris et que la propriété `mouseUpScript` est définie, Lingo exécute en premier les instructions spécifiées dans la propriété `mouseUpScript`. A moins que les instructions ne contiennent la commande `pass` autorisant la transmission du message `mouseUp` vers d'autres objets de l'animation, aucun autre gestionnaire `on mouseUp` ne sera exécuté.

Lorsque les instructions spécifiées pour la propriété `mouseUpScript` ne sont plus appropriées, désactivez-les à l'aide de l'instruction `set the mouseUpScript to empty`.

L'utilisation de la propriété `mouseUpScript` produit le même résultat que la commande `when mouseUp then` des versions précédentes de Director.

Cette propriété peut être testée et définie. La valeur par défaut est `EMPTY`.

Exemple

Lorsque l'instruction suivante est activée et que l'animation est sur pause, la lecture de l'animation reprend dès que l'utilisateur relâche le bouton de la souris :

```
the mouseUpScript = "go to the frame +1"
```

L'instruction suivante provoque l'émission d'un bip sonore à chaque fois que l'utilisateur relâche le bouton de la souris après avoir cliqué sur un endroit quelconque de la scène :

```
the mouseUpScript = "if the clickOn = 0 then beep"
```

L'instruction suivante affecte la propriété `mouseUpScript` au gestionnaire personnalisé *`monGestionnairePersonnalisé`*. Un gestionnaire personnalisé Lingo doit être encadré de guillemets droits lorsque utilisé avec la propriété `mouseUpScript`.

```
the mouseUpScript = "monGestionnairePersonnalisé"
```

Voir aussi

`stopEvent()`, `mouseDownScript`, `on mouseDown (gestionnaire d'événement)`, `on mouseUp (gestionnaire d'événement)`

mouseV

Utilisation

```
-- Syntaxe Lingo
_mouse.mouseV

// Syntaxe JavaScript
_mouse.mouseV;
```

Description

Propriété de souris ; indique la position verticale du curseur de la souris, en pixels, calculée à partir du haut de la scène. En lecture seule.

La valeur de cette propriété augmente lorsque le curseur se déplace vers le bas et diminue lorsqu'il se déplace vers le haut.

La propriété `mouseV` est pratique pour aligner les images-objets avec la position verticale du curseur de la souris et vérifier si celui-ci se trouve sur une zone de la scène. L'utilisation conjointe des propriétés `mouseH` et `mouseV` permet d'identifier la position exacte du curseur.

Exemple

Le gestionnaire suivant place l'image-objet 1 à la position du pointeur et met la scène à jour chaque fois que l'utilisateur appuie sur le bouton de la souris :

```
-- Syntaxe Lingo
on mouseDown
  sprite(1).locH = _mouse.mouseH
  sprite(1).locV = _mouse.mouseV
end

// Syntaxe JavaScript
function mouseDown() {
  sprite(1).locH = _mouse.mouseH;
  sprite(1).locV = _mouse.mouseV;
}
```

L'instruction suivante vérifie si le pointeur se trouve à plus de 10 pixels au-dessus ou en dessous d'un point de départ et, le cas échéant, donne à la variable `glissementSuffisant` la valeur `TRUE` :

```
-- Syntaxe Lingo
startV = 7

if (abs(_mouse.mouseV - startV) > 10) then
  glissementSuffisant = TRUE
end if

// Syntaxe JavaScript
var startV = 7

if (Math.abs(_mouse.mouseV - startV) > 10) {
  var glissementSuffisant = true;
}
```

Voir aussi

`locH`, `locV`, `Souris`, `mouseH`, `mouseLoc`

mouseWord

Utilisation

```
-- Syntaxe Lingo
_mouse.mouseWord

// Syntaxe JavaScript
_mouse.mouseWord;
```

Description

Propriété de souris ; lorsque appelée et que le curseur se trouve au-dessus d'une image-objet champ, renvoie le numéro du mot sur lequel se trouve le pointeur. En lecture seule.

Le compte commence au début du champ. Si la souris n'est pas dans un champ, le résultat est -1.

La valeur de la propriété `mouseWord` peut changer dans un gestionnaire ou une boucle. Lorsqu'un gestionnaire ou une boucle utilise cette propriété plusieurs fois, il est d'usage d'appeler la fonction une seule fois et d'affecter sa valeur à une variable locale.

Exemple

L'instruction suivante vérifie si le pointeur se trouve au-dessus d'une image-objet champ et, si ce n'est pas le cas, remplace le contenu de l'acteur champ Instructions par Veuillez pointer sur un mot :

```
-- Syntaxe Lingo
if (_mouse.mouseWord = -1) then
  member("Instructions").text = "Veuillez pointer sur un mot"
else
  member("Instructions").text = "Merci."
end if

// Syntaxe JavaScript
if (_mouse.mouseWord == -1) {
  member("Instructions").text = "Veuillez pointer sur un mot";
}
else {
  member("Instructions").text = "Merci.";
}
```

L'instruction suivante affecte à la variable `motCourant` le numéro du mot qui se trouve sous le pointeur dans le champ spécifié.

```
-- Syntaxe Lingo
motCourant = member(_mouse.mouseMember).word[_mouse.mouseWord]

// Syntaxe JavaScript
var motCourant = member(_mouse.mouseMember).getProp("word",
  _mouse.mouseWord);
```

Voir aussi

`Souris`, `mouseChar`, `mouseItem`

moveableSprite

Utilisation

```
sprite(quelleImageObjet).moveableSprite
the moveableSprite of sprite quelleImageObjet
```

Description

Propriété d'image-objet ; indique si une image-objet peut être déplacée par l'utilisateur (`TRUE`) ou non (`FALSE`).

Vous pouvez rendre une image-objet déplaçable en sélectionnant l'option Déplaçable dans le scénario. Toutefois, pour contrôler si une image-objet est déplaçable et permettre ou non son déplacement, utilisez Lingo. Par exemple, pour permettre à l'utilisateur de faire glisser des images-objets une par une, puis l'empêcher de les déplacer une fois qu'elles sont positionnées à l'endroit souhaité, vous pouvez activer et désactiver la propriété `moveableSprite` au moment approprié.

Remarque : Pour permettre un contrôle personnalisé tel que le retour automatique des images-objets à leur position d'origine ou leur animation pendant le glissement, créez un comportement permettant de gérer ces fonctionnalités supplémentaires.

Cette propriété peut être testée et définie.

Exemple

Le gestionnaire suivant rend les images-objets de la piste 5 déplaçables :

```
on spriteMove
   sprite(5).moveableSprite = TRUE
end
```

L'instruction suivante vérifie si une image-objet est déplaçable et affiche un message si elle ne l'est pas :

```
if sprite(13).moveableSprite = FALSE then
  member("Notice").text = "Impossible de déplacer cet élément avec la souris."
```

Voir aussi

mouseLoc

movie

Utilisation

```
-- Syntaxe Lingo
réfObjFenêtre.movie

// Syntaxe JavaScript
réfObjFenêtre.movie;
```

Description

Propriété de fenêtre ; renvoie une référence à l'objet d'animation en cours de lecture dans une fenêtre donnée. En lecture seule.

Exemple

L'instruction suivante affiche, dans la fenêtre Messages, l'objet animation en cours de lecture dans la fenêtre Empires :

```
-- Syntaxe Lingo
trace(window("Empires").movie)

// Syntaxe JavaScript
trace(window("Empires").movie);
```

Voir aussi

Fenêtre

multiSound

Utilisation

```
the multiSound
```

Description

Propriété système ; spécifie si le système supporte plusieurs pistes audio et si une carte audio multipistes est requise sur un ordinateur Windows (TRUE) ou non (FALSE).

Exemple

L'instruction suivante lit le fichier audio Musique sur la piste audio 2 si l'ordinateur supporte plus d'une piste audio :

```
if the multiSound then sound playFile 2, "Musique.wav"
```

name

Utilisation

```
-- Syntaxe Lingo
réfObjDistribution.name
réfObjActeur.name
_movie.name
réfObjFenêtre.name

// Syntaxe JavaScript
réfObjDistribution.name;
réfObjActeur.name;
_movie.name;
réfObjFenêtre.name;
```

Description

Propriété de distribution, d'acteur, d'animation et de fenêtre ; renvoie ou définit le nom d'un objet. Lecture/écriture pour les objets Distribution, Acteur et Fenêtre, lecture seule pour les objets Animation.

Exemple

L'instruction suivante donne à la fenêtre Hier le nom Aujourd'hui :

```
-- Syntaxe Lingo
window("Hier").name = "Aujourd'hui"

// Syntaxe JavaScript
window("Hier").name = "Aujourd'hui";
```

Voir aussi

Bibliothèque de distribution, Acteur, Animation, Fenêtre

name (3D)

Utilisation

```
member(quelActeur).texture(quelleTexture).name
member(quelActeur).shader(quelMatériau).name
member(quelActeur).motion(quelMouvement).name
member(quelActeur).modelResource(quelleRessourceDeModèle).name
member(quelActeur).model(quelModèle).name
member(quelActeur).light(quelleLumière).name
member(quelActeur).camera(quelleCaméra).name
member(quelActeur).group(quelGroupe).name
nœud.name
```

Description

Propriété 3D ; utilisée avec une référence d'objet, cette propriété permet d'obtenir le nom de l'objet référencé. Vous ne pouvez qu'obtenir le nom, qui ne peut pas être changé.

Tous les noms doivent être uniques. S'il est créé avec du code Lingo, le nom renvoyé est celui donné par la fonction de constructeur. S'il est créé par une application de programmation 3D, il se peut que le nom renvoyé soit le nom du modèle.

Exemple

L'instruction suivante donne à la cinquième caméra de l'acteur scèneDeTable le nom camOiseau.

```
member("scèneDeTable").camera[5].name = "camOiseau"
```

name (propriété de menu)

Utilisation

```
the name of menu(quelMenu)
the name of menu quelMenu
```

Description

Propriété de menu ; renvoie une chaîne contenant le nom du menu spécifié.

Cette propriété peut être testée, mais pas définie. Utilisez la commande `installMenu` pour créer une barre de menu personnalisée.

Remarque : Les menus ne sont pas disponibles dans Shockwave Player.

Exemple

L'instruction suivante affecte le nom du menu numéro 1 à la variable `premierMenu` :

```
premierMenu = menu(1).name
```

Le gestionnaire suivant renvoie une liste de noms de menus, à raison d'un par ligne :

```
on listeDesMenus
  laListe = []
  repeat with i = 1 to the number of menus
    laListe[i] = the name of menu i
  end repeat
  return laListe
end listeDesMenus
```

Voir aussi

number (menus), name (propriété d'élément de menu)

name (propriété d'élément de menu)

Syntaxe

```
the name of menuItem(quelElément) of menu(quelMenu)
the name of menuItem quelElément of menu quelMenu
```

Description

Propriété de menu ; détermine le texte qui apparaît dans l'élément de menu spécifié par *quelElément* du menu spécifié par *quelMenu*. L'argument *quelElément* est le nom ou le numéro de l'article de menu et *quelMenu* correspond au nom ou au numéro du menu.

Cette propriété peut être testée et définie.

Remarque : Les menus ne sont pas disponibles dans Shockwave Player.

Exemple

L'instruction suivante donne à la variable `nomDélément` le nom du huitième élément du menu Edition :

```
set nomDélément = the name of menuItem(8) of menu("Edition")
```

L'instruction suivante fait suivre la commande *Ouvrir* du menu Fichier par un nom de fichier spécifique :

```
the name of menuItem("Ouvrir") of menu("menuFichier") = "Ouvrir" && fileName
```

Voir aussi

`name (propriété de menu)`, `number (éléments de menu)`

name (image-objet)

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.name

// Syntaxe JavaScript
réfObjImageObjet.name;
```

Description

Propriété d'image-objet ; identifie le nom d'une image-objet. Lecture/écriture au cours d'une session d'enregistrement du scénario uniquement.

A la différence des propriétés d'affichage d'images-objets telles que `backColor` et `blend`, aucun script ne peut être affecté à la propriété `name` d'une image-objet. Cela signifie que la propriété `name` ne peut être définie que pendant une session d'enregistrement du scénario – entre des appels aux méthodes `beginRecording()` et `endRecording()` de l'objet Animation. Vous ne pouvez définir la propriété `name` que si `beginRecording()` est appelé sur ou avant une image du scénario qui contient l'image-objet.

Remarque : Démarrer une session d'enregistrement du scénario à l'aide de `beginRecording()` redéfinit les propriétés de toutes les images-objets ayant un script et des pistes d'images-objets.

Si vous utilisez un script pour créer une nouvelle image-objet pendant une session d'enregistrement du scénario et utilisez `updateFrame()` pour appliquer les données de l'image-objet à la session, vous ne pouvez définir le nom de l'image-objet qu'une fois que vous revenez à l'image dans laquelle l'image-objet a été créée. Utilisez une méthode telle que `go()` pour revenir à une image spécifique.

Exemple

L'instruction suivante donne à l'image-objet 5 le nom Fond Sonore :

```
-- Syntaxe Lingo
sprite(5).name = "Fond Sonore"

// Syntaxe JavaScript
sprite(5).name = "Fond Sonore";
```

Voir aussi

`beginRecording()`, `endRecording()`, `go()`, `Image-objet`, `updateFrame()`

name (piste d'image-objet)

Utilisation

```
-- Syntaxe Lingo
réfObjPisteDimageObjet.name

// Syntaxe JavaScript
réfObjPisteDimageObjet.name;
```

Description

Propriété de piste d'image-objet ; identifie le nom d'une piste d'image-objet. Lecture/écriture au cours d'une session d'enregistrement du scénario uniquement.

Définit la propriété `name` d'une piste d'image-objet pendant une session d'enregistrement du scénario – entre des appels aux méthodes `beginRecording()` et `endRecording()` de l'objet Animation.

Remarque : Démarrer une session d'enregistrement du scénario à l'aide de `beginRecording()` redéfinit les propriétés de toutes les images-objets ayant un script et des pistes d'images-objets.

A la différence de la propriété `name` d'un objet Image-objet, qui ne peut être définie que sur ou après une image contenant une image-objet apparaissant dans le scénario, la propriété `name` d'un objet Piste d'image-objet peut être définie sur une piste vide. Cela signifie que vous n'avez pas à appeler `updateFrame()` avant de définir la propriété `name` de la piste d'image-objet.

Tout changement apporté à l'aide d'un script au nom d'une piste d'image-objet n'est pas reflété dans la fenêtre Scénario.

Exemple

L'instruction suivante donne à la piste d'image-objet 6 le nom Ficelle de cerf-volant pendant une session d'enregistrement du scénario :

```
-- Syntaxe Lingo
on mouseDown
  _movie.beginRecording()
  channel(6).name = "Ficelle de cerf-volant"
  _movie.endRecording()
end

// Syntaxe JavaScript
function mouseDown() {
  _movie.beginRecording();
  channel(6).name = "Ficelle de cerf-volant";
  _movie.endRecording();
}
```

Voir aussi

`beginRecording()`, `endRecording()`, `Piste d'image-objet`

name (temporisation)

Utilisation

objetDeTemporisation.name

Description

Cette propriété de temporisation est le nom de l'objet de temporisation tel qu'il a été défini lors de la création de l'objet. La commande `new()` permet de créer des objets de temporisation.

Exemple

Le gestionnaire de temporisation suivant affiche une alerte contenant le nom de la temporisation qui a transmis l'événement :

```
on gestionDeTemporisation objetDeTemporisation
  alert "Délai dépassé :" && objetDeTemporisation.name
end
```

Voir aussi

forget() (temporisation), new(), period, persistent, target, time (objet de temporisation), timeout(), timeoutHandler, timeoutList

name (XML)

Utilisation

nœudXML.name

Description

Propriété XML ; renvoie le nom du nœud XML spécifié.

Exemple

Avec le code XML suivant :

```
<?xml version="1.0"?>
  <e1>
    <nomDeBalise attr1="val1" attr2="val2"/>
    <e2>élém 2</e2>
    <e3>élém 3</e3>
  </e1>
```

L'instruction Lingo suivante renvoie le nom de la seconde balise imbriquée dans la balise `<e1>` :

```
put gObjetDanalyse.child[1].child[2].name
-- "e2"
```

Voir aussi

attributeName

near (brouillard)

Utilisation

```
member(quelActeur).camera(quelleCaméra).fog.near
référenceDeCaméra.fog.near
member(quelActeur).camera(quelleCaméra).fog.near
référenceDeCaméra.fog.near
```

Description

Propriété 3D ; permet d'obtenir ou de définir la distance séparant l'avant de la caméra du point où le brouillard commence si `fog.enabled` a pour valeur `TRUE`.

La valeur par défaut de cette propriété est 0.0.

Exemple

L'instruction suivante donne à la propriété `near` du brouillard de la caméra vueParDéfaut la valeur 100. Si la propriété `enabled` du brouillard a pour valeur `TRUE` et que sa propriété `decayMode` a pour valeur `#linear`, le brouillard apparaîtra à 100 unités d'univers devant la caméra.

```
member("Univers 3D").camera("vueParDéfaut").fog.near = 100.0
```

Voir aussi

`fog`, `far (brouillard)`, `enabled (brouillard)`, `decayMode`

nearFiltering

Utilisation

```
member(quelActeur).texture(quelleTexture).nearFiltering
member(quelActeur).shader(quelMatériau).\
  texture(quelleTexture).nearFiltering
member(quelActeur).model(quelModèle).shader.texture\
  (quelleTexture).nearFiltering
member(quelActeur).model(quelModèle).shaderList\
  [indexDeListeDeMatériaux].texture(quelleTexture).nearFiltering
```

Description

Propriété 3D de texture ; permet de savoir ou de définir si le filtrage bilinéaire est utilisé pour le rendu d'une texture projetée qui couvre une plus grande partie de l'écran que l'original. Le filtrage bilinéaire estompe les erreurs de texture pour en améliorer l'apparence. Le filtrage bilinéaire estompe les erreurs en deux dimensions. Le filtrage trilinéaire estompe les erreurs en trois dimensions. Le filtrage améliore l'apparence au détriment des performances, le filtrage bilinéaire étant toutefois plus rapide que le filtrage trilinéaire.

Lorsque la propriété a pour valeur `TRUE`, le filtrage bilinéaire est utilisé. Lorsque la propriété a pour valeur `FALSE`, le filtrage bilinéaire n'est pas utilisé. La valeur par défaut est `TRUE`.

Exemple

L'instruction suivante désactive le filtrage bilinéaire pour la texture `gbTexture` dans l'acteur Séquence :

```
member("Séquence").texture("gbTexture").nearFiltering = FALSE
```

netPresent

Utilisation

```
-- Syntaxe Lingo
_player.netPresent

// Syntaxe JavaScript
_player.netPresent;
```

Description

Propriété de lecteur ; détermine si les Xtras requis pour accéder à Internet sont disponibles, mais n'indique pas si une connexion à Internet est actuellement active. En lecture seule.

Si les Xtras réseau ne sont pas disponibles, `netPresent` fonctionnera correctement, mais son utilisation provoquera une erreur de script.

Exemple

L'instruction suivante envoie un message d'alerte si les Xtras ne sont pas disponibles :

```
-- Syntaxe Lingo
if (not(_player.netPresent)) then
  _player.alert("Désolé, les Xtras réseau sont introuvables.")
end if

// Syntaxe JavaScript
if (!(_player.netPresent)) {
  _player.alert("Désolé, les Xtras réseau sont introuvables.");
}
```

Voir aussi

Lecteur

netThrottleTicks

Utilisation

```
-- Syntaxe Lingo
_player.netThrottleTicks

// Syntaxe JavaScript
_player.netThrottleTicks;
```

Description

Propriété de lecteur ; dans un environnement auteur Macintosh, permet de contrôler la fréquence de service d'une opération réseau. Lecture/écriture.

La valeur par défaut est de 15. Plus cette valeur est élevée, plus la lecture de l'animation et des effets animés est régulière, mais le temps passé à desservir les opérations réseau est diminué. Une valeur plus faible permet de passer plus de temps à desservir les opérations réseau, mais affecte négativement les performances de lecture des animations et des effets animés.

Cette propriété n'affecte l'environnement auteur et les projections que sur Macintosh. Elle n'a aucun effet sous Windows ou dans Shockwave Player sur Macintosh.

Voir aussi

Lecteur

node

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.node

// Syntaxe JavaScript
réfObjImageObjet.node;
```

Description

Propriété d'image-objet QuickTime VR ; renvoie l'identifiant de nœud affiché par l'image-objet.

Cette propriété peut être testée et définie.

nodeEnterCallback

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.nodeEnterCallback

// Syntaxe JavaScript
réfObjImageObjet.nodeEnterCallback;
```

Description

Propriété d'image-objet QuickTime VR ; contient le nom du gestionnaire exécuté lorsque l'animation QuickTime VR est passée à un nouveau nœud actif de la scène. Ce message a deux arguments : le paramètre `me` et l'identifiant du nœud affiché.

L'image-objet QuickTime VR reçoit le message en premier.

Pour annuler l'appel, donnez à cette propriété une valeur de 0.

Pour des performances optimales, ne définissez de propriété d'appel que lorsque absolument nécessaire.

Cette propriété peut être testée et définie.

nodeExitCallback

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.nodeExitCallback

// Syntaxe JavaScript
réfObjImageObjet.nodeExitCallback;
```

Description

Propriété d'image-objet QuickTime VR ; contient le nom du gestionnaire exécuté lorsque l'animation QuickTime VR est sur le point de passer à un nouveau nœud actif de la scène. Le message contient trois arguments : le paramètre `me`, l'identifiant du nœud que l'animation est sur le point de quitter et celui du nœud auquel elle s'apprête à passer.

La valeur renvoyée par le gestionnaire détermine si l'animation passe ou non au nœud suivant. Si le gestionnaire renvoie `#continue`, l'image-objet QuickTime VR passe normalement au nœud suivant. Par contre, s'il renvoie `#cancel`, la transition n'a pas lieu et l'animation reste sur le nœud d'origine.

Cette propriété doit être réglée sur 0 pour effacer l'instruction d'appel.

L'image-objet QuickTime VR reçoit le message en premier.

Pour des performances optimales, ne définissez de propriété d'appel que lorsque absolument nécessaire.

Cette propriété peut être testée et définie.

nodeType

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.nodeType

// Syntaxe JavaScript
réfObjImageObjet.nodeType;
```

Description

Propriété d'image-objet QuickTime VR ; renvoie le type du nœud de l'image-objet spécifiée actuellement sur la scène. Les valeurs possibles sont `#object`, `#panorama` ou `#unknown`. (`#unknown` correspond à la valeur d'une image-objet qui n'est pas une image-objet QuickTime VR).

Cette propriété peut être testée, mais pas définie.

normalList

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  normalList
model.meshDeform.mesh[index].normalList
```

Description

Propriété 3D ; utilisée avec une ressource de modèle de type `#mesh`, cette propriété permet d'obtenir ou de définir la propriété `normalList` de la ressource de modèle.

La propriété `normalList` est une liste de vecteurs linéaire à partir de laquelle vous pouvez spécifier des normales de sommets pour la construction des faces de votre maille.

Cette propriété doit être définie à l'aide d'une liste comprenant exactement le même nombre de vecteurs que dans l'appel `newMesh()`.

Autrement, la propriété `normalList` peut être générée par la méthode `generateNormals()` des ressources de modèle de maille.

De même, dans le cas du modificateur `meshDeform`, la propriété `normalList` est une liste de vecteurs linéaire à partir de laquelle vous pouvez spécifier les normales des sommets pour déformer votre maille.

Pour plus d'informations sur les normales de faces et de sommets, consultez `normals`.

Exemple

```
put member(5,2).modelResource("carré de maille").normalList
-- [vector(0,0,1)]
member(2).modelResource("maille3").normalList[2] = vector\
  (205.0000, -300.0000, 27.0000)
```

Voir aussi

face, meshDeform (modificateur)

normals

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  face[index].normals
```

Utilisation

Propriété 3D de face ; pour les ressources de modèle de type #mesh (créées à l'aide de la commande newMesh), cette propriété permet d'obtenir et de définir la liste des vecteurs de normales utilisés par la face spécifiée par le paramètre *index*.

Définissez cette propriété à l'aide d'une liste linéaire de nombres entiers correspondant à la position d'index de chaque normale de sommet dans la propriété normalList de la ressource de modèle.

Cette propriété doit être définie à la même longueur que la liste face[index].vertices ou peut être une liste vide [ ].

Ne définissez aucune valeur pour cette propriété si vous prévoyez de générer les vecteurs de normales à l'aide de la commande generateNormals().

Si vous apportez des modifications à cette propriété ou utilisez la commande generateNormals(), vous devrez appeler la commande build() pour reconstruire la maille.

Exemple

L'instruction suivante définit la propriété normals de la cinquième face de la ressource de modèle Lecteur à l'aide d'une liste de nombres entiers.

```
member("3D").modelResource("Lecteur").face[5].normals = [2,32,14]
```

Voir aussi

face, normalList, vertices

number (distribution)

Utilisation

```
-- Syntaxe Lingo
réfObjDistribution.number

// Syntaxe JavaScript
réfObjDistribution.number;
```

Description

Propriété de bibliothèque de distribution ; renvoie le numéro d'une bibliothèque de distribution donnée. En lecture seule.

Exemple

La boucle suivante utilise la fenêtre Messages pour afficher le nombre d'acteurs contenus dans chaque distribution de l'animation :

```
-- Syntaxe Lingo
repeat with n = 1 to _movie.castLib.count
  put(castLib(n).name && "contient" && castLib(n).member.count \
  && "acteurs.")
end repeat

// Syntaxe JavaScript
for (var n=1; n<=_movie.castLib.count; n++) {
  put(castLib(n).name + " contient " + castLib(n).member.count
  + " acteurs.")
}
```

Voir aussi

Bibliothèque de distribution

number (caractères)

Utilisation

`the number of chars in` *`expressionSousChaîne`*

Description

Expression de sous-chaîne ; renvoie le nombre total de caractères d'une expression de sous-chaîne.

Une expression de sous-chaîne peut être un caractère (y compris des espaces et des caractères de contrôle tels que Tab et Retour), un mot, un élément ou la ligne d'un conteneur de caractères. Les conteneurs peuvent être des acteurs champ et des variables contenant des chaînes, ainsi que des caractères, des mots, des éléments, des lignes et des plages spécifiés dans des conteneurs.

Remarque : La fonction `count()` est plus pratique pour déterminer le nombre de caractères d'une sous-chaîne.

Exemple

L'instruction suivante affiche le nombre de caractères de la chaîne Macromedia, la société multimédia dans la fenêtre Messages :

```
put the number of chars in "Macromedia, la société multimédia"
```

Le résultat est 33.

L'instruction suivante donne à la variable `compteurDeCaractères` le nombre de caractères du mot i de la chaîne Noms :

```
compteurDeCaractères = the number of chars in member("Noms").word[i]
```

Vous pouvez obtenir les mêmes résultats avec les acteurs texte en utilisant la syntaxe suivante :

```
compteurDeCaractères = member("Noms").word[i].char.count
```

Voir aussi

length(), char...of, count(), number (éléments), number (lignes), number (mots)

number (éléments)

Utilisation

```
the number of items in expressionSousChaîne
```

Description

Sous-chaîne ; renvoie le nombre total d'éléments d'une expression de sous-chaîne. Un élément de sous-chaîne est une série de caractères délimitée par des virgules.

Une expression de sous-chaîne peut être un caractère, un mot, un élément ou la ligne d'un conteneur de caractères. Les conteneurs peuvent être des acteurs champ et des variables contenant des chaînes, ainsi que des caractères, des mots, des éléments, des lignes et des plages spécifiés dans des conteneurs.

Remarque : La fonction `count()` est plus pratique pour déterminer le nombre d'éléments d'une expression de sous-chaîne.

Exemple

L'instruction suivante affiche le nombre d'éléments de la chaîne Macromedia, la société multimédia dans la fenêtre Messages :

```
put the number of items in "Macromedia, la société multimédia"
```

Le résultat est 2.

L'instruction suivante donne à la variable `compteurDéléments` le nombre d'éléments du champ Noms :

```
compteurDéléments = the number of items in member("Noms").text
```

Vous pouvez obtenir les mêmes résultats avec les acteurs texte en utilisant la syntaxe suivante :

```
compteurDéléments = member("Noms").item.count
```

Voir aussi

`item...of`, `count()`, `number (caractères)`, `number (lignes)`, `number (mots)`

number (lignes)

Utilisation

```
the number of lines in expressionSousChaîne
```

Description

Sous-chaîne ; renvoie le nombre total de lignes d'une expression de sous-chaîne. Les lignes sont délimitées par des retours de chariot et non par des retours à la ligne automatiques.

Une expression de sous-chaîne peut être un caractère, un mot, un élément ou la ligne d'un conteneur de caractères. Les conteneurs peuvent être des acteurs champ et des variables contenant des chaînes, ainsi que des caractères, des mots, des éléments, des lignes et des plages spécifiés dans des conteneurs.

Remarque : La fonction `count()` est plus pratique pour déterminer le nombre de lignes d'une sous-chaîne.

Exemple

L'instruction suivante affiche le nombre de lignes contenues dans la chaîne Macromedia, la société multimédia dans la fenêtre Messages :

```
put the number of lines in "Macromedia, la société multimédia"
```

Le résultat est 1.

L'instruction suivante associe la variable `compteurDeLignes` au nombre de lignes du champ Noms :

```
compteurDeLignes = the number of lines in member("Noms").text
```

Vous pouvez obtenir les mêmes résultats avec les acteurs texte en utilisant la syntaxe suivante :

```
compteurDeLignes = member("Noms").line.count
```

Voir aussi

`line...of`, `count()`, `number (caractères)`, `number (éléments)`, `number (mots)`

number (acteur)

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.number

// Syntaxe JavaScript
réfObjActeur.number;
```

Description

Propriété d'acteur ; renvoie le numéro de la bibliothèque de distribution d'un acteur donné. En lecture seule.

La valeur de cette propriété est un identifiant unique de l'acteur. C'est un nombre entier décrivant l'emplacement et la position de l'acteur dans la bibliothèque de distribution.

Exemple

L'instruction suivante donne le numéro de l'acteur Interrupteur à la variable *`quelActeur`* :

```
-- Syntaxe Lingo
quelActeur = member("Interrupteur").number

// Syntaxe JavaScript
var quelActeur = member("Interrupteur").number;
```

L'instruction suivante affecte l'acteur Ballon rouge à l'image-objet 1 :

```
-- Syntaxe Lingo
sprite(1).member = member("Ballon rouge").number

// Syntaxe JavaScript
sprite(1).member = member("Ballon rouge").number;
```

L'instruction qui suit vérifie qu'un acteur existe réellement avant d'essayer de le remplacer dans l'image-objet :

```
-- Syntaxe Lingo
property spriteNum

on mouseUp me
  if (member("Visage").number > 0) then
    sprite(spriteNum).member = "Visage"
  end if
end

// Syntaxe JavaScript
function mouseUp() {
  if (member("Visage").number > 0) {
    sprite(this.spriteNum).member = "Visage";
  }
}
```

Voir aussi

`castLib()`, `Acteur`

number (menus)

Utilisation

```
the number of menus
```

Description

Propriété de menu ; indique le nombre de menus de l'animation courante.

Cette propriété peut être testée, mais non définie. Utilisez la commande `installMenu` pour créer une barre de menu personnalisée.

Remarque : Les menus ne sont pas disponibles dans Shockwave Player.

Exemple

L'instruction suivante détermine si l'animation contient des menus personnalisés et, si ce n'est pas le cas, installe le menu barreDeMenus :

```
if the number of menus = 0 then installMenu "barreDeMenus"
```

L'instruction suivante affiche le nombre de menus de l'animation courante dans la fenêtre Messages :

```
put the number of menus
```

Voir aussi

`installMenu`, `number (éléments de menu)`

number (éléments de menu)

Utilisation

```
the number of menuItems of menu quelMenu
```

Description

Propriété de menu ; indique le nombre d'éléments du menu personnalisé spécifié par *quelMenu*. Le paramètre *quelMenu* peut être un nom ou un numéro de menu.

Cette propriété peut être testée, mais non définie. Utilisez la commande `installMenu` pour créer une barre de menu personnalisée.

Remarque : Les menus ne sont pas disponibles dans Shockwave Player.

Exemple

L'instruction suivante donne à la variable `élémentsDeFichier` le nombre d'éléments du menu personnalisé Fichier :

```
élémentsDeFichier = the number of menuItems of menu "Fichier"
```

L'instruction suivante donne à la variable `compteDesEléments` le nombre d'éléments du menu personnalisé dont le numéro est égal à la variable `i` :

```
compteDesEléments = the number of menuItems of menu i
```

Voir aussi

`installMenu`, `number (menus)`

number (piste d'image-objet)

Utilisation

```
-- Syntaxe Lingo
réfObjPisteDimageObjet.number

// Syntaxe JavaScript
réfObjPisteDimageObjet.number;
```

Description

Propriété de piste d'image-objet ; renvoie le numéro d'une piste d'image-objet. En lecture seule.

Exemple

L'instruction affiche, dans la fenêtre Messages, le numéro d'une piste d'image-objet nommée :

```
-- Syntaxe Lingo
put(channel("Ficelle de cerf-volant").number)

// Syntaxe JavaScript
put(channel("Ficelle de cerf-volant").number);
```

Voir aussi

`Image-objet`

number (système)

Utilisation

```
the number of castLibs
```

Description

Propriété système ; renvoie le nombre de distributions de l'animation courante.

Cette propriété peut être testée, mais pas définie.

Exemple

La boucle suivante utilise la fenêtre Messages pour afficher le nombre d'acteurs contenus dans chaque distribution de l'animation :

```
repeat with n = 1 to the number of castLibs
  put castLib(n).name && "contient" && the number of \
  members of castLib(n) && "acteurs."
end repeat
```

number (mots)

Utilisation

```
the number of words in expressionSousChaîne
```

Description

Expression de sous-chaîne ; renvoie le nombre de mots dans l'expression de sous-chaîne spécifiée par *expressionSousChaîne*.

Une expression de sous-chaîne peut être un caractère, un mot, un élément ou une ligne d'un conteneur de caractères. Les conteneurs peuvent être des acteurs champ et des variables contenant des chaînes, ainsi que des caractères, des mots, des éléments, des lignes et des plages spécifiés dans des conteneurs.

Pour compter le nombre de mots contenus dans les acteurs texte, consultez `count`.

Remarque : La fonction `count()` est plus pratique pour déterminer le nombre de mots d'une sous-chaîne.

Exemple

L'instruction suivante affiche le nombre de mots de la chaîne Macromedia, la société multimédia dans la fenêtre Messages :

```
put the number of words in "Macromedia, la société multimédia"
```

Le résultat est 4.

Le gestionnaire suivant inverse l'ordre des mots spécifiés par l'argument `listeDeMots` :

```
on reverse listeDeMots
  laListe = EMPTY
  repeat with i = 1 to the number of words in listeDeMots
    put word i of listeDeMots & " " before laListe
  end repeat
  delete laListe.char[laListe.char.count]
  return laListe
end
```

Voir aussi

`count()`, `number (caractères)`, `number (éléments)`, `number (lignes)`, `word...of`

number of members

Utilisation

```
the number of members of castLib quelleDistribution
```

Description

Propriété d'acteur ; indique le numéro du dernier acteur de la distribution spécifiée.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante affiche le type de chaque acteur de la distribution Distribution Centrale dans la fenêtre Messages. La propriété `number of members of castLib` est utilisée pour déterminer le nombre de répétitions de la boucle.

```
repeat with i = 1 to the number of members of castLib("Distribution Centrale")
    put "L'acteur" && i && "est un" && member(i, "Distribution Centrale").type
  end repeat
```

number of xtras

Utilisation

```
the number of xtras
```

Description

Propriété système ; renvoie le nombre d'Xtras de programmation disponibles pour l'animation. Ces Xtras peuvent avoir été ouverts avec la commande `openxlib` ou se trouver dans le dossier Configuration\Xtras.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante affiche le nombre d'Xtras de programmation disponibles pour l'animation dans la fenêtre Messages :

```
put the number of xtras
```

numChannels

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.numChannels

// Syntaxe JavaScript
réfObjActeur.numChannels;
```

Description

Propriété d'acteur Shockwave Audio (SWA) ; renvoie le nombre de pistes de l'acteur SWA spécifié lu en flux continu. Les valeurs sont 1 pour mono ou 2 pour stéréo.

Cette propriété n'est disponible qu'après le début de la lecture en flux continu de l'acteur SWA ou après le préchargement du fichier avec la commande `preLoadBuffer`.

Cette propriété peut être testée, mais pas définie.

Exemple

Dans l'exemple suivant, le nombre de pistes audio de l'acteur SWA transféré Duke Ellington est affecté à l'acteur champ Affichage des pistes :

```
-- Syntaxe Lingo
maVariable = member("Duke Ellington").numChannels
  if maVariable = 1 then
    member("Affichage des pistes").text = "Mono"
  else
    member("Affichage des pistes").text = "Stéréo"
end if

// Syntaxe JavaScript
var maVariable = member("Duke Ellington").numChannels;
if (maVariable == 1) {
  member("Affichage des pistes").text = "Mono";
} else {
  member("Affichage des pistes").text = "Stéréo";
}
```

numParticles

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  emitter.numParticles
référenceDobjetDeRessourceDeModèle.emitter.numParticles
```

Description

Propriété 3D ; utilisée avec une ressource de modèle de type `#particle`, cette propriété permet d'obtenir ou de définir la propriété `numParticles` de l'émetteur de particules de la ressource. La valeur doit être supérieure à 0 et inférieure ou égale à 100 000, la valeur par défaut étant 1000.

Exemple

Dans l'exemple suivant, systèmeThermique est une ressource de modèle de type #particle. L'instruction suivante donne au nombre de particules de systèmeThermique la valeur 50 000.

```
member("Chauffage").modelResource("systèmeThermique").emitter.\
  numParticles = 50000
```

Voir aussi

emitter

numSegments

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  numSegments
```

Description

Propriété 3D ; utilisée avec une ressource de modèle de type #cylinder, cette propriété permet d'obtenir ou de définir la propriété numSegments de la ressource de modèle.

La propriété numSegments détermine le nombre de segments entre l'extrémité supérieure et l'extrémité inférieure du cylindre. Cette propriété doit être supérieure ou égale à la valeur par défaut de 2.

Le lissé de la surface du cylindre dépend de la valeur spécifiée pour cette propriété. Plus la valeur de la propriété est élevée, plus la surface du cylindre apparaît lisse.

Exemple

L'instruction suivante donne à la propriété numSegments de la ressource de modèle Cylindre03 la valeur 10.

```
member("Univers 3D").modelResource("Cylindre03").numSegments = 10
```

obeyScoreRotation

Utilisation

```
member(acteurFlash).obeyScoreRotation
```

Description

Propriété d'acteur Flash ; définie comme TRUE ou FALSE pour déterminer si l'image-objet animation Flash utilise les informations de rotation du scénario ou l'ancienne propriété de rotation des éléments Flash.

Cette propriété est automatiquement définie comme FALSE pour toutes les animations créées par des versions de Director antérieures à 7 afin de conserver l'ancienne fonctionnalité consistant à utiliser la propriété de rotation d'acteurs pour toutes les images-objets contenant cet acteur Flash.

La propriété des nouveaux éléments créés par la version 7 ou ultérieure est automatiquement définie comme TRUE.

Si définie comme TRUE, la propriété de rotation de l'acteur est ignorée et ce sont les paramètres de rotation du scénario qui sont utilisés.

Exemple

Le script d'image-objet suivant définit la propriété obeyScoreRotation de l'acteur Dalmatien sur 1 (TRUE) et fait ensuite pivoter de 180° l'image-objet qui contient l'acteur.

```
on mouseUp me
  member("dalmatien").obeyScoreRotation = 1
  sprite(1).rotation = sprite(1).rotation + 180
end
```

Cette propriété peut être testée et définie.

Voir aussi

rotation

optionDown

Utilisation

```
-- Syntaxe Lingo
_key.optionDown

// Syntaxe JavaScript
_key.optionDown;
```

Description

Propriété de touche ; détermine si l'utilisateur appuie sur la touche Alt (Windows) ou Option (Macintosh). En lecture seule.

Cette propriété renvoie TRUE si l'utilisateur appuie sur la touche Alt ou Option. Sinon, elle renvoie FALSE.

Sous Windows, optionDown ne fonctionne pas dans les projections si la touche Alt est enfoncée sans autre touche normale (autre qu'une touche de modification). Evitez d'utiliser optionDown si vous prévoyez de diffuser une animation sous forme de projection Windows et que vous avez besoin de détecter uniquement l'enfoncement de la touche de modification ; utilisez controlDown ou shiftDown à la place.

Sur le Macintosh, l'enfoncement de la touche Option change la valeur de key ; utilisez donc keyCode à la place.

Exemple

Le gestionnaire suivant vérifie si la touche Alt ou Option est enfoncée et, le cas échéant, appelle le gestionnaire toucheDoption :

```
-- Syntaxe Lingo
on keyDown
  if (_key.optionDown) then
    toucheDoption(_key.key)
  end if
end

// Syntaxe JavaScript
function keyDown() {
  if (_key.optionDown) {
    toucheDoption(_key.key);
```

```
    }
}
```

Voir aussi

controlDown, Touche, key, keyCode, shiftDown

organizationName

Utilisation

```
-- Syntaxe Lingo
_player.organizationName

// Syntaxe JavaScript
_player.organizationName;
```

Description

Propriété de lecteur ; contient le nom de société entré lors de l'installation de Director. En lecture seule.

Cette propriété est uniquement disponible dans l'environnement de programmation. Elle peut s'utiliser dans une animation dans une fenêtre personnalisée pour afficher des informations utilisateur.

Exemple

Le gestionnaire suivant serait normalement placé dans un script d'animation d'une animation MIAW. Il place le nom de l'utilisateur et le numéro de série dans une zone d'affichage dès l'ouverture de la fenêtre :

```
-- Syntaxe Lingo
on prepareMovie
  chaîneAffichée = _player.userName & RETURN & _player.organizationName \
  & RETURN & _player.serialNumber
  member("Infos utilisateur").text = chaîneAffichée
end

// Syntaxe JavaScript
function prepareMovie() {
  var chaîneAffichée = _player.userName + "\n" + _player.organizationName
  + "\n" + _player.serialNumber;
  member("Infos utilisateur").text = chaîneAffichée;
}
```

Voir aussi

Lecteur

originalFont

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.originalFont

// Syntaxe JavaScript
réfObjActeur.originalFont;
```

Description

Propriété d'acteur police ; renvoie le nom exact de la police d'origine importée lors de la création de l'acteur concerné.

Exemple

L'instruction suivante affiche le nom de la police importée lors de la création de l'acteur 11 :

```
-- Syntaxe Lingo
put(member(11).originalFont)

// Syntaxe JavaScript
put(member(11).originalFont);
```

Voir aussi

recordFont, bitmapSizes, characterSet

originH

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.originH

// Syntaxe JavaScript
réfObjActeurOuImageObjet.originH;
```

Description

Propriété d'acteur et d'image-objet ; contrôle la coordonnée horizontale du point d'origine d'une animation Flash ou d'une forme vectorielle, en pixels. La valeur peut être exprimée sous forme de nombre à virgule flottante.

Le point d'origine est la coordonnée d'une animation Flash ou d'une forme vectorielle autour de laquelle la mise à l'échelle et la rotation se produisent. Le point d'origine peut être défini avec une précision à virgule flottante au moyen des propriétés séparées `originH` et `originV` ou avec une précision en nombre entier au moyen de la propriété unique `originPoint`.

Vous ne pouvez définir la propriété `originH` que si la propriété `originMode` a pour valeur `#point`.

Cette propriété peut être testée et définie. La valeur par défaut est 0.

Remarque : Cette propriété doit avoir la valeur par défaut si la propriété `scaleMode` a pour valeur `#autoSize`. Sinon, l'image-objet n'est pas correctement affichée.

Exemple

Le script d'image-objet suivant utilise la propriété `originMode` pour définir une image-objet animation Flash de sorte que son point d'origine soit un point spécifique. Il définit ensuite les points d'origine horizontal et vertical.

```
-- Syntaxe Lingo
property spriteNum

on beginSprite me
  sprite(spriteNum).originMode = #point
  sprite(spriteNum).originH = 100
  sprite(spriteNum).originV = 80
end

// Syntaxe JavaScript
function beginSprite() {
  sprite(this.spriteNum).originMode = symbol("point");
  sprite(this.spriteNum).originH = 100;
  sprite(this.spriteNum).originV = 80;
}
```

Voir aussi

originV, originMode, originPoint, scaleMode

originMode

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.originMode

// Syntaxe JavaScript
réfObjActeurOuImageObjet.originMode;
```

Description

Propriété d'acteur et d'image-objet ; définit le point d'origine autour duquel la mise à l'échelle et la rotation se produisent, comme suit :

- `#center` (valeur par défaut) – Le point d'origine est au centre de l'animation Flash.
- `#topleft` – Le point d'origine est dans l'angle supérieur gauche de l'animation Flash.
- `#point` – Le point d'origine est un point spécifié par les propriétés `originPoint`, `originH` et `originV`.

Cette propriété peut être testée et définie.

Remarque : Cette propriété doit avoir la valeur par défaut si la propriété `scaleMode` a pour valeur `#autoSize`. Sinon, l'image-objet n'est pas correctement affichée.

Exemple

Le script d'image-objet suivant utilise la propriété `originMode` pour définir une image-objet animation Flash de sorte que son point d'origine soit un point spécifique. Il définit ensuite les points d'origine horizontal et vertical.

```
-- Syntaxe Lingo
property spriteNum

on beginSprite me
  sprite(spriteNum).originMode = #point
  sprite(spriteNum).originH = 100
  sprite(spriteNum).originV = 80
end

// Syntaxe JavaScript
function beginSprite() {
  sprite(this.spriteNum).originMode = symbol("point");
  sprite(this.spriteNum).originH = 100;
  sprite(this.spriteNum).originV = 80;
}
```

Voir aussi

`originH`, `originV`, `originPoint`, `scaleMode`

originPoint

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.originPoint

// Syntaxe JavaScript
réfObjActeurOuImageObjet.originPoint;
```

Description

Propriété d'acteur et d'image-objet ; contrôle le point d'origine autour duquel la mise à l'échelle et la rotation se produisent dans une animation Flash ou une forme vectorielle.

La propriété `originPoint` est spécifiée en tant que valeur d'un point de Director : par exemple, point (100,200). La définition du point d'origine d'une animation Flash ou d'une forme vectorielle au moyen de la propriété `originPoint` est équivalente à la définition individuelle des propriétés `originH` et `originV`. Ainsi, la définition de la propriété `originPoint` comme `point(50,75)` est équivalente à la définition de la propriété `originH` comme 50 et de la propriété `originV` 75.

Les valeurs de point Director spécifiées pour la propriété `originPoint` doivent être des nombres entiers, alors que `originH` et `originV` peuvent être spécifiées au moyen de nombres à virgule flottante. Lorsque vous testez la propriété `originPoint`, les valeurs du point sont tronquées pour proposer des nombres entiers. En règle générale, utilisez les propriétés `originH` et `originV` pour la précision ; utilisez la propriété `originPoint` pour la rapidité et la facilité.

Vous ne pouvez définir la propriété `originPoint` que si la propriété `originMode` est réglée sur `#point`.

Cette propriété peut être testée et définie. La valeur par défaut est 0.

Remarque : Cette propriété doit avoir la valeur par défaut si la propriété `scaleMode` a pour valeur `#autoSize`. Sinon, l'image-objet n'est pas correctement affichée.

Exemple

Le script d'image-objet suivant utilise la propriété `originMode` pour définir une image-objet animation Flash de sorte que son point d'origine soit un point spécifique. Il définit ensuite les points d'origine.

```
-- Syntaxe Lingo
property spriteNum

on beginSprite me
  sprite(spriteNum).scaleMode = #showAll
  sprite(spriteNum).originMode = #point
  sprite(spriteNum).originPoint = point(100, 80)
end

// Syntaxe JavaScript
function beginSprite() {
  sprite(this.spriteNum).scaleMode = symbol("showAll");
  sprite(this.spriteNum).originMode = symbol("point");
  sprite(this.spriteNum).originPoint = point(100, 80);
}
```

Voir aussi

`originH`, `originV`, `scaleMode`

originV

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.originv

// Syntaxe JavaScript
réfObjActeurOuImageObjet.originV;
```

Description

Propriété d'acteur et d'image-objet ; contrôle la coordonnée verticale en pixels du point d'origine d'une animation Flash ou d'une forme vectorielle, autour duquel la mise à l'échelle ou la rotation s'effectue. La valeur peut être exprimée sous forme de nombre à virgule flottante.

Le point d'origine peut être défini avec une précision à virgule flottante au moyen des propriétés séparées `originH` et `originV` ou avec une précision en nombre entier au moyen de la propriété unique `originPoint`.

Vous ne pouvez définir la propriété `originV` que si la propriété `originMode` est réglée sur `#point`.

Cette propriété peut être testée et définie. La valeur par défaut est 0.

Remarque : Cette propriété doit avoir la valeur par défaut si la propriété `scaleMode` a pour valeur `#autoSize`. Sinon, l'image-objet n'est pas correctement affichée.

Exemple

Le script d'image-objet suivant utilise la propriété `originMode` pour définir une image-objet animation Flash de sorte que son point d'origine soit un point spécifique. Il définit ensuite les points d'origine horizontal et vertical.

```
-- Syntaxe Lingo
property spriteNum

on beginSprite me
  sprite(spriteNum).scaleMode = #showAll
  sprite(spriteNum).originMode = #point
  sprite(spriteNum).originH = 100
  sprite(spriteNum).originV = 80
end

// Syntaxe JavaScript
function beginSprite() {
  sprite(this.spriteNum).scaleMode = symbol("showAll");
  sprite(this.spriteNum).originMode = symbol("point");
  sprite(this.spriteNum).originH = 100;
  sprite(this.spriteNum).originV = 80;
}
```

Voir aussi

`originH`, `originPoint`, `scaleMode`

orthoHeight

Utilisation

```
member(quelActeur).camera(quelleCaméra).orthoHeight
member(quelActeur).camera[indexDeCaméra].orthoHeight
sprite(quelleImageObjet).camera.orthoHeight
```

Description

Propriété 3D ; lorsque `camera.projection` a pour valeur `#orthographic`, la valeur `camera.orthoHeight` renvoie le nombre d'unités perpendiculaires de l'univers tenant verticalement dans l'image-objet. Les unités de l'univers sont les unités de mesure de l'univers 3D concerné. Elles sont cohérentes en interne mais choisies arbitrairement et, de ce fait, susceptibles de varier d'un univers 3D à l'autre.

Il n'est pas utile de spécifier l'index de caméra (*quelleCaméra*) pour accéder à la première caméra de l'image-objet.

La valeur par défaut de cette propriété est 200.0.

Exemple

L'instruction suivante donne à la propriété `orthoHeight` de la caméra de l'image-objet 5 la valeur 200. Cela signifie que 200 unités d'univers logeront verticalement dans l'image-objet.

```
sprite(5).camera.orthoheight = 200.0
```

Voir aussi

`projection`

overlay

Utilisation

```
member(quelActeur).camera(quelleCaméra).\
   overlay[indexDeRecouvrement].nomDePropriété
member(quelActeur).camera(quelleCaméra).overlay.count
```

Description

Propriété 3D de caméra ; permet d'obtenir et de définir l'accès aux propriétés des recouvrements contenus dans la liste des recouvrements de la caméra à afficher. Utilisée comme `overlay.count`, cette propriété renvoie le nombre total de recouvrements (contenus dans la liste des recouvrements de la caméra) à afficher.

Les recouvrements sont des textures affichées devant tous les modèles qui apparaissent dans le frustrum de vue d'une caméra donnée. Les recouvrements sont dessinés dans l'ordre dans lequel ils apparaissent dans la liste de recouvrements de la caméra ; le premier élément de la liste apparaît derrière tous les autres recouvrements alors que le dernier se trouve devant.

Chaque recouvrement de la liste de recouvrements de la caméra a les propriétés suivantes :

- `loc` permet d'obtenir ou de définir la position spécifique de la propriété `regPoint` du recouvrement, par rapport au coin supérieur gauche du rectangle de la caméra.
- `source` permet d'obtenir ou de définir la texture utilisée comme image source du recouvrement.
- `scale` permet d'obtenir ou de définir la valeur de l'échelle utilisée par le recouvrement. L'échelle détermine l'agrandissement du recouvrement, la valeur par défaut de cette propriété étant 1.0.
- `rotation` permet d'obtenir ou de définir la rotation du recouvrement, en degrés.
- `regPoint` permet d'obtenir ou de définir le point d'alignement du recouvrement par rapport au coin supérieur gauche de la texture.
- `blend` permet d'obtenir ou de définir la fusion du recouvrement à l'aide d'un nombre entier compris entre `0` et `100`, ce qui définit la transparence (`0`) ou l'opacité (`100`) du recouvrement.

Exemple

L'instruction suivante affiche la propriété d'échelle du troisième recouvrement de la liste des recouvrements de la caméra de l'image-objet.

```
put sprite(5).camera.overlay[3].scale
-- 0.5000
```

Voir aussi

`addOverlay`, `removeOverlay`, `bevelDepth`

pageHeight

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.pageHeight

// Syntaxe JavaScript
réfObjActeur.pageHeight;
```

Description

Propriété d'acteur champ ; renvoie la hauteur, en pixels, de la zone de l'acteur champ visible sur la scène.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante renvoie la hauteur de la portion visible de l'acteur champ Nouvelles du jour :

```
-- Syntaxe Lingo
trace(member("Nouvelles du jour").pageHeight)

// Syntaxe JavaScript
trace(member("Nouvelles du jour").pageHeight);
```

palette

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.palette

// Syntaxe JavaScript
réfObjActeur.palette;
```

Description

Propriété d'acteur ; détermine, pour les acteurs bitmap uniquement, la palette associée à l'acteur spécifié par *réfObjActeur*.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante affiche la palette affectée à l'acteur Feuilles dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(member("Feuilles").palette)

// Syntaxe JavaScript
put(member("Feuilles").palette);
```

paletteMapping

Utilisation

```
-- Syntaxe Lingo
_movie.paletteMapping

// Syntaxe JavaScript
_movie.paletteMapping;
```

Description

Propriété d'animation ; détermine si l'animation convertit (`TRUE`) ou non (`FALSE`, valeur par défaut) la palette des acteurs si elle est différente de la palette courante de l'animation. Lecture/écriture.

L'effet de cette propriété est similaire à celui obtenu avec la case à cocher Conversion automatique des palettes de la boîte de dialogue Propriétés de l'animation.

Pour afficher simultanément différents bitmaps avec des palettes différentes, donnez la valeur `TRUE` à `paletteMapping`. Director analyse la palette de référence de chaque acteur affiché (la palette affectée dans sa boîte de dialogue Propriétés de l'acteur) et, si elle diffère de la palette courante, trouve le plus proche équivalent pour chaque pixel dans la nouvelle palette.

Les couleurs du bitmap ne correspondant pas seront proches des couleurs d'origine.

La conversion est assez coûteuse en termes d'utilisation du processeur et il est généralement plus judicieux de régler la palette du bitmap à l'avance.

La conversion peut aussi produire des résultats indésirables. Si la palette change au milieu d'une séquence d'image-objet, le bitmap passe immédiatement à la nouvelle palette et apparaît dans des couleurs incorrectes. Cependant, si quelque chose rafraîchit l'écran – une transition ou une image-objet passant sur la scène – le rectangle affecté de l'écran apparaît dans les nouvelles couleurs.

Exemple

L'instruction suivante indique à l'animation de toujours convertir à sa palette lorsque nécessaire :

```
-- Syntaxe Lingo
_movie.paletteMapping = TRUE

// Syntaxe JavaScript
_movie.paletteMapping = true;
```

Voir aussi

`Animation`

paletteRef

Utilisation

```
member(quelActeur).paletteRef
the paletteRef
```

Description

Propriété d'acteur bitmap ; détermine la palette associée à un acteur bitmap. Les palettes intégrées de Director sont indiquées par des symboles (#systemMac, #rainbow, etc.). Les acteurs palettes sont considérés comme des références d'acteurs. Cette propriété est différente du comportement de la propriété d'acteur palette, qui renvoie un nombre positif pour les palettes de distribution et un nombre négatif pour les palettes intégrées de Director.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante affecte la palette système du Macintosh à l'acteur bitmap Coquille :

```
member("Coquille").paletteRef = #systemMac
```

pan

Utilisation

```
-- Syntaxe Lingo
réfObjPisteAudio.pan

// Syntaxe JavaScript
réfObjPisteAudio.pan;
```

Description

Propriété de piste audio ; indique la balance gauche/droite du son en cours de lecture sur une piste audio. Lecture/écriture.

La plage de valeurs s'étend de –100 à 100. La valeur -100 indique la lecture de la piste gauche uniquement. La valeur 100 indique la lecture de la piste droite uniquement. La valeur 0 indique une balance gauche/droite, entraînant le centrage du son. Pour les sons mono, pan affecte le haut-parleur (gauche ou droite) par lequel passera le son.

Vous pouvez modifier la balance d'un son à tout moment, mais si la piste audio exécute un fondu, la définition de la balance ne prendra effet qu'après l'exécution du fondu.

Vous pourrez voir un exemple de pan dans une animation en consultant l'animation Sound Control du dossier Learning/Lingo, lui-même dans le dossier de Director.

Exemple

Les instructions suivantes équilibrent le son de la piste audio 2, de la piste gauche vers la piste droite :

```
-- Syntaxe Lingo
repeat with x = -100 to 100
  sound(2).pan = x
end repeat

// Syntaxe JavaScript
for (var x = -100; x <= 100; x++) {
  sound(2).pan = x;
}
```

Voir aussi

Piste audio

pan (propriété QTVR)

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.pan

// Syntaxe JavaScript
réfObjImageObjet.pan;
```

Description

Propriété d'image-objet QuickTime VR ; le panoramique courant de l'animation QuickTime VR. La valeur est exprimée en degrés.

Cette propriété peut être testée et définie.

paragraph

Utilisation

```
expressionSousChaîne.paragraph[quelParagraphe]
expressionSousChaîne.paragraph[premierParagraphe..dernierParagraphe]
```

Description

Propriété d'acteur texte ; cette expression de sous-chaîne permet d'accéder à différents paragraphes dans un acteur texte.

Le paragraphe est délimité par un retour chariot.

```
put member("Texte Animé").paragraph[3]
```

Voir aussi

line...of

parent

Utilisation

```
member(quelActeur).model(quelModèle).parent
member(quelActeur).camera(quelleCaméra).parent
member(quelActeur).light(quelleLumière)parent
member(quelActeur).group(quelGroupe).parent
```

Description

Propriété 3D ; utilisée avec une référence de modèle, caméra, lumière ou groupe, cette propriété permet d'obtenir ou de définir le nœud parent de l'objet référencé. Le nœud parent peut être tout autre objet de modèle, de caméra, de lumière ou de groupe.

La propriété `transform` d'un objet définit son échelle, sa position et son orientation par rapport à son objet parent.

Donner à la propriété de parent d'un objet la valeur `Void` revient à retirer l'objet de l'univers à l'aide de la commande `removeFromWorld()`.

Donner à la propriété de parent d'un objet la valeur de l'objet de groupe World (`group("World")`) revient à ajouter un objet à l'univers à l'aide de la commande `addToWorld()`.

Vous pouvez aussi modifier la valeur de cette propriété à l'aide de la commande `addChild`.

Exemple

L'instruction suivante définit la propriété de parent du modèle Pneu. Son parent est le modèle Voiture.

```
member("Séquence").model("Pneu").parent = \
  member("Séquence").model("Voiture")
```

Voir aussi

`child (3D)`, `addChild`

password

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.password

// Syntaxe JavaScript
réfObjActeurOuImageObjet.password;
```

Description

Propriété d'acteur et image-objet RealMedia ; permet de définir le mot de passe nécessaire à l'accès à un flux RealMedia protégé. Vous ne pouvez pas utiliser cette propriété pour récupérer un mot de passe spécifié auparavant. Si un mot de passe a été défini, la valeur de cette propriété est la chaîne `"********"`. Si aucun mot de passe n'a encore été défini, la valeur de cette propriété est une chaîne vide.

Exemple

Les exemples suivants indiquent que le mot de passe a été défini pour le flux RealMedia de l'acteur Real ou de l'image-objet 2.

```
-- Syntaxe Lingo
put(sprite(2).password) -- "********"
put(member("Real").password) -- "********"

// Syntaxe JavaScript
put(sprite(2).password); // "********"
put(member("Real").password); // "********"
```

Les exemples suivants indiquent que le mot de passe n'a jamais été défini pour le flux RealMedia de l'acteur Real ou de l'image-objet 2.

```
-- Syntaxe Lingo
put(sprite(2).password) -- ""
put(member("Real").password) -- ""

// Syntaxe JavaScript
put(sprite(2).password); // ""
put(member("Real").password); // ""
```

Les exemples suivants définissent le mot de passe pour le flux RealMedia de l'image-objet 2 et l'acteur Real à `abracadabra`.

```
-- Syntaxe Lingo
sprite(2).password = "abracadabra"
member("Real").password = "abracadabra"

// Syntaxe JavaScript
sprite(2).password = "abracadabra";
member("Real").password = "abracadabra";
```

Voir aussi

`userName (RealMedia)`

path (animation)

Utilisation

```
-- Syntaxe Lingo
_movie.path

// Syntaxe JavaScript
_movie.path;
```

Description

Propriété d'animation ; indique le chemin d'accès du dossier de l'animation courante. En lecture seule.

Pour les noms de fichiers fonctionnant sur Windows et Macintosh, utilisez l'opérateur de chemin d'accès *@*.

Vous pourrez voir un exemple de `path` dans une animation en consultant l'animation Read and Write Text du dossier Learning/Lingo, lui-même dans le dossier de Director.

Exemple

L'instruction suivante affiche le chemin d'accès du dossier de l'animation courante :

```
-- Syntaxe Lingo
trace(_movie.path)

// Syntaxe JavaScript
trace(_movie.path);
```

L'instruction suivante lit le fichier son Accident.aif stocké dans le sous-dossier Sons du dossier de l'animation courante :

```
-- Syntaxe Lingo
sound(1).playFile(_movie.path & "Sons\Accident.aif")

// Syntaxe JavaScript
sound(1).playFile(_movie.path + "Sons\\Accident.aif");
```

Voir aussi

Animation

path (3D)

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  emitter.path
```

Description

Propriété 3D ; utilisée avec une ressource de modèle de type #particle, cette propriété permet d'obtenir ou de définir la propriété path de l'émetteur de particules de la ressource.

Cette propriété est une liste de vecteurs qui définit le chemin suivi par les particules tout au long de leur existence. La valeur par défaut de cette propriété est une liste vide [].

Exemple

Dans l'exemple suivant, systèmeThermique est une ressource de modèle de type #particle. L'instruction suivante spécifie que les particules de systèmeThermique suivront le chemin décrit par la liste de vecteurs.

```
member("Feux").modelResource("systèmeThermique").emitter.path = \
  [vector(0,0,0), vector(15,0,0), vector(30,30,-10)]
```

Voir aussi

pathStrength, emitter

pathName (acteur Flash)

Utilisation

```
member(quelActeurFlash).pathName
the pathName of member quelActeurFlash
```

Description

Propriété d'acteur ; contrôle l'emplacement d'un fichier externe dans lequel sont stockés les éléments d'un acteur animation Flash. Vous pouvez lier une animation Flash à un chemin d'accès sur un disque local ou réseau, ou vers une URL.

La définition du chemin d'accès d'un acteur non lié le convertit en un acteur lié.

Cette propriété peut être testée et définie. La propriété `pathName` d'un acteur non lié est une chaîne vide.

Cette propriété est identique à la propriété `fileName` pour d'autres types d'acteurs, `fileName` pouvant s'utiliser à la place de `pathName`.

Exemple

Le script `startMovie` suivant crée un nouvel acteur Flash au moyen de la commande `new`, définit la propriété `linked` de l'acteur nouvellement créé de sorte que ses éléments soient stockés dans un fichier externe, puis définit la propriété `pathName` de l'acteur comme emplacement d'une animation Flash sur le web :

```
on startMovie
  member(new(#flash)).pathName = \
"http://www.uneURL.com/monFlash.swf"
end
```

Voir aussi

`fileName (acteur)`, `linked`

pathStrength

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  emitter.pathStrength
```

Description

Propriété 3D ; utilisée avec une ressource de modèle de type `#particle`, cette propriété détermine la façon dont les particules suivent le chemin spécifié par la propriété `path` de l'émetteur. Sa valeur peut être comprise entre `0.0` (aucune force – les particules ne seront pas attirées par le chemin) et l'infini. Sa valeur par défaut est 0.1. Donner à la propriété `pathStrength` la valeur `0.0` est utile pour désactiver temporairement le chemin.

Plus la valeur de `pathStrength` sera élevée, plus le système de particules sera rigide. Des valeurs `pathStrength` élevées provoqueront le rebondissement très rapide des particules, à moins qu'une force modératrice soit également utilisée, telle que la propriété de particule `drag`.

Cette propriété peut être testée et définie.

Exemple

Dans l'exemple suivant, systèmeThermique est une ressource de modèle de type `#particle`. L'instruction suivante donne à la propriété `pathStrength` de systèmeThermique la valeur `0.97`. Si un chemin est indiqué par la propriété `emitter.path` de systèmeThermique, les particules suivront ce chemin de très près.

```
member("Chauffage").modelResource("systèmeThermique").emitter.\
  pathStrength = 0.97
```

Voir aussi

`path (3D)`, `emitter`

pattern

Utilisation

```
member(quelActeur).pattern
the pattern of member quelActeur
```

Description

Propriété d'acteur ; détermine le motif associé à la forme spécifiée. Les valeurs possibles sont les nombres correspondant aux puces de la palette des motifs de la fenêtre Outils. Si l'acteur forme est vide, le motif est appliqué à son bord externe.

Cette propriété peut être utile dans les animations possédant un contenu Shockwave pour changer des images en modifiant la mosaïque appliquée à une forme, ce qui permet d'économiser la mémoire requise par des bitmaps plus grands.

Cette propriété peut être testée et définie.

Exemple

Les instructions suivantes font de l'acteur forme Zone cible une forme remplie et lui affectent le motif 1, qui est une couleur unie :

```
member("Zone cible").filled = TRUE
member("Zone cible").pattern = 1
```

Le gestionnaire suivant effectue un cycle sur huit mosaïques, en décalant le numéro de chacune par rapport au précédent, ce qui permet de créer des animations utilisant des bitmaps de plus petite taille :

```
on exitFrame
  motifActuel = member("Arrière-plan").pattern
  motifSuivant = 57 + ((motifActuel - 56) mod 8)
  member("Arrière-plan").pattern = motifSuivant
  go the frame
end
```

pausedAtStart (Flash, vidéo numérique)

Utilisation

```
member(quelActeurFlashOuVidéoNumérique).pausedAtStart
the pausedAtStart of member quelActeurFlashOuVidéoNumérique
```

Description

Propriété d'acteur ; contrôle si la vidéo numérique ou l'animation Flash est lue lorsqu'elle apparaît sur la scène. Si cette propriété est `TRUE`, la vidéo numérique ou l'animation Flash n'est pas lue lorsqu'elle apparaît. Si elle est `FALSE`, la vidéo numérique ou l'animation Flash est lue dès qu'elle apparaît.

Pour un acteur vidéo numérique, la propriété spécifie si la case En pause de la boîte de dialogue Propriétés de l'acteur vidéo numérique est cochée ou non.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante coche la case En pause dans la boîte de dialogue Propriétés de l'acteur vidéo numérique pour l'animation QuickTime Chaise pivotante :

```
member("Chaise pivotante").pausedAtStart = TRUE
```

pausedAtStart (RealMedia, Windows Media)

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.pausedAtStart

// Syntaxe JavaScript
réfObjActeurOuImageObjet.pausedAtStart;
```

Description

Propriété d'acteur ou image-objet RealMedia et Windows Media ; permet d'obtenir ou de définir si la lecture d'un flux RealMedia ou Windows Media présent sur la scène démarre automatiquement à la fin de la mise en tampon (`FALSE` ou 0) ou non (`TRUE` ou 1). Lecture/écriture.

Cette propriété peut être définie en tant qu'expression évaluant `TRUE` ou `FALSE`. Les valeurs entières autres que `1` ou `0` sont traitées comme `TRUE`. Le paramètre par défaut de cette propriété est `FALSE`. Vous pouvez définir cette propriété sur `TRUE` en sélectionnant En pause dans l'affichage graphique de l'inspecteur des propriétés.

Si cette propriété est définie sur `FALSE`, vous devez cliquer sur le bouton Lire dans la fenêtre RealMedia ou Windows Media (ou sur un bouton que vous avez créé dans ce but dans votre animation), ou appeler la méthode `play()` pour lire l'image-objet une fois la mise en tampon effectuée.

Cette propriété n'affecte que la lecture basée sur le scénario, et non la lecture dans la fenêtre RealMedia ou Windows Media.

Exemple

Les exemples suivants indiquent que la propriété `pausedAtStart` de l'image-objet 2 et de l'acteur Real a pour valeur `FALSE`, ce qui signifie que le flux RealMedia sera automatiquement lu à la fin de la mise en tampon.

```
-- Syntaxe Lingo
put(sprite(2).pausedAtStart) -- 0
put(member("Real").pausedAtStart) -- 0

// Syntaxe JavaScript
put(sprite(2).pausedAtStart); // 0
put(member("Real").pausedAtStart); // 0
```

Les exemples suivants donnent à la propriété `pausedAtStart` de l'image-objet 2 et de l'acteur Real la valeur `TRUE`, ce qui signifie que le flux RealMedia ne sera pas lu jusqu'à l'appel de la commande `play`.

```
-- Syntaxe Lingo
sprite(2).pausedAtStart = TRUE
member("Real").pausedAtStart = TRUE

// Syntaxe JavaScript
sprite(2).pausedAtStart = 1;
member("Real").pausedAtStart = 1;
```

L'exemple suivant utilise la propriété `pausedAtStart` pour mettre en tampon une image-objet RealMedia hors de la scène, puis la lire sur la scène, une fois la mise en tampon effectuée. Dans cet exemple, la propriété `pausedAtStart` de l'acteur RealMedia est définie sur `TRUE`. Une occurrence de cet acteur est située en dehors de la scène, sur la piste d'image-objet 1. Le script d'image suivant doit être placé dans la plage de l'image-objet :

```
-- Syntaxe Lingo
on exitFrame me
  if sprite(1).state > 3 then -- vérifier si la mise en tampon est effectuée
    sprite(1).locH = 162
    sprite(1).locV = 118
    sprite(1).play() -- positionner et lire l'image-objet
  end if
end

// Syntaxe JavaScript
function exitFrame() {
  var st = sprite(1).state;
  if (st > 3) { // vérifier si la mise en tampon est effectuée
    sprite(1).locH = 162;
    sprite(1).locV = 118;
    sprite(1).play(); // positionner et lire l'image-objet
  }
}
```

L'image-objet RealMedia sera mise en tampon en dehors de la scène, puis apparaîtra sur la scène et sera lue dès que la mise en tampon sera effectuée.

percentBuffered

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.percentBuffered

// Syntaxe JavaScript
réfObjActeurOuImageObjet.percentBuffered;
```

Description

Propriété d'acteur ou d'image-objet RealMedia ; renvoie le pourcentage du tampon rempli par le flux RealMedia chargé à partir d'un fichier local ou du serveur. Lorsque cette propriété atteint 100, la mémoire tampon est remplie et la lecture du flux RealMedia démarre si la propriété `pausedAtStart` n'est pas définie sur `TRUE`. Cette propriété est dynamique en cours de lecture et ne peut pas être définie.

La mémoire tampon est un type de mémoire cache qui contient la partie de l'animation qui va être lue, généralement les premières secondes. Le flux entre en mémoire tampon lors de sa lecture en flux continu dans RealPlayer et quitte la mémoire tampon dès le démarrage de la lecture du clip. La mémoire tampon permet de visualiser le contenu sans télécharger l'ensemble du fichier, permet d'éviter les congestions sur le réseau et empêche que des défaillances de disponibilité de la bande passante interrompent la lecture.

Le processus de mise en mémoire tampon est initialisé par la commande `play` et la lecture de la section du flux comprise dans la mémoire tampon démarre une fois la mémoire tampon remplie à 100 %. Le processus de mise en mémoire tampon prenant quelques secondes, on notera un délai entre l'appel de la commande `play` et le début réel de la lecture du flux. L'utilisation de la commande `pausedAtStart` permet de lire le flux en dehors de la scène pendant le processus de mise en mémoire tampon, puis d'afficher le flux sur la scène lorsque la lecture démarre. Pour plus d'informations, consultez l'entrée `pausedAtStart (RealMedia, Windows Media)`, page 968.

Exemple

Les exemples suivants indiquent que 56 % du flux RealMedia de l'image-objet 2 et de l'acteur Real a été mis en tampon.

```
-- Syntaxe Lingo
put(sprite(2).percentBuffered) -- 56
put(member("Real").percentBuffered) -- 56

// Syntaxe JavaScript
put(sprite(2).percentBuffered); // 56
put(member("Real").percentBuffered); // 56
```

Voir aussi

`mediaStatus (RealMedia, Windows Media)`, `pausedAtStart (RealMedia, Windows Media)`, `state (RealMedia)`

percentPlayed

Utilisation

```
member(quelActeur).percentPlayed
the percentPlayed of member quelActeur
```

Description

Propriété d'acteur Shockwave Audio (SWA) ; renvoie le pourcentage du fichier SWA spécifié qui a été lu.

Cette propriété ne peut être lue qu'une fois la lecture du son SWA déclenchée ou lorsqu'il a été préchargé à l'aide de la commande `preLoadBuffer`. Cette propriété ne peut pas être définie.

Exemple

Le gestionnaire suivant affiche le pourcentage de l'acteur SWA Frank Sinatra lu en flux continu et place cette valeur dans l'acteur champ Pourcentage lu :

```
on exitFrame
  quelEtat = member("Frank Sinatra").state
  if quelEtat > 1 AND quelEtat < 9 then
    member("Pourcentage lu").text = /
string(member("Frank Sinatra").percentPlayed)
  end if
end
```

Voir aussi

percentStreamed (acteur)

percentStreamed (3D)

Utilisation

```
member(quelActeur).percentStreamed
```

Description

Propriété 3D ; permet d'obtenir le pourcentage d'un acteur 3D qui a été chargé en mémoire. Cette propriété indique la quantité qui a été chargée du fichier initial ou du dernier fichier demandé. La valeur renvoyée est un nombre entier compris entre 0 et 100. Il n'y a pas de valeur par défaut pour cette propriété.

Exemple

L'instruction suivante indique que le chargement de l'acteur séquenceDeFête est terminé.

```
put member("séquenceDeFête").percentStreamed
-- 100
```

percentStreamed (acteur)

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.percentStreamed

// Syntaxe JavaScript
réfObjActeurOuImageObjet.percentStreamed;
```

Description

Propriété d'acteur Shockwave Audio (SWA) et Flash, et propriété d'image-objet QuickTime.

Pour les sons SWA en flux continu, prend la valeur du pourcentage d'un fichier SWA déjà lu en flux continu d'un serveur HTTP ou FTP. Pour les sons SWA, cette propriété diffère de la propriété percentPlayed en cela qu'elle indique la quantité du fichier mise en tampon mais pas encore lue. Cette propriété ne peut être lue qu'une fois la lecture du son SWA déclenchée ou lorsqu'il a été préchargé à l'aide de la commande preLoadBuffer.

Pour les acteurs animation Flash, cette propriété a pour valeur le pourcentage d'une animation Flash lue en flux continu dans la mémoire.

Pour les images-objets QuickTime, cette propriété prend la valeur du pourcentage du fichier QuickTime lu.

Cette propriété est une valeur comprise entre 0 et 100 %. Pour un fichier situé sur un disque local, la valeur est 100. Pour les fichiers lus en continu depuis Internet, la valeur `percentStreamed` augmente au fur et à mesure de la réception des octets. Cette propriété ne peut pas être définie.

Exemple

L'exemple suivant indique le pourcentage de l'acteur SWA Ray Charles lu en flux continu déjà récupéré et l'affiche dans un champ :

```
-- Syntaxe Lingo
on exitFrame
  quelEtat = member("Ray Charles").state
  if quelEtat > 1 AND quelEtat < 9 then
    member("Affichage du pourcentage lu").text = \
  string(member("Ray Charles").percentStreamed)
  end if
end

// Syntaxe JavaScript
function exitFrame() {
  var quelEtat = member("Ray Charles").state;
  var pcStm = new String(member("Ray Charles").percentStreamed);
  if (quelEtat > 1 && quelEtat < 9) {
    member("Affichage du pourcentage lu").text = pcStm;
  }
}
```

Le script d'image suivant impose une boucle à la tête de lecture dans l'image courante tant que moins de 60 % de l'animation Flash appelée Ecran d'accueil ont été transférés en mémoire :

```
-- Syntaxe Lingo
on exitFrame
  if member("Ecran d'accueil").percentStreamed < 60 then
    _movie.go(_movie.frame)
  end if
end

// Syntaxe JavaScript
function exitFrame() {
  var ssStrm = member("Ecran d'accueil").percentStreamed;
  if (ssStrm < 60) {
    _movie.go(_movie.frame);
  }
}
```

Voir aussi

`percentPlayed`

period

Utilisation

objetDeTemporisation.period

Description

Propriété d'objet ; nombre de millisecondes compris entre les événements de temporisation transmis par l'objetDeTemporisation au gestionnaire de temporisation.

Cette propriété peut être testée et définie.

Exemple

Le gestionnaire de temporisation suivant diminue la valeur period de temporisation d'une seconde à chaque appel, jusqu'au moment où une période minimale de 2 secondes (2 000 millisecondes) est atteinte :

```
on gestionDeTemporisation objetDeTemporisation
  if objetDeTemporisation.period > 2000 then
    objetDeTemporisation.period = objetDeTemporisation.period - 1000
  end if
end handleTimeout
```

Voir aussi

name (temporisation), persistent, target, time (objet de temporisation), timeout(), timeoutHandler, timeoutList

persistent

Utilisation

objetDeTemporisation.persistent

Description

Propriété d'objet ; détermine si l'*objetDeTemporisation* est supprimé de la liste timeoutList lors de l'arrêt de l'animation en cours. Si la valeur est TRUE, *objetDeTemporisation* reste actif. Si la valeur est FALSE, l'objet de temporisation est supprimé lors de l'arrêt de l'animation. La valeur par défaut est FALSE.

La définition de cette propriété sur TRUE permet à un objet de temporisation de continuer à générer des événements de temporisation dans d'autres animations. Ceci peut s'avérer pratique lorsqu'une animation passe à une autre animation par l'intermédiaire de la commande go to movie.

Exemple

Le gestionnaire `prepareMovie` suivant crée un objet de temporisation qui restera actif après la déclaration d'arrêt de l'animation :

```
on prepareMovie
  -- génère un objet de temp. qui envoie un événement toutes les 60 min.
  timeout("reste").new(1000 * 60 * 60, #gestionnaireDuReste)
  timeout("reste").persistent = TRUE
end
```

Voir aussi

name (temporisation), period, target, time (objet de temporisation), timeout(), timeoutHandler, timeoutList

picture (acteur)

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.picture

// Syntaxe JavaScript
réfObjActeur.picture;
```

Description

Propriété d'acteur ; détermine l'image associée à un acteur bitmap, texte ou PICT. Pour mettre à jour le point d'alignement d'un acteur ou les modifications d'une image après l'avoir lié de nouveau au moyen de la propriété `fileName`, utilisez l'instruction suivante :

```
member(quelActeur).picture = member(quelActeur).picture
```

où vous remplacez *quelActeur* par le nom ou le numéro de l'acteur.

Les modifications effectuées sur les acteurs étant conservées en RAM, cette propriété trouve une meilleure utilisation au cours de la phase de programmation. Evitez de l'utiliser dans les projections.

Cette propriété peut être testée et modifiée.

Exemple

L'instruction suivante associe la variable `contenuPict` à l'image de l'acteur Crépuscule :

```
-- Syntaxe Lingo
contenuPict = member("Crépuscule").picture

// Syntaxe JavaScript
var contenuPict = member("Crépuscule").picture;
```

Voir aussi

type (image-objet)

picture (fenêtre)

Utilisation

```
-- Syntaxe Lingo
réfObjFenêtre.picture

// Syntaxe JavaScript
réfObjFenêtre.picture;
```

Description

Propriété de fenêtre ; offre un moyen d'obtenir un aperçu du contenu courant d'une fenêtre – la fenêtre Scène ou une animation dans une fenêtre (MIAW). En lecture seule.

Vous pouvez appliquer les données du bitmap résultant à un bitmap existant ou les utiliser pour en créer un nouveau.

S'il n'existe aucune image, cette propriété renvoie `VOID` (Lingo) ou `null` (syntaxe JavaScript).

Exemple

L'instruction suivante saisit le contenu courant de la scène et le place dans un acteur bitmap :

```
-- Syntaxe Lingo
member("Image de la scène").picture = _movie.stage.picture

// Syntaxe JavaScript
member("Image de la scène").picture = _movie.stage.picture;
```

Voir aussi

`Fenêtre`

platform

Utilisation

```
the platform
```

Description

Propriété système ; indique le type de plate-forme pour lequel la projection a été créée.

Cette propriété peut être testée, mais pas définie.

Les valeurs possibles sont :

Valeur possible	Plate-forme correspondante
`Macintosh,PowerPC`	Macintosh PowerPC
`Windows,32`	Windows 95 ou Windows NT

Pour assurer une compatibilité future et permettre l'addition de valeurs, il est préférable d'utiliser `contains`.

Exemple

L'instruction suivante détermine si une projection a été créée pour Windows 95 ou Windows NT :

```
on exitFrame
  if the platform contains "Windows,32" then
    castLib("Graphiques Win95").name = "Interface"
  end if
end
```

Voir aussi

runMode

playBackMode

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.playBackMode

// Syntaxe JavaScript
réfObjActeurOuImageObjet.playBackMode;
```

Description

Propriété d'acteur et d'image-objet ; contrôle la cadence d'un acteur animation Flash ou d'un acteur GIF animé selon les valeurs suivantes :

- #normal (valeur par défaut) – Lit l'animation Flash ou le fichier GIF à une cadence aussi proche que possible de la cadence originale.
- #lockStep – Lit l'animation Flash ou le fichier GIF à la cadence de l'animation Director.
- #fixed – Lit l'animation Flash ou le fichier GIF à la cadence spécifiée par la propriété fixedRate.

Cette propriété peut être testée et définie.

Exemple

Le script d'image-objet suivant définit la cadence d'une image-objet d'animation Flash comme étant celle de l'animation Director :

```
-- Syntaxe Lingo
property spriteNum

on beginSprite (me)
  sprite(spriteNum).playBackMode = #lockStep
end

// Syntaxe JavaScript
function beginSprite() {
  sprite(this.spriteNum).playBackMode = symbol("lockStep");
}
```

Voir aussi

fixedRate

playing

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.playing

// Syntaxe JavaScript
réfObjImageObjet.playing;
```

Description

Propriété d'image-objet Flash ; indique si une animation Flash est en cours de lecture (TRUE) ou arrêtée (FALSE).

Cette propriété peut être testée, mais pas définie.

Exemple

Le script d'image suivant détermine si l'image-objet animation Flash de la piste 5 est en cours de lecture. Dans la négative, il démarre l'animation :

```
-- Syntaxe Lingo
on enterFrame
  if not sprite(5).playing then
    sprite(5).play()
  end if
end

// Syntaxe JavaScript
function enterFrame() {
  var plg = sprite(5).playing;
  if (plg == 0) {
    sprite(5).play();
  }
}
```

playing (3D)

Utilisation

```
member(quelActeur).model(quelModèle).keyframePlayer.playing
member(quelActeur).model(quelModèle).bonesPlayer.playing
```

Description

Propriété de modificateur 3D #keyframePlayer et #bonesPlayer ; indique si le moteur de lecture d'animation du modificateur est en marche (TRUE) ou en pause (FALSE).

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante indique que le moteur de lecture d'animation #keyframePlayer du modèle Martien3 est actuellement en marche.

```
put member("nouveauxMartiens").model("Martien3").keyframePlayer.playing
-- 1
```

Voir aussi

play() (3D), pause() (3D), playlist, queue() (3D)

playlist

Utilisation

```
member(quelActeur).model(quelModèle).keyframePlayer.playlist
member(quelActeur).model(quelModèle).bonesPlayer.playlist
```

Description

Propriété de modificateur 3D `#keyframePlayer` et `#bonesPlayer` ; renvoie une liste linéaire de listes de propriétés, chacune représentant un mouvement dont la lecture est mise en pause par le modificateur.

Chaque liste de propriétés doit contenir les propriétés suivantes :

- `#name` est le nom du mouvement à lire.
- `#loop` indique si le mouvement doit être lu en boucle.
- `#startTime` est le moment, en millisecondes, où la lecture de l'animation doit commencer.
- `#endTime` est le moment, en millisecondes, où la lecture de l'animation se termine ou le moment où le mouvement doit être mis en boucle. Une valeur négative indique que le mouvement doit être lu jusqu'à la fin.
- `#scale` est le taux de lecture du mouvement qui doit être multiplié par la propriété `playRate` du modificateur pour déterminer la cadence réelle de la lecture du mouvement.

La propriété `playlist` peut être testée mais pas définie. Utilisez les commandes `queue()`, `play()`, `playNext()` et `removeLast()` pour la manipuler.

Exemple

L'instruction suivante affiche les mouvements mis en attente pour le modèle Promeneur dans la fenêtre Messages. Deux mouvements sont actuellement en attente : Marche et Saut.

```
put member("Parc").model("Promeneur").bonesPlayer.playList
-- [[#name: "Marche", #loop: 1, #startTime: 1500, #endTime: 16000, \
  #scale:1.0000, #offset: 0], [#name: "Saut", #loop: 1, \
  #startTime: 0, #endTime: 1200, #scale: 1.0000, #offset: 0]]
```

Voir aussi

`play() (3D)`, `playNext() (3D)`, `removeLast()`, `queue() (3D)`

playRate (3D)

Utilisation

```
member(quelActeur).model(quelModèle).bonesPlayer.playRate
member(quelActeur).model(quelModèle).keyframePlayer.playRate
```

Description

Propriété 3D de modificateur `#keyframePlayer` et `#bonesPlayer` ; multiplicateur d'échelle pour le temps local des mouvements lorsqu'ils sont lus. Cette propriété ne détermine que partiellement la cadence à laquelle les mouvements sont exécutés par le modèle.

La lecture d'un mouvement par un modèle est le résultat de la commande `play()` ou `queue()`. Le paramètre `scale` de la commande `play()` ou `queue()` est multiplié par la propriété `playRate` du modificateur et la valeur résultante est la cadence à laquelle le mouvement en question sera lu.

Exemple

L'instruction suivante donne à la propriété `playRate` du modificateur `keyframePlayer` du modèle martienVert la valeur 3.

```
member("nouveauxMartiens").model("MartienVert").keyframePlayer.playRate = 3
```

Voir aussi

play() (3D), queue() (3D), playlist, currentTime (3D)

playRate (DVD)

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.playRate

// Syntaxe JavaScript
réfObjDvd.playRate;
```

Description

Propriété DVD ; spécifie la cadence à laquelle un DVD lance une lecture vers l'avant ou vers l'arrière à partir de l'emplacement courant. Lecture/écriture.

Une valeur négative lance la lecture du DVD vers l'arrière et une valeur positive lance sa lecture vers l'avant.

Voir aussi

DVD

playRate (QuickTime, AVI)

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.playRate

// Syntaxe JavaScript
réfObjDvd.playRate;
```

Description

Propriété d'image-objet vidéo numérique ; contrôle la vitesse de lecture de la vidéo numérique dans une piste spécifique. `playRate` est une valeur qui dirige la lecture d'une vidéo numérique. Une valeur de 1 spécifie une lecture normale vers l'avant, une valeur de -1 spécifie une lecture en arrière, et une valeur de 0 spécifie l'arrêt de la lecture. Vous pouvez utiliser des valeurs plus ou moins élevées. Par exemple, une valeur de 0,5 provoque une lecture plus lente que la normale. Notez cependant qu'il peut arriver que certaines images soient ignorées lorsque la valeur de la propriété d'image-objet `playRate` dépasse 1. La quantité d'images ignorées dépend d'autres facteurs, tels que les performances de l'ordinateur sur lequel l'animation est lue ou de l'étirement possible de l'image-objet vidéo numérique.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante définit une vitesse de lecture normale de la vidéo numérique contenue dans l'image-objet vidéo numérique de la piste 9 :

```
-- Syntaxe Lingo
sprite(9).playRate = 1

// Syntaxe JavaScript
sprite(9).playRate = 1;
```

L'instruction suivante provoque la lecture en sens inverse de la vidéo numérique de l'image-objet de la piste 9 :

```
-- Syntaxe Lingo
sprite(9).playRate = -1

// Syntaxe JavaScript
sprite(9).playRate = -1;
```

Voir aussi

duration (acteur), currentTime (QuickTime, AVI)

playRate (Windows Media)

Utilisation

```
-- Syntaxe Lingo
réfObjWindowsMedia.playRate

// Syntaxe JavaScript
réfObjWindowsMedia.playRate;
```

Description

Propriété Windows Media. Détermine la cadence de lecture d'un acteur Windows Media. Lecture/écriture.

Exemple

L'instruction suivante affiche, dans la fenêtre Messages, la cadence de lecture de l'acteur 10 :

```
-- Syntaxe Lingo
trace(member(10).playRate)

// Syntaxe JavaScript
trace(member(10).playRate);
```

Voir aussi

Windows Media

pointAtOrientation

Utilisation

```
member(quelActeur).model(quelModèle).pointAtOrientation
member(quelActeur).group(quelGroupe).pointAtOrientation
member(quelActeur).light(quelleLumière).pointAtOrientation
member(quelActeur).camera(quelleCaméra).pointAtOrientation
```

Description

Propriété 3D de modèle, de groupe, de lumière et de caméra ; permet d'obtenir ou de définir la réponse de l'objet référencé à la commande `pointAt`. Cette propriété est une liste linéaire de deux vecteurs relatifs à l'objet, le premier définissant la direction avant de l'objet et le second la direction vers le haut de l'objet.

Les directions avant et vers le haut de l'objet n'ont pas besoin d'être perpendiculaires, mais ne doivent pas être parallèles.

Exemple

L'instruction suivante affiche le vecteur de direction avant et le vecteur vertical relatif à l'objet du modèle bip01 :

```
put member("séquence").model("bip01").pointAtOrientation
-- [vector(0.0000, 0.0000, -1.0000), vector(0.0000, 1.0000, 0.0000)]
```

Voir aussi

`pointAt`

pointOfContact

Utilisation

```
collisionData.pointOfContact
```

Description

Propriété 3D `collisionData` ; renvoie un vecteur décrivant le point de contact d'une collision entre deux modèles.

L'objet `collisionData` est envoyé comme argument avec les événements `#collideWith` et `#collideAny` au gestionnaire spécifié dans les commandes `registerForEvent`, `registerScript` et `setCollisionCallback`.

Les événements `#collideWith` et `#collideAny` sont envoyés lors d'une collision entre deux modèles associés à des modificateur de collision. La propriété `resolve` des modificateurs des modèles doit être `TRUE`.

Cette propriété peut être testée, mais pas définie.

Exemple

L'exemple suivant est constitué de deux parties. La première partie est la première ligne de code, qui enregistre le gestionnaire `#explose` de l'événement `#collideAny`. La seconde partie est le gestionnaire `#explose`. Lorsque deux modèles de l'acteur maSéquence entrent en collision, le gestionnaire `#explose` est appelé, et l'argument `collisionData` lui est envoyé. Les neuf premières lignes du gestionnaire `#explose` créent la ressource de modèle sourceDétincelles et en définissent les propriétés. Cette ressource de modèle est une simple explosion de particules. La dixième ligne du gestionnaire crée un modèle modèleDétincelles à l'aide de la ressource de modèle sourceDétincelles. La dernière ligne du gestionnaire définit la position de modèleDétincelles à l'emplacement de la collision. L'effet général est une explosion d'étincelles causée par une collision.

```
member("maSéquence").registerForEvent(#collideAny, #explose, 0)

on explose me, collisionData
  nmr = member("maSéquence").newModelResource("sourceDétincelles", #particle)
  nmr.emitter.mode = #burst
  nmr.emitter.loop = 0
  nmr.emitter.minSpeed = 30
  nmr.emitter.maxSpeed = 50
  nmr.emitter.direction = vector(0, 0, 1)
  nmr.colorRange.start = rgb(0, 0, 255)
  nmr.colorRange.end = rgb(255, 0, 0)
  nmr.lifetime = 5000
  nm = member("maSéquence").newModel("modèleDétincelles", nmr)
  nm.transform.position = collisionData.pointOfContact
end
```

Voir aussi

`modelA`, `modelB`

position (transformation)

Utilisation

```
member(quelActeur).node(quelNœud).transform.position
member(quelActeur).node(quelNoeud).getWorldTransform().\
  position
transformation.position
```

Description

Propriété 3D ; permet d'obtenir ou de définir le composant de position d'une transformation. Une transformation définit une échelle, une position et une rotation au sein d'un cadre de référence donné. La valeur par défaut de cette propriété est `vector(0,0,0)`.

Un nœud peut être un objet de caméra, groupe, lumière ou modèle. La définition de la `position` d'une transformation de nœud définit la position de cet objet dans le cadre de référence de la transformation. La définition de la propriété `position` de la transformation relative à l'univers d'un objet à l'aide de `getWorldTransform().position` définit la position de l'objet par rapport à l'origine de l'univers. La définition de la propriété `position` de la transformation relative à l'univers d'un objet à l'aide de `transform.position` définit la position de l'objet par rapport à son nœud parent.

La propriété `worldPosition` d'un objet de modèle, de lumière, de caméra ou de groupe, est un raccourci de la version `getWorldTransform().position` de cette propriété pour cet objet.

Exemple

L'instruction suivante affiche la position relative au parent du modèle Pneu.

```
put member("Séquence").model("Pneu").transform.position
-- vector( -2.5000, -15 000, 20.0000 )
```

L'instruction suivante affiche la position relative à l'univers du modèle Pneu.

```
put member("Séquence").model("Pneu").getWorldTransform().position
-- vector( -2.5000, 5.0000, -10.0000 )
```

Les instructions suivantes enregistrent d'abord la transformation d'univers du modèle Pneu dans la variable `transformTemp`, puis affichent le composant de position de cette transformation.

```
transformTemp = member("Séquence").model("Pneu").getWorldTransform()
put transformTemp.position
-- vector( -2.5000, 5.0000, -10.0000 )
```

Voir aussi

`transform (propriété)`, `getWorldTransform()`, `rotation (transformation)`, `scale (transformation)`

positionReset

Utilisation

```
member(quelActeur).model(quelModèle).bonesPlayer.\
  positionReset
member(quelActeur).model(quelModèle).keyframePlayer.\
  positionReset
```

Description

Propriété 3D de modificateur `keyframePlayer` et `bonesPlayer` ; indique si le modèle retourne à sa position de départ à la fin d'un mouvement (`TRUE`) ou non (`FALSE`).

La valeur par défaut de cette propriété est `TRUE`.

Exemple

L'instruction suivante empêche le modèle Monstre de retourner à sa position originale lorsqu'il finit l'exécution d'un mouvement.

```
member("nouveauMartien").model("Monstre").keyframePlayer.\
  positionReset = FALSE
```

Voir aussi

`currentLoopState`

posterFrame

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.posterFrame

// Syntaxe JavaScript
réfObjActeur.posterFrame;
```

Description

Propriété d'acteur Flash ; contrôle l'image d'un acteur animation Flash utilisée pour son image miniature. Cette propriété spécifie un nombre entier correspondant au numéro d'une image de l'animation Flash.

Cette propriété peut être testée et définie. La valeur par défaut est 1.

Exemple

Le gestionnaire suivant accepte comme paramètres une référence à un acteur animation Flash et un numéro d'image, puis place la miniature de l'animation spécifiée sur le numéro d'image spécifié :

```
-- Syntaxe Lingo
on redéfinirLaMiniature(quelleAnimationFlash, quelleImage)
  member(quelleAnimationFlash).posterFrame = quelleImage
end

// Syntaxe JavaScript
function redéfinirLaMiniature(quelleAnimationFlash, quelleImage) {
  member(quelleAnimationFlash).posterFrame = quelleImage;
}
```

preferred3dRenderer

Utilisation

```
-- Syntaxe Lingo
_movie.preferred3dRenderer

// Syntaxe JavaScript
_movie.preferred3dRenderer;
```

Description

Propriété d'animation ; permet d'obtenir ou de définir le moteur de rendu par défaut utilisé pour tracer des images-objets 3D d'une animation particulière si ce moteur de rendu est disponible sur la machine cliente. Lecture/écriture.

Si le moteur de rendu n'est pas disponible sur la machine cliente, l'animation sélectionne le moteur de rendu disponible le plus approprié.

Les valeurs possibles de cette propriété sont les suivantes :

- `#openGL` spécifie les pilotes openGL d'accélération matérielle fonctionnant sur les plates-formes Macintosh et Windows.
- `#directX7_0` spécifie les pilotes DirectX 7 d'accélération matérielle fonctionnant uniquement sur les plates-formes Windows.

- `#directX5_2` spécifie les pilotes DirectX 5.2 d'accélération matérielle fonctionnant uniquement sur les plates-formes Windows.
- `#software` spécifie le moteur de rendu logiciel intégré à Director fonctionnant avec les plates-formes Macintosh et Windows.
- `#auto` spécifie que le moteur de rendu le plus approprié doit être choisi. Il s'agit de la valeur par défaut de cette propriété.

La valeur définie pour cette propriété est utilisée par défaut pour la propriété `renderer` de l'objet de services de rendu.

Cette propriété diffère de la propriété `renderer` de l'objet `getRendererServices()` car `preferred3dRenderer` spécifie le moteur de rendu préféré à utiliser, alors que la propriété `renderer` de l'objet `getRendererServices()` indique le moteur de rendu actuellement utilisé par l'animation.

Shockwave Player permet de spécifier le moteur de rendu à l'aide du menu contextuel correspondant. Si l'utilisateur sélectionne l'option visant à respecter les paramètres de contenu, le moteur de rendu spécifié par les propriétés `renderer` ou `preferred3DRenderer` est utilisé pour dessiner l'animation (s'il est disponible sur le système de l'utilisateur) ; sinon, c'est le moteur de rendu sélectionné par l'utilisateur qui est utilisé.

Exemple

L'instruction suivante permet à l'animation de choisir le meilleur moteur de rendu 3D disponible sur le système de l'utilisateur.

```
-- Syntaxe Lingo
_movie.preferred3dRenderer = #auto

// Syntaxe JavaScript
_movie.preferred3dRenderer = "auto";
```

Voir aussi

`getRendererServices()`, `Animation`, `renderer`

preLoad (3D)

Utilisation

```
member(quelActeur).preload
référenceDacteur.preload
```

Description

Propriété 3D ; permet de savoir ou de définir si les données sont chargées avant la lecture (`TRUE`) ou si leur chargement s'effectue en flux continu pendant la lecture (`FALSE`). Cette propriété ne peut être utilisée qu'avec des fichiers liés. La valeur par défaut est `FALSE`.

Dans Director, la définition de la propriété `preLoad` sur `TRUE` entraîne le chargement complet de l'acteur avant le début de la lecture. Dans Shockwave Player, la définition de la propriété `preLoad` sur `TRUE` entraîne la mise en flux continu de l'acteur au début de la lecture. Avant d'exécuter les opérations Lingo dans un acteur 3D en cours de lecture en flux continu, vérifiez si la propriété `state` de l'acteur a une valeur supérieure ou égale à 2.

Exemple

L'instruction suivante donne à la propriété `preload` de l'acteur séquenceDeFête la valeur `FALSE`, ce qui permet la lecture en flux continu des médias externes liés à l'animation dans l'acteur séquenceDeFête en cours de lecture.

```
member("séquenceDeFête").preload = FALSE
member("Univers 3D").preload
```

Voir aussi

`state (3D)`

preLoad (acteur)

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.preLoad

// Syntaxe JavaScript
réfObjActeur.preLoad;
```

Description

Propriété d'acteur ; détermine si l'acteur vidéo numérique spécifié par *`réfObjActeur`* peut être préchargé en mémoire (`TRUE`) ou non (`FALSE`, valeur par défaut). L'état `TRUE` a le même effet que la sélection de Activer le préchargement dans la boîte de dialogue Propriétés de l'acteur vidéo numérique.

Pour les acteurs animation Flash, cette propriété détermine si une animation Flash doit être entièrement chargée en RAM avant l'affichage de la première image d'une image-objet (`TRUE`) ou si l'animation peut être transférée en mémoire pendant sa lecture (`FALSE`, valeur par défaut). Cette propriété ne fonctionne qu'avec les animations Flash liées dont les éléments sont stockés dans un fichier externe ; elle n'a aucun effet sur les acteurs dont les éléments sont stockés dans la distribution. Les propriétés `streamMode` et `bufferSize` déterminent comment l'acteur est transféré en mémoire.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante indique dans la fenêtre Messages si l'animation QuickTime Chaise pivotante peut être préchargée en mémoire :

```
-- Syntaxe Lingo
put(member("Chaise pivotante").preload)

// Syntaxe JavaScript
put(member("Chaise pivotante").preload);
```

Voir aussi

`bufferSize`, `streamMode`

preLoadEventAbort

Utilisation

```
-- Syntaxe Lingo
_movie.preLoadEventAbort

// Syntaxe JavaScript
_movie.preLoadEventAbort;
```

Description

Propriété d'animation ; spécifie si le fait d'appuyer sur une touche ou de cliquer avec la souris peut arrêter le préchargement des acteurs (`TRUE`) ou non (`FALSE`, valeur par défaut). Lecture/écriture.

La modification de cette propriété affecte l'animation courante.

Exemple

L'instruction suivante permet à l'utilisateur d'arrêter le préchargement des acteurs en appuyant sur des touches ou en cliquant avec la souris :

```
-- Syntaxe Lingo
_movie.preLoadEventAbort = TRUE

// Syntaxe JavaScript
_movie.preLoadEventAbort = true;
```

Voir aussi

`Animation`

preLoadMode

Utilisation

```
-- Syntaxe Lingo
réfObjDistribution.preLoadMode

// Syntaxe JavaScript
réfObjDistribution.preLoadMode;
```

Description

Propriété de bibliothèque de distribution ; détermine le mode de préchargement d'une bibliothèque de distribution donnée. Lecture/écriture.

Les valeurs correctes de `preLoadMode` sont les suivantes :

- 0. Charger la bibliothèque de distribution au fur et à mesure. C'est la valeur par défaut.
- 1. Charger la bibliothèque de distribution avant l'image 1.
- 2. Charger la bibliothèque de distribution après l'image 1.

L'utilisation de cette propriété équivaut à définir la commande de chargement de la distribution de la boîte de dialogue Propriétés de la distribution.

Exemple

L'instruction suivante demande à Director de charger les acteurs de la distribution Boutons avant que l'animation n'entre dans l'image 1 :

```
-- Syntaxe Lingo
castLib("Boutons").preLoadMode = 1

// Syntaxe JavaScript
castLib("Boutons").preLoadMode = 1;
```

Voir aussi

Bibliothèque de distribution

preLoadRAM

Utilisation

```
the preLoadRAM
```

Description

Propriété système ; spécifie la quantité de mémoire RAM pouvant être utilisée pour le préchargement d'une vidéo numérique. Cette propriété peut être définie et testée.

Cette propriété est utile pour la gestion de la mémoire, puisqu'elle limite les acteurs vidéo numérique à une certaine quantité de mémoire, pour permettre le préchargement d'autres types d'acteurs. Lorsque `preLoadRAM` reçoit la valeur `FALSE`, toute la mémoire disponible peut être utilisée pour le préchargement des acteurs vidéo numérique.

Il n'est cependant pas possible de prédire de manière fiable la RAM qu'une vidéo numérique préchargée va exiger, la mémoire requise étant affectée par le contenu de l'animation, le taux de compression utilisé, le nombre d'images-clés, la modification des images, etc.

Il est généralement possible de précharger sans crainte les deux ou trois premières secondes d'une vidéo puis de continuer la lecture en flux continu à partir de cet endroit.

Si vous connaissez le taux de transfert de votre animation, vous pouvez estimer le paramétrage de `preLoadRAM`. Par exemple, si votre animation utilise un taux de transfert de 300 Ko par seconde, réglez `preLoadRAM` sur 600 Ko pour précharger les 2 premières secondes du fichier vidéo. Même s'il ne s'agit que d'une estimation, elle convient dans la plupart des situations.

Exemple

L'instruction suivante règle `preLoadRAM` sur 600 Ko, pour précharger les 2 premières secondes d'une animation dont le taux de transfert est de 300 Ko par seconde :

```
set the preLoadRAM = 600
```

Voir aussi

loop (mot-clé), next

preLoadTime

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.preLoadTime

// Syntaxe JavaScript
réfObjActeur.preLoadTime;
```

Description

Propriété d'acteur et de piste audio ; pour les acteurs, spécifie la quantité (en secondes) de l'acteur Shockwave Audio (SWA) en flux continu à télécharger avant que sa lecture ne commence ou, lors de l'utilisation d'une commande `preLoadBuffer`. La valeur par défaut est 5 secondes.

Cette propriété ne peut être définie que si l'acteur SWA lu en flux continu est arrêté.

Pour les pistes audio, la valeur concerne le son défini dans la file d'attente ou le son courant, si aucun son n'est spécifié.

Exemple

Le gestionnaire suivant définit à 6 secondes le temps de préchargement de l'acteur SWA Louis Armstrong lu en flux continu. Le préchargement réel se produit lorsqu'une commande `preLoadBuffer` ou `play` est émise.

```
-- Syntaxe Lingo
on mouseDown
  member("Louis Armstrong").stop()
  member("Louis Armstrong").preLoadTime = 6
end

// Syntaxe JavaScript
function mouseDown() {
  member("Louis Armstrong").stop();
  member("Louis Armstrong").preLoadTime = 6;
}
```

L'instruction suivante renvoie la valeur `preLoadTime` de l'acteur son lu sur la piste 1.

```
-- Syntaxe Lingo
put sound(1).preLoadTime

// Syntaxe JavaScript
trace(sound(1).preLoadTime);
```

Voir aussi

`preLoadBuffer()`

primitives

Utilisation

```
getRendererServices().primitives
```

Description

Fonction 3D ; renvoie une liste des types de primitives qui peuvent être utilisés pour créer de nouvelles ressources de modèle.

Exemple

L'instruction suivante affiche les types de primitives disponibles.

```
put getRendererServices().primitives
-- [#sphere, #box, #cylinder, #plane, #particle]
```

Voir aussi

getRendererServices(), newModelResource

productName

Utilisation

```
-- Syntaxe Lingo
_player.productName

// Syntaxe JavaScript
_player.productName;
```

Description

Propriété de lecteur ; renvoie le nom de l'application Director. En lecture seule.

Exemple

L'instruction suivante affiche, dans la fenêtre Messages, le nom de l'application Director.

```
-- Syntaxe Lingo
trace(_player.productName)

// Syntaxe JavaScript
trace(_player.productName);
```

Voir aussi

Lecteur

productVersion

Utilisation

```
-- Syntaxe Lingo
_player.productVersion

// Syntaxe JavaScript
_player.productVersion;
```

Description

Propriété de lecteur ; renvoie le numéro de version de l'application Director. En lecture seule.

Exemple

L'instruction suivante affiche, dans la fenêtre Messages, la version de l'application Director.

```
-- Syntaxe Lingo
trace(_player.productVersion)

// Syntaxe JavaScript
trace(_player.productVersion);
```

Voir aussi

Lecteur

projection

Utilisation

```
sprite(quelleImageObjet).camera.projection
camera(quelleCaméra).projection
member(quelActeur).camera(quelleCaméra).projection
```

Description

Propriété 3D ; permet d'obtenir ou de définir le style de projection de la caméra. Les valeurs possibles sont #perspective (la valeur par défaut) et #orthographic.

Lorsque la projection a pour valeur #perspective, les objets les plus proches de la caméra apparaissent plus grands que les objets les plus éloignés, et les propriétés projectionAngle ou fieldOfView spécifient l'angle de projection verticale (ce qui détermine l'espace visible de l'univers). L'angle de projection horizontale est déterminé par le rapport hauteur/largeur de la propriété rect de la caméra.

Lorsque la projection a pour valeur #orthographic, la taille apparente des objets ne dépend pas de la distance de la caméra et la propriété orthoHeight spécifie le nombre d'unités d'univers qui logent verticalement dans l'image-objet (ce qui détermine l'espace visible de l'univers). La largeur de la projection orthographique est déterminée par le rapport hauteur/largeur de la propriété rect de la caméra.

Exemple

L'instruction suivante donne à la propriété de projection de la caméra de l'image-objet 5 la valeur #orthographic.

```
sprite(5).camera.projection = #orthographic
```

Voir aussi

fieldOfView (3D), orthoHeight, fieldOfView (3D)

purgePriority

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.purgePriority

// Syntaxe JavaScript
réfObjActeur.purgePriority;
```

Description

Propriété d'acteur ; spécifie la priorité de purge d'un acteur. Lecture/écriture.

La priorité de purge d'un acteur détermine l'ordre de priorité suivi par Director pour sélectionner les acteurs à supprimer de la mémoire lorsque celle-ci est saturée. Plus la priorité de purge est élevée, plus il est probable que l'acteur sera supprimé. Les valeurs suivantes sont disponibles pour `purgePriority` :

- 0 – Jamais
- 1 – Dernier
- 2 – Suivant
- 3 – Normale (valeur par défaut)

Le paramètre Normale permet à Director de purger la mémoire des acteurs d'une manière aléatoire. Les paramètres Suivant, Dernier et Jamais permettent de contrôler plus ou moins la purge. Cependant, si vous réglez les paramètres Dernier ou Jamais pour de nombreux acteurs, votre animation risque de se trouver à court de mémoire.

En réglant la propriété `purgePriority` des acteurs, vous serez à même de gérer la mémoire lorsque la taille de la bibliothèque de distribution de l'animation dépassera la mémoire disponible. En règle générale, vous pouvez minimiser les pauses pendant que l'animation charge les acteurs et réduire le nombre de nouveaux chargements d'acteurs que Director doit exécuter en affectant une faible priorité de purge aux acteurs fréquemment utilisés au cours de l'animation.

Exemple

L'instruction suivante affecte la valeur 3 à la priorité de purge de l'acteur Arrière-plan, ce qui signifie qu'il sera l'un des premiers acteurs à être purgés lorsque Director aura besoin de mémoire :

```
-- Syntaxe Lingo
member("Arrière-plan").purgePriority = 3

// Syntaxe JavaScript
member("Arrière-plan").purgePriority = 3;
```

Voir aussi

Acteur

quad

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.quad

// Syntaxe JavaScript
réfObjImageObjet.quad;
```

Description

Propriété d'image-objet ; contient une liste de quatre points, qui sont les valeurs flottantes servant à décrire les coins d'une image-objet sur la scène. Lecture/écriture.

Les points du quadrilatère sont organisés dans l'ordre suivant : supérieur gauche, supérieur droit, inférieur droit et inférieur gauche.

Les points eux-mêmes peuvent être manipulés pour obtenir des effets de perspective et autres distorsions des images.

Si vous manipulez le quadrilatère d'une image-objet, vous pouvez lui redonner les valeurs du scénario en désactivant l'image-objet à laquelle un script est affecté au moyen de `puppetSprite (numImageObjetInt, FALSE)`. La désactivation du quadrilatère d'une image-objet interdit d'autre part de la faire pivoter ou de l'incliner.

Exemple

L'instruction suivante affiche une liste type décrivant une image-objet :

```
-- Syntaxe Lingo
put(sprite(1).quad)

// Syntaxe JavaScript
put(sprite(1).quad);
```

Lorsque vous modifiez la propriété d'image-objet `quad`, vous devez rétablir la liste de points si vous modifiez l'une de ces valeurs. En effet, lorsque vous affectez la valeur d'une propriété à une variable, vous placez une copie de la liste, et non la liste elle-même, dans la variable. Pour appliquer un changement, utilisez une syntaxe comme (Lingo uniquement) :

```
-- Syntaxe Lingo
listeQuadCourante = sprite(5).quad
listeQuadCourante[1] = listeQuadCourante[1] + point(50, 50)
sprite(5).quad = listeQuadCourante
```

Voir aussi

`point()`, `puppetSprite()`, `Image-objet`

quality

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.quality

// Syntaxe JavaScript
réfObjActeurOuImageObjet.quality;
```

Description

Propriété d'acteur et d'image-objet Flash ; contrôle si Director utilise l'anti-aliasing pour le rendu d'une image-objet animation Flash, produisant une haute qualité de rendu, mais aussi une lecture plus lente. La propriété `quality` peut prendre les valeurs suivantes :

- `#autoHigh` – Director commence par effectuer le rendu de l'image-objet avec l'anti-aliasing. Si la cadence d'images tombe en dessous de celle spécifiée pour l'animation, Director désactive l'anti-aliasing. Ce réglage donne priorité à la vitesse de lecture sur la qualité visuelle.
- `#autoLow` – Director commence par effectuer le rendu de l'animation sans anti-aliasing. Le lecteur Flash active l'anti-aliasing s'il détermine que le processeur de l'ordinateur est capable de le gérer. Ce réglage donne priorité à la qualité visuelle si cela est possible.
- `#high` (valeur par défaut) – L'animation est toujours lue avec anti-aliasing.
- `#low` – L'animation est toujours lue sans anti-aliasing.

La propriété `quality` peut être testée et définie.

Exemple

Le script d'image-objet suivant détermine le nombre de couleurs de l'ordinateur sur lequel l'animation est lue. Si le nombre de couleurs est réglé sur 8 bits ou moins (256 couleurs), le scénario règle la qualité de l'image-objet dans la piste 5 sur `#low`.

```
-- Syntaxe Lingo
on beginSprite me
  if _system.colorDepth <= 8 then
    sprite(1).quality = #low
  end if
end

// Syntaxe JavaScript
function beginSprite() {
  var clrDp = _system.colorDepth;
  if (clrDp <= 8) {
    sprite(1).quality = symbol("low");
  }
}
```

quality (3D)

Utilisation

```
member(quelActeur).texture(quelleTexture).quality
member(quelActeur).shader(quelMatériau).texture\
  (quelleTexture).quality
member(quelActeur).model(quelModèle).shader.texture\
  (quelleTexture).quality
member( quelActeur ).model( quelModèle ).\
  shader.texturelist[IndexDeListeDeTextures].quality
member(quelActeur).model(quelModèle).shaderList\
  [indexDeListeDeMatériaux]. texture(quelleTexture).quality
member( quelActeur ).model( quelModèle ).shaderList\
  [ indexDeListeDeMatériaux ]. texturelist[ IndexDeListeDeTextures ].quality
```

Description

Propriété de texture 3D ; permet d'obtenir ou de définir la qualité d'image d'une texture en contrôlant le niveau de mipmapping qui lui est appliqué. Le mipmapping est un processus qui crée des versions supplémentaires de l'image de texture, créées de tailles variées et plus petites que l'image d'origine. L'Xtra 3D affiche ensuite à l'écran la version la plus appropriée de l'image en fonction de la taille courante du modèle et change la version de l'image utilisée selon les besoins. Le mipmapping trilinéaire produit une qualité supérieure au mipmapping bilinéaire mais demande aussi plus de mémoire.

Le mipmapping est différent du filtrage, bien que les deux processus améliorent l'apparence de la texture. Le filtrage répartit les erreurs sur l'ensemble de la texture pour qu'elles soient moins concentrées. Le mipmapping rééchantillonne l'image pour lui donner la taille appropriée.

Cette propriété peut avoir les valeurs suivantes :

- `#low` correspond à la désactivation, le mipmapping n'étant pas utilisé pour la texture.
- `#medium` définit un mipmapping de faible qualité (bilinéaire) pour la texture.
- `#high` définit un mipmapping de qualité élevée (trilinéaire) pour la texture.

La valeur par défaut est `#low`.

Exemple

L'instruction suivante donne à la propriété `quality` de la texture de placageMars la valeur `#medium`.

```
member("Séquence").texture("placageMars").quality = #medium
```

Voir aussi

`nearFiltering`

radius

Utilisation

```
référenceDobjetDeRessDeModèle.radius
member(quelActeur).modelResource(quelleRessourceDeModèle).radius
```

Description

Propriété de modèle 3D ; utilisée avec une ressource de modèle de type `#sphere` ou `#cylinder`, cette propriété permet d'obtenir ou de définir le rayon du modèle.

La propriété `radius` détermine le rayon de balayage utilisé pour générer la ressource de modèle. La valeur de cette propriété doit toujours être supérieure à `0.0` et est définie par défaut à 25.0.

Exemple

L'exemple suivant indique que le rayon de la ressource de modèle sphère01 est 24.0.

```
put member("Univers 3D").modelResource("sphère01").radius
-- 24.0
```

randomSeed

Utilisation

```
the randomSeed
```

Description

Propriété système ; spécifie la valeur de départ utilisée pour générer des nombres aléatoires obtenus au moyen de la fonction `random()`.

L'utilisation de la même valeur de départ produit la même séquence de nombres aléatoires. Cette propriété peut être utile pour le débogage au cours du développement. L'utilisation de la propriété `ticks` facilite la production d'une valeur de départ aléatoire unique étant donné qu'il est peu probable que la valeur de `ticks` soit répétée dans les utilisations suivantes.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante affiche le nombre de départ aléatoire dans la fenêtre Messages :

```
put the randomSeed
```

Voir aussi

`random()`, `milliseconds`

recordFont

Utilisation

```
recordFont(quelActeur, police {[,type]} {,[tailleDesBitmaps]}
  {,sousEnsembleDeCaractères} {, nouveauNom})
```

Description

Commande ; inclut une police TrueType ou Type 1 comme acteur. Une fois incluses, ces polices sont disponibles à l'auteur tout comme les autres polices installées sur le système.

Vous devez créer un acteur police vide à l'aide de la commande `new()` avant d'utiliser `recordFont`.

- *`police`* – Nom de la police d'origine à enregistrer.
- *`style`* – Liste de symboles indiquant le style de la police d'origine, les valeurs possibles étant `#plain`, `#bold`, `#italic`. Si vous ne définissez aucune valeur pour cet argument, c'est `#plain` qui sera utilisé par défaut.

- *tailleDesBitmaps* – Liste d'entiers spécifiant les tailles pour lesquelles les bitmaps sont enregistrés. Cet argument peut être vide. Si vous omettez cet argument, aucun bitmap n'est généré. Ces bitmaps donnent généralement de meilleurs résultats pour les petites tailles (inférieures à 14 points), mais occupent plus de mémoire.
- *sousEnsembleDeCaractères* – Chaîne de caractères à coder. Seuls les caractères spécifiés seront disponibles dans la police. Si cet argument est fourni, tous les caractères qu'il contient sont encodés. Si seuls certains caractères sont codés mais qu'un caractère non codé est utilisé, ce caractère apparaît comme une case vide.
- *nouveauNom* – Une chaîne utilisée comme nom de l'acteur police nouvellement enregistré.

La commande crée une police shockée dans *quelActeur* en utilisant la police nommée dans l'argument *police*. La valeur renvoyée par la commande indique si l'opération a réussi. La valeur zéro indique que l'opération a réussi.

Exemple

L'instruction suivante crée une police shockée simple n'utilisant que les deux arguments pour l'acteur et la police à enregistrer :

```
monNouvelActeurPolice = new(#font)
recordFont(monNouvelActeurPolice, "Module lunaire")
```

L'instruction suivante spécifie les tailles de bitmaps à générer et les caractères pour lesquels les données de police doivent être créées :

```
monNouvelActeurPolice = new(#font)
recordfont(monNouvelActeur,"Module lunaire", [], [14, 18, 45], "Nom du
  meilleur score du jeu Module lunaire")
```

Remarque : `recordFont` resynthétisant les données de la police au lieu de les utiliser directement, la distribution des polices shockées n'est soumise à aucune restriction légale.

Voir aussi

`new()`

rect (caméra)

Utilisation

`sprite(`*quelleImageObjet*`).camera(`*quelleCaméra*`).rect`

Description

Propriété 3D de caméra ; permet d'obtenir ou de définir le rectangle qui contrôle la taille et la position de la caméra. Ce rectangle est analogue à celui que vous observez dans le viseur d'une caméra réelle.

La valeur par défaut de la propriété `rect` de toutes les caméras est `rect(0,0,1,1)` qui les rend invisibles jusqu'à la modification du paramètre. Cependant, lorsque `sprite.camera(1)` est rendu, son `rect` est réinitialisé à `rect(0, 0, sprite(`*quelleImageObjet*`).width, sprite(`*quelleImageObjet*`).height)` de façon à ce que la caméra remplisse l'écran. Toutes les coordonnées de cadre de caméra sont calculées par rapport au coin supérieur gauche de l'image-objet.

Si la valeur de *quelleCaméra* est supérieure à 1, le `rect` n'est pas redimensionné en même temps que l'image-objet et il sera donc nécessaire, si vous le souhaitez, de gérer ce redimensionnement au moyen d'un script.

Lorsque la valeur de *quelleCaméra* est supérieure à 1, les valeurs des propriétés `rect.top` et `rect.left` doivent être supérieures ou égales aux valeurs `rect.top` et `rect.left` de `sprite.camera(1)`.

Exemple

L'instruction suivante donne au rectangle de la caméra par défaut de l'image-objet 5 la valeur `rect(0, 0, 200, 550)` :

```
sprite(5).camera.rect = rect(0, 0, 200, 550)
```

Voir aussi

`cameraPosition`, `cameraRotation`

rect (image)

Utilisation

```
-- Syntaxe Lingo
réfObjImage.rect

// Syntaxe JavaScript
réfObjImage.rect;
```

Description

Propriété d'image. Renvoie un rectangle décrivant la taille d'une image donnée. En lecture seule.

Les coordonnées du rectangle renvoyé sont calculées par rapport au coin supérieur gauche de l'image. Les valeurs gauche et supérieure du rectangle s'élèvent donc à 0 et les valeurs inférieure et droite constituent la largeur et la hauteur de l'acteur.

Exemple

L'instruction suivante affiche le rectangle de l'acteur Lever de soleil de 300 x 400 pixels dans la fenêtre Messages :

```
-- Syntaxe Lingo
member("lever de soleil").image.rect -- rect(0, 0, 300, 400)

 // Syntaxe JavaScript
member("lever de soleil").image.rect; // rect(0, 0, 300, 400)
```

L'instruction suivante examine les 50 premiers acteurs et affiche le rectangle et le nom de chaque acteur bitmap :

```
-- Syntaxe Lingo
on showAllRects
  repeat with x = 1 to 50
    if member(x).type = #bitmap then
      put member(x).image.rect && "-" && member(x).name
    end if
  end repeat
end
```

```
// Syntaxe JavaScript
function showAllRects() {
  var x = 1;
  while(x < 51) {
    var tp = member(x).type.toString();
    if (tp == "#bitmap") {
      trace(member(x).image.rect + " - " + member(x).name);
      i++;
    }
  }
}
```

Voir aussi

height, image(), width

rect (acteur)

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.rect

// Syntaxe JavaScript
réfObjActeur.rect;
```

Description

Propriété d'acteur ; spécifie les coordonnées gauche, supérieure, droite et inférieure, sous la forme d'un rectangle, du rectangle de n'importe quel acteur graphique, tel qu'un bitmap, une forme, une animation ou une vidéo numérique. Lecture seule pour tous les acteurs, lecture/écriture pour les acteurs champ uniquement.

Pour un bitmap, la propriété `rect` est mesurée à partir du coin supérieur gauche du bitmap, et non à partir du coin supérieur gauche du bord de la fenêtre Dessin.

Pour un acteur Xtra, la propriété `rect` est un rectangle dont le coin supérieur gauche est à (0,0).

Exemple

L'instruction suivante affiche les coordonnées de l'acteur bitmap 20 :

```
-- Syntaxe Lingo
put(member(20).rect)

// Syntaxe JavaScript
put(member(20).rect);
```

L'instruction suivante définit les coordonnées de l'acteur bitmap Bandeau :

```
-- Syntaxe Lingo
member("Bandeau").rect = rect(100, 150, 300, 400)

// Syntaxe JavaScript
member("Bandeau").rect = rect(100, 150, 300, 400);
```

Voir aussi

Acteur

rect (image-objet)

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.rect

// Syntaxe JavaScript
réfObjImageObjet.rect;
```

Description

Propriété d'image-objet ; spécifie les coordonnées gauche, supérieure, droite et inférieure, sous la forme d'un rectangle, du rectangle de n'importe quel image-objet graphique, tel qu'un bitmap, une forme, une animation ou une vidéo numérique. Lecture/écriture.

Exemple

L'instruction suivante affiche les coordonnées de l'image-objet bitmap 20 :

```
-- Syntaxe Lingo
put(sprite(20).rect)

// Syntaxe JavaScript
put(sprite(20).rect);
```

Voir aussi

rect(), Image-objet

rect (fenêtre)

Utilisation

```
-- Syntaxe Lingo
réfObjFenêtre.rect

// Syntaxe JavaScript
réfObjFenêtre.rect;
```

Description

Propriété de fenêtre ; spécifie les coordonnées gauche, supérieure, droite et inférieure, renvoyées sous la forme d'un rectangle, d'une fenêtre. Lecture/écriture.

Si la taille du rectangle spécifié est inférieure à celle de la scène sur laquelle l'animation a été créée, l'animation est recadrée dans la fenêtre, mais n'est pas mise à l'échelle.

Pour effectuer un panoramique ou une mise à l'échelle de l'animation lue dans la fenêtre, modifiez la propriété `drawRect` ou `sourceRect` de la fenêtre.

Exemple

L'instruction suivante affiche les coordonnées de la fenêtre Tableau_de_commande :

```
-- Syntaxe Lingo
put(window("Tableau_de_commande").rect)

// Syntaxe JavaScript
put(window("Tableau_de_commande").rect);
```

Voir aussi

drawRect, sourceRect, Fenêtre

ref

Utilisation

expressionSousChaîne.ref

Description

Propriété d'expression de sous-chaîne de texte ; offre un raccourci pratique permettant de faire référence à une expression de sous-chaîne dans un acteur texte.

Exemple

En l'absence de références, vous devriez utiliser une instruction telle que :

```
member(acteurTexte).line[ligne].word[premMot..derMot].font = "Palatino"
member(acteurTexte).line[ligne].word[premMot..derMot].fontSize = 36
member(acteurTexte).line[ligne].word[premMot..derMot].fontStyle = [#bold]
```

Avec une propriété `ref`, en revanche, vous pouvez faire référence à la même sous-chaîne, par exemple :

```
maRéf = member(acteurTexte).line[ligne].word[premMot..derMot].ref
```

La variable `maRéf` est maintenant un raccourci pour toute l'expression de sous-chaîne. Cela permet quelque chose comme :

```
put maRéf.font
-- "Palatino"
```

Vous pouvez aussi définir une propriété de la sous-chaîne avec :

```
maRéf.fontSize = 18
maRéf.fontStyle = [#italic]
```

Vous pouvez accéder à la chaîne indiquée par la référence en utilisant sa propriété texte :

```
put maRéf.text
```

Cela produirait les données réelles de la chaîne et non les informations la concernant.

reflectionMap

Utilisation

```
member(quelActeur).shader(quelMatériau).reflectionMap
```

Description

Propriété 3D de matériau ; permet d'obtenir et de définir la texture utilisée pour créer des reflets à la surface d'un modèle. Cette texture est appliquée à la troisième couche de texture du matériau. Cette propriété est ignorée si le modificateur `toon` est appliqué à la ressource de modèle.

Cette propriété fournit une interface plus simple pour la définition d'une utilisation commune du placage de réflexion. Les propriétés suivantes produisent le même effet :

```
shader.textureModeList[3] = #reflection
shader.blendFunctionList[3] = #blend
shader.blendSourceList[3] = #constant
shader.blendConstantList[3] = 50.0
```

Cette propriété, lorsque testée, renvoie la texture associée à la troisième couche de texture du modèle. La valeur par défaut est `void`.

Exemple

L'instruction suivante entraîne le reflet de la texture Portrait sur la surface du modèle sphèreEnVerre.

```
member("planète3D").model("sphèreEnVerre").shader.reflectionMap = \
  member("planète3D").texture("Portrait")
```

Voir aussi

`textureModeList`, `blendFunctionList`, `blendConstantList`

reflectivity

Utilisation

```
member(quelActeur).reflectivity
```

Description

Propriété 3D de matériau ; permet d'obtenir ou de définir le niveau de brillant du matériau par défaut de l'acteur référencé. Cette valeur à virgule flottante représente le pourcentage, de 0.0 à 100.00, de lumière à refléter sur la surface d'un modèle utilisant le matériau par défaut. La valeur par défaut est 0.0.

Exemple

L'instruction suivante définit le degré de brillant du matériau par défaut à 50 % pour l'acteur Séquence :

```
member("Séquence").reflectivity = 50
```

region

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  emitter.region
référenceDobjetDeRessourceDeModèle.emitter.region
```

Description

Propriété 3D d'émission ; utilisée avec une ressource de modèle de type #particle, cette propriété permet d'obtenir et de définir la propriété region de l'émetteur de particules de la ressource.

Cette propriété de région définit la position à partir de laquelle les particules sont émises. Si sa valeur est un seul vecteur, ce vecteur en question est utilisé pour définir un point à partir duquel les particules vont être émises dans l'univers 3D.

Si sa valeur est une liste de deux vecteurs, ces vecteurs sont utilisés pour définir les points d'extrémité d'un segment de ligne à partir duquel les particules vont être émises.

Si sa valeur est une liste de quatre vecteurs, ces vecteurs sont utilisés pour définir les sommets du quadrilatère à partir duquel les particules vont être émises.

La valeur par défaut de cette propriété est [vector(0,0,0)].

Exemple

Dans l'exemple suivant, systèmeThermique est une ressource de modèle de type #particle. L'instruction suivante spécifie les quatre coins du rectangle d'où proviennent les particules de systèmeThermique.

```
member("Feux").modelResource("systèmeThermique").emitter.region = \
  [vector(20,90,100), vector(30,90,100), vector(30,100,100), \
  vector(20,100,100)]
```

Voir aussi

emitter

regPoint

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.regPoint

// Syntaxe JavaScript
réfObjActeur.regPoint;
```

Description

Propriété d'acteur ; spécifie le point d'alignement d'un acteur. Lecture/écriture.

Le point d'alignement est listé en tant que coordonnées horizontale et verticale d'un point sous la forme point(*horizontal, vertical*). Les acteurs non affichés, tels que les sons, ne possèdent pas de propriété regPoint utile.

Vous pouvez utiliser la propriété regPoint pour animer des graphiques individuels dans une boucle, ce qui change la position de la boucle par rapport à d'autres objets de la scène.

Vous pouvez aussi utiliser regPoint pour ajuster la position d'un masque sur une image-objet.

Lorsqu'un acteur animation Flash est initialement inséré dans la bibliothèque de distribution, son point d'alignement est son centre et sa propriété `centerRegPoint` a pour valeur `TRUE`. Si vous utilisez plus tard la propriété `regPoint` pour repositionner le point d'alignement, la propriété `centerRegPoint` est automatiquement définie comme `FALSE`.

Exemple

L'instruction suivante affiche le point d'alignement de l'acteur bitmap Bureau dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(member("Bureau").regPoint)

// Syntaxe JavaScript
put(member("Bureau").regPoint);
```

L'instruction suivante change le point d'alignement de l'acteur bitmap Bureau en fonction des valeurs de la liste :

```
-- Syntaxe Lingo
member("Bureau").regPoint = point(300, 400)

// Syntaxe JavaScript
member("Bureau").regPoint = point(300, 400);
```

Voir aussi

`Acteur`, `Image-objet`

regPoint (3D)

Utilisation

```
sprite(quelleImageObjet).camera.backdrop[indexDeFond].regPoint
member(quelActeur).camera(quelleCaméra).backdrop
[indexDeFond].regPoint
```

Description

Propriété 3D de fond et de recouvrement ; permet d'obtenir ou de définir le point d'alignement du fond ou du recouvrement. Le point d'alignement représente les coordonnées *x*, *y* et *z* du centre du fond ou du recouvrement dans l'espace 3D. La valeur par défaut de cette propriété est `point(0,0)`.

Exemple

L'instruction suivante modifie le point d'alignement du premier fond de la caméra de l'image-objet 13. Le point d'alignement du fond sera le point point(50, 0), mesuré à partir du coin supérieur gauche du fond.

```
sprite(13).camera.backdrop[1].regPoint = point(50, 0)
```

Voir aussi

`loc (fond et recouvrement)`

regPointVertex

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.regPointVertex

// Syntaxe JavaScript
réfObjActeur.regPointVertex;
```

Description

Propriété d'acteur ; indique si un sommet de *acteurVecteur* est utilisé comme point d'alignement pour cet acteur. Si la valeur est égale à zéro, le point d'alignement est déterminé normalement, à l'aide des propriétés `centerRegPoint` et `regPoint`. Si la valeur est différente de zéro, il indique la position dans la liste `vertextList` du sommet utilisé comme point d'alignement. La propriété `centerRegPoint` est définie sur `FALSE` et la propriété `regPoint` est définie sur l'emplacement de ce sommet.

Exemple

L'instruction suivante fait correspondre le point d'alignement de l'acteur forme vectorielle Quelconque à l'emplacement du troisième sommet :

```
-- Syntaxe Lingo
member("quelconque").regPointVertex=3

// Syntaxe JavaScript
member("quelconque").regPointVertex=3;
```

Voir aussi

`centerRegPoint`, `regPoint`

renderer

Utilisation

```
getRendererServices().renderer
```

Description

Propriété 3D ; permet d'obtenir ou de définir le moteur de rendu actuellement utilisé par une animation. La plage des valeurs de cette propriété est déterminée par la liste des moteurs de rendu disponibles, renvoyée par la propriété `rendererDeviceList` de l'objet de services de rendu.

Shockwave Player permet de spécifier le moteur de rendu à l'aide du menu contextuel correspondant. Si l'utilisateur sélectionne l'option visant à respecter les paramètres de contenu, le moteur de rendu spécifié par les propriétés `renderer` ou `preferred3DRenderer` est utilisé pour dessiner l'animation (s'il est disponible sur le système de l'utilisateur) ; sinon, c'est le moteur de rendu sélectionné par l'utilisateur qui est utilisé.

La valeur par défaut de cette propriété est déterminée par la propriété `preferred3DRenderer`.

Cette propriété renvoie la même valeur que celle renvoyée par la propriété `the active3dRenderer`.

Exemple

L'instruction suivante indique que le moteur de rendu actuellement utilisé par le système de l'utilisateur est `#openGL` :

```
put getRendererServices().renderer
-- #openGL
```

Voir aussi

`getRendererServices()`, `preferred3dRenderer`, `rendererDeviceList`, `active3dRenderer`

rendererDeviceList

Utilisation

```
getRendererServices().rendererDeviceList
```

Description

Propriété 3D de moteur de rendu ; renvoie une liste de symboles identifiant les moteurs de rendu disponibles sur la machine cliente. Le contenu de cette liste détermine la plage des valeurs qui peuvent être spécifiées pour les propriétés `renderer` et `preferred3DRenderer`. Cette propriété peut être testée, mais pas définie.

Cette propriété est une liste contenant les valeurs possibles suivantes :

- `#openGL` spécifie les pilotes openGL d'accélération matérielle fonctionnant sur les plates-formes Macintosh et Windows.
- `#directX7_0` spécifie les pilotes DirectX 7 d'accélération matérielle fonctionnant uniquement sur les plates-formes Windows.
- `#directX5_2` spécifie les pilotes DirectX 5.2 d'accélération matérielle fonctionnant uniquement sur les plates-formes Windows.
- `#software` spécifie le moteur de rendu logiciel intégré à Director fonctionnant avec les plates-formes Macintosh et Windows.

Exemple

L'instruction suivante indique les moteurs de rendu disponibles sur le système courant.

```
put getRendererServices().rendererDeviceList
-- [#openGL, #software]
```

Voir aussi

`getRendererServices()`, `renderer`, `preferred3dRenderer`, `active3dRenderer`

renderFormat

Utilisation

```
member(quelActeur).texture(quelleTexture).renderFormat
member(quelActeur).texture[index].renderFormat
member(quelActeur).shader(quelMatériau).texture.renderFormat
member(quelActeur).model(quelModèle).shader.texture\
  .renderFormat
member(quelActeur).model(quelModèle).shader.textureList\
  [index].renderFormat
member(quelActeur).model(quelModèle).shaderList[index].\
  texture(quelleTexture).renderFormat
member(quelActeur).model(quelModèle).shaderList[index].\
  textureList[index].renderFormat
```

Description

Propriété 3D ; permet d'obtenir ou de définir la propriété `textureRenderFormat` pour une texture spécifique, en spécifiant l'une des valeurs suivantes :

`#default` utilise la valeur renvoyée par `getRendererServices().textureRenderFormat`.

```
#rgba8888
#rgba8880
#rgba5650
#rgba5550
#rgba5551
#rgba4444
```

Consultez `textureRenderFormat` pour plus d'informations sur ces valeurs.

La définition de cette propriété pour une texture individuelle remplace les paramètres définis à l'aide de `textureRenderFormat`.

La propriété `renderFormat` détermine le format de pixel utilisé par le moteur de rendu lorsqu'il effectue le rendu de la texture spécifiée. Chaque format de pixel est composé de chiffres, indiquant chacun le codage chromatique utilisé pour le rouge, le vert, le bleu et l'alpha. La valeur choisie détermine la précision des couleurs (y compris la précision du canal alpha facultatif) et donc la quantité de mémoire utilisée au niveau de la carte vidéo. Vous pouvez choisir une valeur qui améliore la fidélité des couleurs ou qui vous permet de mettre plus de textures en mémoire au niveau de la carte vidéo. Vous pouvez mettre à peu près deux fois plus de textures 16 bits que de textures 32 bits dans le même espace.

Exemple

L'instruction suivante donne à la propriété `renderFormat` de la texture ImageTexte la valeur `#rgba4444`. Les composants rouge, bleu, vert et alpha de la texture seront chacun dessinés en utilisant 4 bits d'information.

```
member("3d").texture("ImageTexte").renderFormat = #rgba4444
```

Voir aussi

`textureRenderFormat`, `getHardwareInfo()`

renderStyle

Utilisation

```
member(quelActeur).shader(quelMatériau).renderStyle
```

Description

Propriété 3D de matériau standard ; permet d'obtenir ou de définir la propriété `renderStyle` d'un matériau, tel que cela est déterminé par la géométrie de la ressource de modèle sous-jacente. Cette propriété a les valeurs suivantes :

#fill spécifie que le matériau remplit totalement la surface de la ressource de modèle.

#wire spécifie que le matériau n'apparaît qu'au bord des faces de la ressource de modèle.

#point spécifie que le matériau n'apparaît qu'aux sommets de la ressource de modèle.

Tous les matériaux ont accès aux propriétés `#standard` ; en plus de ces propriétés standard, les matériaux de type `#engraver`, `#newsprint` et `#painter` possèdent des propriétés uniques à leur type. Pour plus d'informations, consultez `newShader`.

Exemple

L'instruction suivante entraîne le rendu du matériau Mur seulement là où se trouvent les sommets de la ressource de modèle spécifiée.

```
member("Ville").shader("Mur").renderStyle = #point
```

resizable

Utilisation

```
-- Syntaxe Lingo
réfObjFenêtre.resizable

// Syntaxe JavaScript
réfObjFenêtre.resizable;
```

Description

Propriété de fenêtre ; indique si la fenêtre est redimensionnable (`TRUE`, par défaut) ou pas (`FALSE`). Lecture/écriture.

Exemple

Les instructions suivantes agrandissent la fenêtre Empire s'il s'agit d'une fenêtre redimensionnable.

```
-- Syntaxe Lingo
if (window("Empire").resizable == TRUE) then
  window("Empire").maximize()
end if

// Syntaxe JavaScript
if (window("Empire").resizable = true) {
  window("Empire").maximize();
}
```

Voir aussi

`Fenêtre`

resolution (3D)

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).resolution
```

Description

Propriété 3D ; permet d'obtenir ou de définir la propriété de résolution d'une ressource de modèle de type #sphere ou #cylinder.

La résolution contrôle le nombre de polygones utilisés pour générer la géométrie de la ressource de modèle. Une valeur plus élevée génère des polygones plus nombreux produisant une surface plus régulière. La valeur par défaut de cette propriété est 20.

Exemple

L'instruction suivante donne à la résolution de la ressource de modèle sphère01 la valeur 10.0.

```
member("Univers 3D").modelResource("sphère01").resolution = 10.0
```

resolution (DVD)

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.resolution

// Syntaxe JavaScript
réfObjDvd.resolution;
```

Description

Propriété DVD. Renvoie une liste de propriétés comportant la résolution source de l'axe des x (largeur) et de l'axe des y (hauteur). En lecture seule.

Les valeurs possibles pour l'axe des x sont 352, 704 et 720.

Les valeurs possibles pour l'axe des y sont 240, 480, 288 et 576.

Exemple

L'instruction suivante renvoie un exemple de liste de propriétés de résolutions :

```
-- Syntaxe Lingo
trace(member(1).resolution) -- [#width: 720, #height: 480]

// Syntaxe JavaScript
trace(member(1).resolution); // ["width": 720, "height": 480]
```

Voir aussi

DVD

resolve

Utilisation

```
member(quelActeur).model(quelModèle).collision.resolve
```

Description

Propriété 3D de collision ; permet de savoir ou de définir si les collisions sont résolues à la collision de deux modèles. Si cette propriété a pour valeur `TRUE` pour deux modèles impliqués dans une collision, tous deux s'arrêtent au point de collision. Si la propriété `resolve` d'un seul des modèles a la valeur `TRUE`, ce modèle s'arrête et l'autre modèle, dont la propriété est soit non définie soit `FALSE`, continue son mouvement. La valeur par défaut de cette propriété est `TRUE`.

Exemple

L'instruction suivante donne à la propriété `resolve` du modificateur de collision appliqué au modèle Boîte la valeur `TRUE`. Lorsque le modèle Boîte entre en collision avec un autre modèle auquel le modificateur `#collision` est associé, il s'arrête.

```
member("Univers 3D").model("Boîte").collision.resolve = TRUE
```

Voir aussi

`collisionData`, `collisionNormal`, `modelA`, `modelB`, `pointOfContact`

resource

Utilisation

```
member(quelActeur).model(quelModèle).resource
```

Description

Propriété 3D ; permet d'obtenir ou de définir la propriété de ressource qui définit la géométrie de la ressource de modèle référencée. Cette propriété permet aussi d'accéder à l'objet de ressource de modèle référencé et à ses propriétés associées.

Exemple

L'instruction suivante définit la ressource de modèle utilisée par le modèle nouvelleBoîte. Elle aura maintenant la même géométrie que le modèle Boîte.

```
member("Univers 3D").model("nouvelleBoîte").resource = member\
  ("Univers 3D").model("Boîte").resource
```

L'instruction suivante indique la propriété de résolution de la ressource de modèle utilisée par le modèle Cylindre.

```
put member("Univers 3D").model("Cylindre").resource.resolution
-- 20
```

right

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.right

// Syntaxe JavaScript
réfObjImageObjet.right;
```

Description

Propriété d'image-objet ; indique la distance, en pixels, séparant le bord droit d'une image-objet du bord gauche de la scène. Lecture/écriture.

Les coordonnées des images-objets sont exprimées par rapport au coin supérieur gauche de la scène.

Exemple

L'instruction suivante renvoie la distance du bord droit d'une image-objet :

```
-- Syntaxe Lingo
put(sprite(6).right)

// Syntaxe JavaScript
put(sprite(6).right);
```

Voir aussi

bottom, height, left, locH, locV, Image-objet, top, width

right (3D)

Utilisation

```
member(quelActeur).modelResource
(quelleRessourceDeModèle).right
référenceDobjetDeRessourceDeModèle.right
```

Description

Propriété 3D ; permet d'obtenir ou de définir la propriété `right` d'une ressource de modèle de type #box.

La propriété `right` détermine si le côté droit de la boîte est fermé (`TRUE`) ou ouvert (`FALSE`). La valeur par défaut est `TRUE`.

Exemple

L'instruction suivante donne à la propriété `right` de la ressource de modèle Caisse la valeur `TRUE`, ce qui signifie que le côté droit de la caisse sera fermé.

```
member("Univers 3D").modelResource("Caisse").right = TRUE
```

Voir aussi

bottom (3D), left (3D), top (3D)

rightIndent

Utilisation

expressionSousChaîne.rightIndent

Description

Propriété d'acteur texte ; contient la distance de décalage, en pixels, de la marge droite de *expressionSousChaîne* par rapport au côté droit de l'acteur texte.

La valeur est un nombre entier supérieur ou égal à 0.

Cette propriété peut être testée et définie.

Voir aussi

firstIndent, leftIndent

rightMouseDown

Utilisation

```
-- Syntaxe Lingo
_mouse.rightMouseDown

// Syntaxe JavaScript
_mouse.rightMouseDown;
```

Description

Propriété de souris ; indique si le bouton droit de la souris (Windows) ou le bouton de la souris et la touche Ctrl (Macintosh) sont enfoncés (TRUE) ou non (FALSE). En lecture seule.

Sur le Macintosh, rightMouseDown est TRUE uniquement si la propriété emulateMultiButtonMouse est elle-même TRUE.

Exemple

L'instruction suivante vérifie si le bouton droit de la souris (sous Windows) est enfoncé et, le cas échéant, lit le son Désolé dans la piste audio 2 :

```
-- Syntaxe Lingo
if (_mouse.rightMouseDown) then
  sound(2).play(member("Désolé"))
end if

// Syntaxe JavaScript
if (_mouse.rightMouseDown) {
  sound(2).play(member("Désolé"));
}
```

Voir aussi

emulateMultibuttonMouse, Souris

rightMouseUp

Utilisation

```
-- Syntaxe Lingo
_mouse.rightMouseUp

// Syntaxe JavaScript
_mouse.rightMouseUp;
```

Description

Propriété de souris ; indique si le bouton droit de la souris (Windows) ou le bouton de la souris et la touche Contrôle (Macintosh) sont relâchés (`TRUE`) ou enfoncés (`FALSE`). En lecture seule.

Sur le Macintosh, `rightMouseUp` n'a la valeur `TRUE` que si la propriété `emulateMultiButtonMouse` a elle-même la valeur `TRUE`.

Exemple

L'instruction suivante vérifie si le bouton droit de la souris (sous Windows) est relâché et, le cas échéant, lit le son Cliquez :

```
-- Syntaxe Lingo
if (_mouse.rightMouseUp) then
  sound(2).play(member("Cliquez"))
end if

// Syntaxe JavaScript
if (_mouse.rightMouseUp) {
  sound(2).play(member("Cliquez"));
}
```

Voir aussi

`emulateMultibuttonMouse`, `Souris`

romanLingo

Utilisation

```
the romanLingo
```

Description

Propriété système ; spécifie si Lingo utilise un interprète simple octet (`TRUE`) ou double octet (`FALSE`).

L'interprète Lingo est plus rapide avec un jeu de caractères simple octet. Certaines versions du logiciel système Macintosh (la version japonaise par exemple) utilisent un jeu de caractères double octet. En revanche, le système français utilise un jeu de caractères simple octet. Normalement, `romanLingo` est défini au démarrage de Director en fonction de la version locale du logiciel système.

Si vous utilisez un système double octet, mais n'utilisez aucun caractère double octet dans votre script, affectez la valeur `TRUE` à cette propriété afin d'accélérer l'exécution des scripts Lingo.

Exemple

L'instruction suivante affecte la valeur `TRUE` à `romanLingo`, ce qui entraîne Lingo à utiliser un jeu de caractères simple octet :

```
set the romanLingo to TRUE
```

Voir aussi

`inlineImeEnabled`

rootLock

Utilisation

```
member(quelActeur).model(quelModèle).keyframePlayer.rootLock
member(quelActeur).model(quelModèle).bonesPlayer.rootLock
```

Description

Propriété 3D de modificateur `#keyframePlayer` et `#bonesPlayer` ; indique si les composants de translation d'un mouvement sont utilisés (`FALSE`) ou ignorés (`TRUE`).

La valeur par défaut de cette propriété est `FALSE`.

Exemple

L'instruction suivante oblige le modèle Martien3 à garder sa position de départ tout en exécutant ses mouvements, ce qui résulte en un personnage marchant sur place.

```
member("nouveauMartien").model("Martien3").keyframePlayer.rootLock = 1
```

rootNode

Utilisation

```
member(quelActeur).camera(quelleCaméra).rootNode
sprite(quelleImageObjet).camera.rootNode
```

Description

Propriété 3D ; permet de connaître ou de définir les objets visibles dans une image-objet. Lors de la première création d'une caméra, elle présente tous les nœuds de l'univers. La propriété `rootNode` permet également de créer une autre vue par défaut qui limite l'affichage à un nœud spécifique et à ses enfants.

Supposons par exemple que la lumière C est un enfant du modèle A. Si vous donnez à la propriété `rootNode` la valeur `camera("defaultView").rootNode=model(A)`, l'image-objet affiche uniquement le modèle A éclairé par la lumière C. La valeur par défaut `group("world")`, indique que tous les nœuds sont utilisés.

Exemple

L'instruction suivante affecte le modèle Pluton à la propriété `rootNode` de la caméra de l'image-objet 5. Seuls Pluton et ses enfants seront visibles dans l'image-objet 5.

```
sprite(5).camera.rootNode = member("Séquence").model("Pluton")
```

rotation

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.rotation

// Syntaxe JavaScript
réfObjImageObjet.rotation;
```

Description

Propriété d'image-objet ; contrôle la rotation d'une image-objet animation QuickTime, GIF animé, animation Flash ou bitmap dans son rectangle de délimitation, sans faire pivoter ce rectangle ou le contrôleur de l'image-objet (dans le cas de QuickTime). Lecture/écriture.

En fait, le rectangle de délimitation de l'image-objet agit comme une fenêtre à travers laquelle vous pouvez voir l'animation Flash ou QuickTime. Le rectangle de délimitation d'un bitmap et d'un GIF animé change en fonction de la rotation de l'image.

La rotation du scénario ne fonctionne dans une animation Flash que si la propriété `obeyScoreRotation` est définie sur `TRUE`.

Une animation Flash pivote autour de son point d'origine spécifié par sa propriété `originMode`. Une animation QuickTime pivote autour du centre du rectangle de délimitation de l'image-objet. Un bitmap pivote autour du point d'alignement de l'image.

Pour un média QuickTime, si la propriété `crop` de l'image-objet a la valeur `TRUE`, une rotation fréquente de l'image-objet déplace une partie de l'image hors de la zone visible ; lorsque la propriété `crop` de l'image-objet a la valeur `FALSE`, l'image est mise à l'échelle pour tenir dans le rectangle de délimitation (ce qui peut provoquer une distorsion de l'image).

Vous spécifiez la rotation en degrés sous la forme d'un nombre à virgule flottante.

Le scénario peut conserver des informations sur la rotation d'une image de +21 474 836,47° à -21 474 836,48°, ce qui permet 59 652 rotations complètes dans chaque direction.

Lorsque la limite de rotation est atteinte (juste après la 59 652ème rotation), la commande rétablit le réglage sur +116,47° ou -116.48° – et non 0,00°. Cela s'explique par le fait que +21 474 836,47° est égal à +116,47° et -21 474 836,48° est égal à -116,48° (ou +243,52°). Pour éviter ce réglage, vous devez contraindre les angles à ±360° lorsque vous utilisez un script pour exécuter une rotation continue.

La valeur par défaut de cette propriété est 0.

Exemple

Le comportement suivant fait pivoter l'image-objet continuellement de 2° à chaque fois que la tête de lecture avance, tout en limitant l'angle à 360° :

```
-- Syntaxe Lingo
property spriteNum

on prepareFrame me
  sprite(spriteNum).rotation = integer(sprite(spriteNum).rotation + 2) mod 360
end
```

```
// Syntaxe JavaScript
function prepareFrame() {
  sprite(this.spriteNum).rotation = parseInt(sprite(this.spriteNum).rotation
  + 2) % 360;
}
```

Le script d'image suivant définit une boucle pour la tête de lecture sur l'image courante pendant qu'il fait pivoter une image-objet QuickTime dans la piste 5 de 360° par incréments de 16°. Lorsque l'image-objet a pivoté de 360°, la tête de lecture continue avec l'image suivante.

```
-- Syntaxe Lingo
on fairePivoterLanimation(quelleImageObjet)
  repeat with i = 1 to 36
    sprite(quelleImageObjet).rotation = i * 10
    _movie.updateStage()
  end repeat
end

// Syntaxe JavaScript
function fairePivoterLanimation(quelleImageObjet) {
  for (var i = 1; i <= 36; i++) {
    sprite(quelleImageObjet).rotation = i * 10;
    _movie.updateStage();
  }
}
```

Voir aussi

obeyScoreRotation, originMode, Image-objet

rotation (fond et recouvrement)

Utilisation

```
sprite(quelleImageObjet).camera.backdrop[indexDeFond].rotation
member(quelActeur).camera(quelleCaméra).backdrop
[indexDeFond].rotation
sprite(quelleImageObjet).camera.overlay[indexDeRecouvrement].rotation
member(quelActeur).camera[indexDeCaméra].overlay
[indexDeRecouvrement].rotation
```

Description

Propriété 3D ; permet de connaître ou de définir la rotation du fond ou du recouvrement vers la caméra par défaut. La valeur par défaut de cette propriété est 0.0.

Exemple

L'instruction suivante fait tourner un fond de 60° autour de son point d'alignement.

```
sprite(4).camera.backdrop[1].rotation = 60.0
```

Voir aussi

bevelDepth, transform (propriété)

rotation (matériau de gravure)

Utilisation

```
member(quelActeur).shader(quelMatériau).rotation
member(quelActeur).model(quelModèle).shader.rotation
member(quelActeur).model(quelModèle).shaderList[index].rotation
```

Description

Propriété 3D de matériau de gravure ; permet d'obtenir ou de définir un angle en degrés (exprimé sous forme de nombre à virgule flottante), qui décrit un décalage de rotation 2D pour les lignes gravées. La valeur par défaut de cette propriété est 0.0.

Exemple

L'instruction suivante fait pivoter de 1° les lignes utilisées pour dessiner le matériau de gravure du modèle gbCyl3.

```
member("séquence").model("gbCyl3").shader.rotation = \
  member("séquence").model("gbCyl3").shader.rotation + 1
```

Voir aussi

transform (propriété)

rotation (transformation)

Utilisation

```
member(quelActeur).node(quelNœud).transform.rotation
member(quelActeur).node(quelNœud).getWorldTransform().rotation
transformation.rotation
```

Description

Propriété 3D ; permet d'obtenir ou de définir le composant de rotation d'une transformation. Une transformation définit une échelle, une position et une rotation au sein d'un cadre de référence donné. La valeur par défaut de cette propriété est `vector(0,0,0)`.

Un nœud peut être un objet de caméra, groupe, lumière ou modèle. La définition de la `rotation` d'une transformation de nœud définit la rotation de cet objet dans le cadre de référence de la transformation. La définition de la propriété `rotation` de la transformation relative à l'univers d'un objet à l'aide de `getWorldTransform().rotation` définit la rotation de l'objet par rapport à l'origine de l'univers. La définition de la propriété `rotation` de la transformation relative au parent d'un objet à l'aide de `transform.rotation` définit la rotation de l'objet par rapport à son nœud parent.

Si vous souhaitez modifier l'orientation d'une transformation, il est recommandé d'utiliser les méthodes `rotate` et `prerotate` au lieu de définir cette propriété.

Exemple

L'instruction suivante donne à la rotation relative au parent de la première caméra de l'acteur la valeur `vector(0,0,0)`.

```
member("Espace").camera[1].transform.rotation = vector(0, 0, 0)
```

L'exemple suivant affiche la rotation par rapport au parent du modèle Lune, ajuste ensuite l'orientation du modèle à l'aide de la commande `rotate`, puis affiche la rotation résultante du modèle par rapport à l'univers.

```
put member("systèmeSolaire").model("Lune").transform.rotation
-- vector( 0.0000, 1.0000, 0.0000 )
member("systèmeSolaire").model("Lune").rotate(15,15,15)
put member("systèmeSolaire").model("Lune").getWorldTransform().rotation
-- vector( 16.5191, 51.3810, 65.8771 )
```

Voir aussi

`getWorldTransform()`, `preRotate`, `rotate`, `transform (propriété)`, `position (transformation)`, `scale (transformation)`

rotationReset

Utilisation

```
member(quelActeur).model(quelModèle).bonesPlayer.rotationReset
member(quelActeur).model(quelModèle).keyframePlayer.\
  rotationReset
```

Description

Propriété 3D de modificateur `keyframePlayer` et `bonesPlayer` ; indique les axes autour desquels les modifications de rotation sont conservées de la fin d'un mouvement au début du suivant, ou de la fin de l'itération d'un mouvement en boucle au début de l'itération suivante.

Les valeurs possibles de cette propriété sont `#none`, `#x`, `#y`, `#z`, `#xy`, `#yz`, `#xz` et `#all`. La valeur par défaut est `#all`.

Exemple

L'instruction suivante donne à la propriété `rotationReset` du modèle Monstre la valeur de l'axe des *z*. Le modèle conservera sa rotation autour de son axe des *z* lorsque le mouvement ou la boucle en cours d'exécution sera terminé.

```
member("NouveauMartien").model("Monstre").bonesPlayer.rotationReset = #z
```

Voir aussi

`positionReset`, `bonesPlayer (modificateur)`

RTF

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.RTF

// Syntaxe JavaScript
réfObjActeur.RTF;
```

Description

Propriété d'acteur ; permet l'accès au texte et aux balises contrôlant la mise en page du texte d'un acteur texte contenant du texte RTF.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante affiche dans la fenêtre Messages les informations de formatage RTF incluses dans l'acteur texte CV :

```
-- Syntaxe Lingo
put(member("CV").RTF)

// Syntaxe JavaScript
trace(member("CV").RTF);
```

Voir aussi

HTML, importFileInto()

safePlayer

Utilisation

```
-- Syntaxe Lingo
_player.safePlayer

// Syntaxe JavaScript
_player.safePlayer;
```

Description

Propriété de lecteur ; contrôle si les fonctions de sécurité de Director sont activées ou non. Lecture/écriture.

Dans une animation possédant un contenu Shockwave, cette propriété peut être testée, mais pas définie. Elle est toujours TRUE dans Shockwave Player.

Dans l'environnement auteur comme dans les projections, la valeur par défaut est FALSE. Cette propriété peut être renvoyée, mais ne peut être définie que sur TRUE. Une fois définie sur TRUE, elle ne peut plus être redéfinie sur FALSE sans redémarrer Director ou la projection.

Lorsque la valeur de safePlayer est TRUE, les fonctions de sécurité suivantes sont activées :

- Seuls les Xtras sûrs peuvent être utilisés.
- La propriété safePlayer ne peut pas être remise à zéro.
- Le collage du contenu du Presse-papiers avec la méthode pasteClipBoardInto() génère une boîte de dialogue d'avertissement et permet à l'utilisateur d'annuler l'opération.
- L'enregistrement d'une animation ou d'une distribution avec un script est désactivé.
- L'impression avec la méthode printFrom() est désactivée.
- L'ouverture d'une application avec la méthode open() est désactivée.
- La possibilité d'arrêter une application ou d'éteindre l'ordinateur de l'utilisateur avec les méthodes restart() ou shutDown() est désactivée.
- L'ouverture d'un fichier qui n'est pas dans le dossier DSWMedia est désactivée.
- La détermination d'un nom de fichier local est désactivée.
- L'utilisation de getNetText() ou postNetText() ou l'accès à une adresse URL ne possédant pas le même domaine que l'animation entraîne l'affichage d'une boîte de dialogue de sécurité.

Voir aussi

Lecteur

sampleCount

Utilisation

```
-- Syntaxe Lingo
réfObjPisteAudio.sampleCount

// Syntaxe JavaScript
réfObjPisteAudio.sampleCount;
```

Description

Propriété de piste audio ; spécifie le nombre de sons échantillonnés du son en cours dans une piste audio. En lecture seule.

Il s'agit du nombre total d'échantillons, qui dépend des propriétés `sampleRate` et `duration` du son. Ce nombre ne dépend pas de la propriété `channelCount` du son.

Un son de 1 seconde et de 44,1 KHz contient 44 100 échantillons.

Exemple

L'instruction suivante affiche le nom et la valeur `sampleCount` de l'acteur lu dans la piste audio 1 dans la fenêtre Messages :

```
-- Syntaxe Lingo
put("L'acteur son" && sound(1).member.name && "contient" && \
  sound(1).sampleCount && "échantillons.")

// Syntaxe JavaScript
put("L'acteur son" + sound(1).member.name + "contient" + \
  sound(1).sampleCount + "échantillons.");
```

Voir aussi

`sampleRate`, `Piste audio`

sampleRate

Utilisation

```
-- Syntaxe Lingo
réfObjPisteAudio.sampleRate

// Syntaxe JavaScript
réfObjPisteAudio.sampleRate;
```

Description

Propriété de piste audio ; renvoie, en échantillons par seconde, la fréquence d'échantillonnage de l'acteur son ou, dans le cas d'un son SWA, du fichier d'origine qui a été codé en Shockwave Audio. En lecture seule.

Cette propriété n'est disponible qu'après le début de la lecture du son SWA ou après le préchargement du fichier avec la méthode `preLoadBuffer()`. Lorsqu'une une piste audio est donnée, le résultat est le taux d'échantillonnage de l'acteur son en cours de lecture dans cette piste audio.

Des valeurs typiques de cette propriété sont 8000, 11025, 16000, 22050 et 44100.

Lorsque plusieurs sons sont placés en file d'attente dans une piste audio, Director les lit tous avec les valeurs `channelCount`, `sampleRate` et `sampleSize` du premier son en attente et effectue un nouvel échantillonnage des autres pour une lecture plus fluide. Director ne réinitialise ces propriétés qu'après le traitement des sons de la file d'attente ou sous l'intervention de la méthode `stop()`. Le prochain son placé en file d'attente ou lu déterminera ensuite les nouveaux paramètres.

Exemple

L'instruction suivante affecte la fréquence d'échantillonnage d'origine du fichier utilisé dans l'acteur SWA en flux continu Paul Robin à l'acteur champ Qualité audio :

```
-- Syntaxe Lingo
member("Qualité audio").text = string(member("Paul Robin").sampleRate)

// Syntaxe JavaScript
member("Qualité audio").text = member("Paul Robin").sampleRate.toString();
```

L'instruction suivante affiche le taux d'échantillonnage du son lu dans la piste audio 1 dans la fenêtre Messages :

```
-- Syntaxe Lingo
trace(sound(1).sampleRate)

// Syntaxe JavaScript
trace(sound(1).sampleRate);
```

Voir aussi

`channelCount`, `sampleSize`, `preLoadBuffer()`, `Piste audio`, `stop() (piste audio)`

sampleSize

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.sampleSize

// Syntaxe JavaScript
réfObjActeur.sampleSize;
```

Description

Propriété d'acteur ; détermine la taille d'échantillonnage de l'acteur spécifié. Le résultat est généralement une taille de 8 ou 16 bits. Si une piste audio est donnée, la valeur est celle de l'acteur son en cours de lecture dans la piste audio concernée.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante détermine la taille d'échantillonnage de l'acteur son Voix-off et affecte cette valeur à la variable `soundSize` :

```
-- Syntaxe Lingo
tailleDuSon = member("Voix-off").sampleSize

// Syntaxe JavaScript
var tailleDuSon = member("Voix-off").sampleSize;
```

L'instruction suivante affiche le taux d'échantillonnage du son lu dans la piste audio 1 dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(sound(1).sampleSize)

// Syntaxe JavaScript
put(sound(1).sampleSize);
```

scale (3D)

Utilisation

```
member(quelActeur).camera(quelleCaméra).backdrop\
  [indexDeFond].scale
member(quelActeur).camera(quelleCaméra).overlay\
  [indexDeRecouvrement].scale
```

Description

Propriété 3D ; permet d'obtenir ou de définir la valeur d'échelle utilisée par un recouvrement ou un fond spécifique dans la liste des recouvrements ou des fonds de la caméra référencée. La largeur et la hauteur du fond ou du recouvrement sont multipliées par la valeur de l'échelle. La valeur par défaut de cette propriété est 1.0.

Exemple

L'instruction suivante double la taille d'un fond.

```
sprite(25).camera.backdrop[1].scale = 2.0
```

Voir aussi

bevelDepth, overlay

scale (fond et recouvrement)

Utilisation

```
member(quelActeur).camera(quelleCaméra).backdrop\
  [indexDeFond].scale
member(quelActeur).camera(quelleCaméra).overlay\
  [indexDeRecouvrement].scale
```

Description

Propriété 3D ; permet d'obtenir ou de définir la valeur d'échelle utilisée par un recouvrement ou un fond spécifique dans la liste des recouvrements ou des fonds de la caméra référencée. La largeur et la hauteur du fond ou du recouvrement sont multipliées par la valeur de l'échelle. La valeur par défaut de cette propriété est 1.0.

Exemple

L'instruction suivante double la taille d'un fond.

```
sprite(25).camera.backdrop[1].scale = 2.0
```

Voir aussi

bevelDepth, overlay

scale (acteur)

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.scale

// Syntaxe JavaScript
réfObjActeurOuImageObjet.scale;
```

Description

Propriété d'acteur et d'image-objet ; contrôle la mise à l'échelle d'une image-objet animation QuickTime, forme vectorielle ou Flash.

Pour QuickTime, cette propriété ne met pas à l'échelle le rectangle de délimitation ou le contrôleur de l'image-objet. En revanche, elle met à l'échelle l'image autour de son centre dans le rectangle de délimitation. La mise à l'échelle est spécifiée sous forme d'une liste Director contenant deux pourcentages enregistrés en tant que nombres à virgule flottante :

`[pourcentageX, pourcentageY]`

Le paramètre *pourcentageX* spécifie la mise à l'échelle horizontale, le paramètre *pourcentageY* spécifiant la mise à l'échelle verticale.

Lorsque la propriété `crop` de l'image-objet est définie comme `TRUE`, la propriété `scale` permet de simuler un zoom dans le rectangle de délimitation de l'image-objet. Lorsque la propriété `crop` est définie comme `FALSE`, la propriété `scale` est ignorée.

Cette propriété peut être testée et définie. La valeur par défaut est `[1.0000,1.0000]`.

Pour les acteurs animation Flash ou forme vectorielle, la mise à l'échelle est exprimée sous la forme d'un nombre à virgule flottante. L'animation est mise à l'échelle depuis son point d'origine, comme spécifié par sa propriété `originMode`.

Remarque : Cette propriété doit avoir la valeur par défaut si la propriété `scaleMode` a pour valeur `#autoSize`. Sinon, l'image-objet n'est pas correctement affichée.

Exemple

Le gestionnaire suivant accepte une référence à une image-objet animation Flash comme paramètre, réduit l'animation à 0 % (de sorte qu'elle disparaisse), puis la met à l'échelle par incréments de 5 % jusqu'à sa taille normale (100 %).

```
-- Syntaxe Lingo
on miseAlEchelleDAnimation quelleImageObjet
  sprite(quelleImageObjet).scale = 0
  _movie.updatestage()
  repeat with i = 1 to 20
    sprite(quelleImageObjet).scale = i * 5
    _movie.updatestage()
  end repeat
end
```

```
// Syntaxe JavaScript
function miseAlEchelleDAnimation(quelleImageObjet) {
  sprite(quelleImageObjet).scale = 0;
  _movie.updatestage();
  var i = 1;
  while (i < 21) {
    sprite(quelleImageObjet).scale = i * 5;
    _movie.updatestage();
    i++;
  }
}
```

Voir aussi

scaleMode, originMode

scale (transformation)

Utilisation

```
member(quelActeur).node(quelNœud).transform.scale
member(quelActeur).node(quelNœud).getWorldTransform().scale
transformation.scale
```

Description

Propriété 3D ; permet d'obtenir ou de définir le composant de redimensionnement d'une transformation. Une transformation définit une échelle, une position et une rotation au sein d'un cadre de référence donné. La propriété `scale` permet d'obtenir ou de définir le degré de redimensionnement de la transformation pour chacun des trois axes. La valeur par défaut de cette propriété est `vector(1.0,1.0,1.0)`.

Un nœud peut être un objet de caméra, groupe, lumière ou modèle. Cette commande n'a aucun effet visuel sur les lumières ou les caméras car elles ne contiennent pas de géométrie. La définition de la propriété `position` d'une transformation de nœud définit le redimensionnement de cet objet le long des axes x, y et z, dans le cadre de référence de la transformation. L'obtention de la propriété `scale` de la transformation relative à l'univers d'un objet à l'aide de `getWorldTransform().scale` renvoie le redimensionnement de l'objet par rapport à l'origine de l'univers. La définition de la propriété `scale` de la transformation relative à l'univers d'un objet à l'aide de `transform.scale` définit le redimensionnement de l'objet par rapport à son nœud parent.

Exemple

L'instruction suivante donne à la propriété `scale` de la transformation du modèle Lune la valeur de `vector(2,5,3)`.

```
member("Séquence").model("Lune").transform.scale = vector(2,5,3)
```

Voir aussi

transform (propriété), getWorldTransform(), position (transformation), rotation (transformation), scale (commande)

scaleMode

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.scaleMode

// Syntaxe JavaScript
réfObjActeurOuImageObjet.scaleMode;
```

Description

Propriété d'acteur et d'image-objet ; contrôle la manière dont une animation Flash ou une forme vectorielle est mise à l'échelle dans le rectangle de délimitation d'une image-objet. Lorsque vous mettez à l'échelle une image-objet animation Flash en définissant ses propriétés `scale` et `viewScale`, elle n'est pas mise à l'échelle ; seule la vue de l'animation dans l'image-objet l'est. La propriété `scaleMode` peut prendre les valeurs suivantes :

- `#showAll` (valeur par défaut pour les animations Director antérieures à la version 7) – Conserve les proportions de l'acteur animation Flash d'origine. Si nécessaire, remplissez tout intervalle vide dans la dimension horizontale ou verticale avec la couleur d'arrière-plan.
- `#noBorder` – Conserve les proportions de l'acteur animation Flash d'origine. Si nécessaire, recadrez la dimension horizontale ou verticale.
- `#exactFit` – Ne conserve pas les proportions de l'acteur animation Flash d'origine. Etirez l'animation Flash pour lui donner les dimensions exactes de l'image-objet.
- `#noScale` – Conserve la taille d'origine du média Flash, quel que soit le mode de mise à l'échelle de l'image-objet sur la scène. Si l'image-objet devient plus petite que l'animation Flash d'origine, l'animation affichée dans l'image-objet est recadrée pour tenir dans les limites de l'image-objet.
- `#autoSize` (valeur par défaut) – Cette valeur spécifie que le rectangle de l'image-objet est automatiquement mis à l'échelle et positionné en fonction des propriétés `rotation`, `skew`, `flipH` et `flipV`. Autrement dit, lorsqu'une image-objet Flash pivote, elle n'est pas recadrée comme dans les versions précédentes de Director. Le paramètre `#autoSize` ne fonctionne correctement que si `scale`, `viewScale`, `originPoint` et `viewPoint` ont leurs valeurs par défaut.

Cette propriété peut être testée et définie.

Exemple

Le script d'image-objet suivant vérifie la couleur de la scène de l'animation Director et, si cette couleur est indexée à la position 0 de la palette courante, il attribue à la propriété `scaleMode` d'une image-objet animation Flash la valeur `#showAll`. Sinon, il lui affecte la valeur `#noBorder`.

```
-- Syntaxe Lingo
property spriteNum

on beginsprite me
  if _movie.stage.bgColor = 0 then
    sprite(spriteNum).scaleMode = #showAll
  else
    sprite(spriteNum).scaleMode = #noBorder
  end if
end
```

```
// Syntaxe JavaScript
function beginsprite() {
  var stgClr = _movie.stage.bgColor;
  if (stgClr = 0) {
    sprite(this.spriteNum).scaleMode = symbol("showAll");
  } else {
    sprite(this.spriteNum).scaleMode = symbol("noBorder");
  }
}
```

Voir aussi

scale (acteur)

score

Utilisation

```
-- Syntaxe Lingo
_movie.score

// Syntaxe JavaScript
_movie.score;
```

Description

Propriété d'animation ; détermine le scénario associé à l'animation courante. Lecture/écriture.

Cette propriété peut servir à conserver le contenu courant du scénario avant de l'effacer, ainsi qu'à générer un nouveau contenu de scénario ou à affecter le contenu courant à une boucle.

Exemple

L'instruction suivante affecte l'acteur boucle Cascade au scénario de l'animation courante :

```
-- Syntaxe Lingo
_movie.score = member("Cascade").media

// Syntaxe JavaScript
_movie.score = member("Cascade").media;
```

Voir aussi

Animation

scoreColor

Utilisation

```
sprite(quelleImageObjet).scoreColor
the scoreColor of sprite quelleImageObjet
```

Description

Propriété d'image-objet ; indique la couleur de scénario affectée à l'image-objet spécifiée par *quelleImageObjet*. Les valeurs possibles correspondent aux puces de couleur 0 à 5 de la palette courante.

Cette propriété peut être testée et définie. La définition de cette propriété n'est utile que pendant la programmation et l'enregistrement du scénario.

Exemple

L'instruction suivante affiche dans la fenêtre Messages la valeur de la couleur de scénario affectée à l'image-objet 7 :

```
put sprite(7).scorecolor
```

scoreSelection

Utilisation

```
-- Syntaxe Lingo
_movie.scoreSelection

// Syntaxe JavaScript
_movie.scoreSelection;
```

Description

Propriété d'animation ; détermine les pistes sélectionnées dans la fenêtre Scénario. Lecture/écriture.

Les informations sont formatées sous forme d'une liste linéaire de listes linéaires. Chaque sélection contiguë figure dans un format de liste comprenant le numéro de la piste initiale, le numéro de la piste finale, le numéro de l'image initiale et celui de l'image finale. Spécifiez les pistes d'images-objets en indiquant leur numéro de piste. Utilisez les numéros suivants pour indiquer d'autres pistes.

Pour spécifier :	Utilisez :
Piste des scripts de l'image	0
Piste audio 1	-1
Piste audio 2	-2
Piste des transitions	-3
Piste des palettes	-4
Piste des cadences	-5

Vous pouvez sélectionner des pistes ou des images non adjacentes.

Exemple

L'instruction suivante sélectionne les pistes d'images-objets 15 à 25 dans les images 100 à 200 :

```
-- Syntaxe Lingo
_movie.scoreSelection = [[15, 25, 100, 200]]

// Syntaxe JavaScript
_movie.scoreSelection = list(list(15, 25, 100, 200));
```

L'instruction suivante sélectionne les pistes d'images-objets 15 à 25 et 40 à 50 dans les images 100 à 200 :

```
-- Syntaxe Lingo
_movie.scoreSelection = [[15, 25, 100, 200], [40, 50, 100, 200]]

// Syntaxe JavaScript
_movie.scoreSelection = list(list(15, 25, 100, 200), list(40, 50, 100, 200));
```

L'instruction suivante sélectionne le script d'image des images 100 à 200 :

```
-- Syntaxe Lingo
_movie.scoreSelection = [[0, 0, 100, 200]]

// Syntaxe JavaScript
_movie.scoreSelection = list(list(0, 0, 100, 200));
```

Voir aussi

Animation

script

Utilisation

```
-- Syntaxe Lingo
_movie.script[nomOuNumDeScript]

// Syntaxe JavaScript
_movie.script[nomOuNumDeScript];
```

Description

Propriété d'animation ; fournit un accès nommé ou indexé aux acteurs script d'une animation. En lecture seule.

L'argument *nomOuNumDeScript* peut être une chaîne spécifiant le nom de l'acteur script ou un entier spécifiant le numéro de cet acteur.

- Si *nomOuNumDeScript* est une chaîne, la propriété `script` fournit un accès à l'acteur script, quelle que soit la bibliothèque de distribution comportant cet acteur.
- Si *nomOuNumDeScript* est un entier, la propriété `script` ne fournit d'accès qu'à l'acteur script trouvé dans la première bibliothèque de distribution de l'animation référencée ; vous ne pouvez pas utiliser d'accès indexé pour spécifier une bibliothèque de distribution autre que la première.

Exemple

L'instruction suivante accède à un script nommé.

```
-- Syntaxe Lingo
bugScript = _movie.script["Fourmi"]

// Syntaxe JavaScript
var bugScript = _movie.script["Fourmi"];
```

Voir aussi

Animation

scripted

Utilisation

```
-- Syntaxe Lingo
réfObjPisteDimageObjet.scripted

// Syntaxe JavaScript
réfObjPisteDimageObjet.scripted;
```

Description

Propriété de piste d'image-objet ; indique si une piste d'image-objet est contrôlée par un script (TRUE) ou par le scénario (FALSE). En lecture seule.

Exemple

Les instructions suivantes créent une image-objet scriptée à partir de l'acteur Cerf-volant dans la piste d'image-objet 5 si cette piste n'est pas déjà contrôlée par scripts.

```
-- Syntaxe Lingo
if (channel(5).scripted = FALSE) then
  channel(5).makeScriptedSprite(member("cerf-volant"))
end if

// Syntaxe JavaScript
if (channel(5).scripted == false) {
  channel(5).makeScriptedSprite(member("cerf-volant"));
}
```

Voir aussi

Piste d'image-objet

scriptingXtraList

Utilisation

```
-- Syntaxe Lingo
_player.scriptingXtraList

// Syntaxe JavaScript
_player.scriptingXtraList;
```

Description

Propriété de lecteur ; renvoie une liste linéaire de tous les Xtras de programmation disponibles sur le lecteur Director. En lecture seule.

Ces Xtras se trouvent dans le dossier Configuration\Xtras.

Exemple

L'instruction suivante affiche, dans la fenêtre Messages, tous les Xtras de programmation disponibles :

```
-- Syntaxe Lingo
trace(_player.scriptingXtraList)

// Syntaxe JavaScript
trace(_player.scriptingXtraList);
```

Voir aussi

mediaXtraList, Lecteur, Objets de scripting, toolXtraList, transitionXtraList, xtraList (lecteur)

scriptInstanceList

Utilisation

```
sprite(quelleImageObjet).scriptInstanceList
the scriptInstanceList of sprite quelleImageObjet
```

Description

Propriété d'image-objet ; crée une liste des références de script liées à une image-objet. Cette propriété n'est disponible que pendant l'exécution. La liste est vide lorsque l'animation n'est pas en cours de lecture. Les modifications apportées à la liste ne sont pas enregistrées dans le scénario. Cette propriété est utile aux tâches suivantes :

- associer un comportement à une image-objet pour une utilisation pendant l'exécution ;
- déterminer si des comportements sont associés à une image-objet et déterminer quels sont ces comportements ;
- trouver une référence de script de comportement à utiliser avec la commande `sendSprite`.

Cette propriété peut être testée et définie. Elle ne peut être modifiée que si l'image-objet existe déjà et qu'au moins une instance d'un comportement y est liée.

Exemple

Le gestionnaire suivant affiche la liste des références de scripts liées à une image-objet :

```
on afficherLesRefsDeScripts spriteNum
  put sprite(spriteNum).scriptInstanceList
end
```

Les instructions suivantes lient le script Grand bruit à l'image-objet 5 :

```
x = script("Grand bruit").new()
sprite(5).scriptInstanceList.add(x)
```

Voir aussi

scriptNum, sendSprite()

scriptList

Utilisation

```
sprite(quelleImageObjet).scriptList
the scriptList of sprite quelleImageObjet
```

Description

Propriété d'image-objet ; renvoie la liste des comportements associés à l'image-objet concernée, ainsi que leurs propriétés. Cette propriété ne peut être définie qu'avec `setScriptList()`. Elle ne peut pas être définie au cours d'une session d'enregistrement du scénario.

Exemple

L'instruction suivante affiche la liste des scripts associés à l'image-objet 1 dans la fenêtre Messages :

```
put sprite(1).scriptList
-- [[(member 2 of castLib 1), "[#monAngleDeRotation: 10.0000, #sensHoraire: 1,
  #angleInitial: 0.0000]"], [(member 3 of castLib 1), "[#angleParImage:
  10.0000, #tours: 10, #décalageHorizontalParImage: 10, #Décalage: 10,
  #TotalDimages: 60, #hauteurDeLaSurface: 0]"]]
```

Voir aussi

`setScriptList()`, `value()`

scriptNum

Utilisation

```
sprite(quelleImageObjet). scriptNum
scriptNum of sprite quelleImageObjet
```

Description

Propriété d'image-objet ; indique le numéro du script lié à l'image-objet spécifiée par *quelleImageObjet*. Si plusieurs scripts sont liés à l'image-objet, la propriété d'image-objet `scriptNum` renvoie le numéro du premier script. Pour une liste complète des scripts liés à une image-objet, consultez la liste de ses comportements dans l'inspecteur de comportement.

Cette propriété peut être testée et définie pendant l'enregistrement du scénario.

Exemple

L'instruction suivante affiche le numéro du script lié à l'image-objet 4 :

```
put sprite(4).scriptNum
```

Voir aussi

`scriptInstanceList`

scriptsEnabled

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.scriptsEnabled

// Syntaxe JavaScript
réfObjActeur.scriptsEnabled;
```

Description

Propriété d'acteur animation Director ; détermine si les scripts d'une animation liée sont activés (`TRUE`) ou non (`FALSE`).

Cette propriété n'est disponible que pour les acteurs animation Director liés.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante désactive les scripts dans l'animation liée Chronique Jazz :

```
-- Syntaxe Lingo
member("Chronique Jazz").scriptsEnabled = FALSE

// Syntaxe JavaScript
member("Chronique Jazz").scriptsEnabled = 0;
```

scriptText

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.scriptText

// Syntaxe JavaScript
réfObjActeur.scriptText;
```

Description

Propriété d'acteur ; indique le contenu du script (s'il existe) affecté à un acteur. Lecture/écriture.

Le texte d'un script est supprimé lorsqu'une animation est convertie en projection, protégée ou compressée au format Shockwave Player. De telles animations perdent alors les valeurs de leur propriété `scriptText`. Par conséquent, les valeurs de la propriété `scriptText` de l'animation ne peuvent pas être récupérées lorsque cette animation est lue en tant que projection. Cependant, Director peut définir de nouvelles valeurs pour la propriété `scriptText` dans la projection. Ces scripts nouvellement affectés sont compilés automatiquement pour en garantir une exécution rapide.

Exemple

L'instruction suivante fait du contenu de l'acteur champ 20 le script de l'acteur 30 :

```
-- Syntaxe Lingo
member(20).text = member(30).scriptText

// Syntaxe JavaScript
member(20).text = member(30).scriptText;
```

Voir aussi

Acteur

scriptType

Utilisation

```
member quelScript.scriptType
the scriptType of member quelScript
```

Description

Propriété d'acteur ; indique le type du script spécifié. Les valeurs possibles sont #movie, #score et #parent.

Exemple

L'instruction suivante fait de l'acteur script Script principal un script d'animation :

```
member("Script principal").scriptType = #movie
```

scrollTop

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.scrollTop

// Syntaxe JavaScript
réfObjActeur.scrollTop;
```

Description

Propriété d'acteur ; détermine la distance, en pixels, séparant le haut d'un acteur champ et le haut du champ actuellement visible dans la zone de défilement. En changeant la valeur de la propriété d'acteur scrollTop pendant la lecture de l'animation, vous pouvez modifier la section du champ qui apparaît dans la zone de défilement.

Cette procédure permet de produire des comportements de défilement personnalisés pour des acteurs texte et champ.

Par exemple, l'instruction suivante déplace l'acteur champ Crédits vers le haut ou le bas dans la zone d'un champ, suivant la valeur de la variable valeurDeGlissière :

```
global valeurDeGlissière

on prepareFrame
  nouvelleValeur = valeurDeGlissière * 100
  member("Crédits").scrolltop = nouvelleValeur
end
```

La variable globale `valeurDeGlissière` pourrait mesurer l'ampleur du déplacement d'une glissière par l'utilisateur. L'instruction `set nouvelleValeur = valeurDeGlissière * 100` multiplie `valeurDeGlissière` pour produire une valeur supérieure à celle du déplacement de la glissière par l'utilisateur. Si `valeurDeGlissière` a une valeur positive, le texte se déplace vers le haut ; si `valeurDeGlissière` a une valeur négative, il se déplace vers le bas.

Exemple

La boucle de répétition suivante fait défiler le champ Crédits en augmentant continuellement la valeur de la propriété d'acteur `scrollTop` :

```
-- Syntaxe Lingo
on wa
  member("Crédits").scrollTop = 1
  repeat with count = 1 to 150
    member("Crédits").scrollTop = member("Crédits").scrollTop + 1
    _movie.updateStage()
  end repeat
end

// Syntaxe JavaScript
function wa() {
  member("Crédits").scrollTop = 1;
  for (var count = 1; count <= 150; count++) {
    member("Crédits").scrollTop = member("Crédits").scrollTop + 1;
    _movie.updateStage();
  }
}
```

sds (modificateur)

Utilisation

`member(`*`quelActeur`*`).model(`*`quelModèle`*`).sds.`*`quellePropriété`*

Description

Modificateur 3D ; ajoute des détails géométriques aux modèles et synthétise ces détails supplémentaires pour adoucir les courbes au fur et à mesure que le modèle s'approche de la caméra. Vous pouvez définir les propriétés de ce modificateur après avoir ajouté le modificateur `sds` (à l'aide de `addModifier()`) à un modèle.

Le modificateur `sds` affecte directement la ressource de modèle. Faites attention lorsque vous utilisez les modificateurs `sds` et `lod` de concert étant donné qu'ils ont des fonctions opposées (le modificateur `sds` ajoute des détails géométriques alors que le modificateur `lod` les supprime). Il est recommandé, avant d'ajouter le modificateur `sds`, de donner à la propriété `lod.auto` la valeur `FALSE` et de choisir la résolution souhaitée pour la propriété `lod.level`, comme suit :

```
member("monActeur").model("monModèle").lod.auto = 0
member("monActeur").model("monModèle").lod.level = 100
member("monActeur").model("monModèle").addmodifier(#sds)
```

Le modificateur `sds` ne peut pas être utilisé avec des modèles qui utilisent déjà le modificateur `inker` ou `toon`.

Vous pouvez obtenir ou définir les propriétés suivantes après avoir ajouté le modificateur `sds` à une ressource de modèle :

`enabled` indique si la fonctionnalité de fractionnement de surface est activée (`TRUE`) ou désactivée (`FALSE`). Le paramètre par défaut de cette propriété est `TRUE`.

`depth` spécifie le nombre maximum de niveaux de résolution que le modèle peut afficher lorsque le modificateur `sds` est utilisé.

`error` indique le niveau de tolérance des erreurs pour la fonctionnalité de fractionnement de surface. Cette propriété s'applique seulement lorsque la propriété `sds.subdivision` a pour valeur `#adaptive`.

`subdivision` indique le mode de fonctionnement du modificateur de fractionnement de surface. Les valeurs possibles sont :

- `#uniform` spécifie que la maille est mise à l'échelle en détail de manière uniforme, chaque face étant fractionnée le même nombre de fois.
- `#adaptive` spécifie que des détails supplémentaires ne sont ajoutés qu'en présence d'un changement d'orientation majeur et seulement aux zones actuellement visibles de la maille.

Remarque : Pour plus d'informations, consultez les entrées des différentes propriétés.

Exemple

L'instruction suivante affiche la valeur de la propriété `sds.depth` du modèle Terrain.

```
put member("3D").model("Terrain").sds.depth
-- 2
```

Voir aussi

`lod (modificateur)`, `toon (modificateur)`, `inker (modificateur)`, `depth (3D)`, `enabled (sds)`, `error`, `subdivision`, `addModifier`

searchCurrentFolder

Utilisation

```
-- Syntaxe Lingo
_player.searchCurrentFolder

// Syntaxe JavaScript
_player.searchCurrentFolder;
```

Description

Propriété de lecteur ; détermine si Director traite le dossier courant lors d'une recherche de noms de fichiers. Lecture/écriture.

- Lorsque la propriété `searchCurrentFolder` a la valeur `TRUE`, Director traite le dossier courant lors de la recherche de noms de fichiers.
- Lorsque la propriété `searchCurrentFolder` a la valeur `FALSE`, Director ne traite pas le dossier courant lors de la recherche de noms de fichiers.

Cette propriété est `TRUE` par défaut.

Exemple

L'instruction suivante affiche l'état de la propriété `searchCurrentFolder` dans la fenêtre Messages. Le résultat est 1, équivalent numérique de `TRUE` :

```
-- Syntaxe Lingo
put(_player.searchCurrentFolder)

// Syntaxe JavaScript
put(_player.searchCurrentFolder);
```

Voir aussi

Lecteur

searchPathList

Utilisation

```
-- Syntaxe Lingo
_player.searchPathList

// Syntaxe JavaScript
_player.searchPathList;
```

Description

Propriété de lecteur ; liste des chemins d'accès sur lesquels Director effectue une recherche pour trouver des médias liés comme les fichiers vidéo numérique, GIF, bitmap ou audio. Lecture/écriture.

Chaque élément de la liste des chemins est un nom de chemin d'accès complet tel qu'il apparaît sur la plate-forme courante pendant l'exécution.

La valeur de `searchPathList` est une liste linéaire que vous pouvez manipuler comme n'importe quelle autre liste en utilisant des commandes telles que `add()`, `addAt()`, `append()`, `deleteAt()` et `setAt()`. La valeur par défaut est une liste vide.

Les adresses URL ne doivent pas être utilisées comme référence de fichier pour la recherche.

Un ajout important de chemins d'accès à `searchPaths` ralentit la recherche. Essayez de minimiser le nombre de chemins figurant dans la liste.

Remarque : Cette propriété fonctionne sur toutes les animations ultérieures après avoir été définie. Les éléments de l'animation courante ayant déjà été chargés, la modification du paramètre n'affecte aucun de ces éléments.

Exemple

L'instruction suivante affiche les chemins d'accès que Director prend en compte lors de la recherche de fichiers :

```
-- Syntaxe Lingo
trace(_player.searchPathList)

// Syntaxe JavaScript
trace(_player.searchPathList);
```

L'instruction suivante affecte deux dossiers à `searchPaths` sous Windows :

```
-- Syntaxe Lingo
_player.searchPathList = ["C:\Director\Projects\", "D:\CDROM\Sources\"]

// Syntaxe JavaScript
_player.searchPathList = list("C:\\Director\\Projects\\",
  "D:\\CDROM\\Sources\\");
```

L'instruction suivante affecte deux dossiers à `searchPaths` sur un Macintosh :

```
-- Syntaxe Lingo
_player.searchPathList = ["Hard Drive:Director:Projects:", "CDROM:Sources:"]

// Syntaxe JavaScript
_player.searchPathList = list("Hard Drive:Director:Projects:",
  "CDROM:Sources:");
```

Voir aussi

`Lecteur`, `searchCurrentFolder`

selectedButton

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.selectedButton

// Syntaxe JavaScript
réfObjDvd.selectedButton;
```

Description

Propriété DVD ; renvoie le bouton actif. En lecture seule.

Voir aussi

`DVD`

selectedText

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.selectedText

// Syntaxe JavaScript
réfObjActeur.selectedText;
```

Description

Propriété d'acteur texte ; renvoie le bloc de texte sélectionné sous forme de référence à un seul objet. Cela permet l'accès aux caractéristiques de police, de même qu'aux informations sur les caractères de la chaîne.

Exemple

Le gestionnaire suivant affiche le texte sélectionné et placé dans un objet de variable locale. Cet objet est ensuite utilisé pour faire référence à différentes caractéristiques du texte, détaillées dans la fenêtre Messages.

```
-- Syntaxe Lingo
property spriteNum

on mouseUp(me)
  monObjetSélection = sprite(spriteNum).member.selectedText
  put(monObjetSélection.text)
  put(monObjetSélection.font)
  put(monObjetSélection.fontSize)
  put(monObjetSélection.fontStyle)
end

// Syntaxe JavaScript
function mouseUp() {
var monObjetSélection = sprite(spriteNum).member.selectedText;
  trace(monObjetSélection.text);
  trace(monObjetSélection.font);
  trace(monObjetSélection.fontSize);
  trace(monObjetSélection.fontStyle);
}
```

selection

Utilisation

```
-- Syntaxe Lingo
réfObjDistribution.selection

// Syntaxe JavaScript
réfObjDistribution.selection;
```

Description

Propriété de bibliothèque de distribution ; renvoie les acteurs sélectionnés dans une fenêtre Distribution donnée. Lecture/écriture.

Exemple

L'instruction suivante sélectionne les acteurs 1 à 10 de la distribution numéro 1 :

```
-- Syntaxe Lingo
castLib(1).selection = [[1, 10]]

// Syntaxe JavaScript
castLib(1).selection = list( list(1, 10) );
```

L'instruction suivante sélectionne les acteurs 1 à 10 et 30 à 40 de la distribution numéro 1 :

```
-- Syntaxe Lingo
castLib(1).selection = [[1, 10], [30, 40]]

// Syntaxe JavaScript
castLib(1).selection = list( list(1, 10), list(30, 40) );
```

Voir aussi

Bibliothèque de distribution

selection (propriété d'acteur texte)

Utilisation

```
member(quelActeurTexte).selection
```

Description

Propriété d'acteur texte ; renvoie une liste du premier et du dernier caractère sélectionné dans l'acteur texte.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante définit la sélection affichée par l'image-objet de l'acteur texte maRéponse de manière à mettre en surbrillance les caractères 6 à 10 :

```
member("maRéponse").selection = [6, 10]
```

Voir aussi

`color()`, `selStart`, `selEnd`

selEnd

Utilisation

```
-- Syntaxe Lingo
selEnd

// Syntaxe JavaScript
selEnd;
```

Description

Propriété d'acteur ; spécifie le dernier caractère d'une sélection. Elle est utilisée avec `selStart` pour identifier une sélection dans le champ modifiable courant, en comptant à partir du caractère initial.

Cette propriété peut être testée et définie. La valeur par défaut est 0.

Exemple

Les instructions suivantes sélectionnent « cde » à partir du champ « abcdefg » :

```
-- Syntaxe Lingo
selStart = 3
selEnd = 5

// Syntaxe JavaScript
selStart = 2;
selEnd = 4;
```

L'instruction suivante opère une sélection d'une longueur de 20 caractères :

```
-- Syntaxe Lingo
selEnd = selStart + 20

// Syntaxe JavaScript
selEnd = selStart + 20;
```

Voir aussi

editable, hilite (commande), selection() (fonction), selStart, text

selStart

Utilisation

```
-- Syntaxe Lingo
selStart

// Syntaxe JavaScript
selStart;
```

Description

Propriété d'acteur ; spécifie le caractère initial d'une sélection. Elle est utilisée avec `selEnd` pour identifier une sélection dans le champ modifiable courant, en comptant à partir du caractère initial.

Cette propriété peut être testée et définie. La valeur par défaut est 0.

Exemple

Les instructions suivantes sélectionnent « cde » à partir du champ « abcdefg » :

```
-- Syntaxe Lingo
selStart = 3
selEnd = 5

// Syntaxe JavaScript
selStart = 2;
selEnd = 4;
```

L'instruction suivante opère une sélection d'une longueur de 20 caractères :

```
-- Syntaxe Lingo
selEnd = selStart + 20

// Syntaxe JavaScript
selEnd = selStart + 20;
```

Voir aussi

selection() (fonction), selEnd, text

serialNumber

Utilisation

```
-- Syntaxe Lingo
_player.serialNumber

// Syntaxe JavaScript
_player.serialNumber;
```

Description

Propriété d'animation ; renvoie une chaîne comprenant le numéro de série saisi pendant l'installation de Director. En lecture seule.

Cette propriété est uniquement disponible dans l'environnement de programmation. Elle peut s'utiliser dans une animation dans une fenêtre personnalisée pour afficher des informations utilisateur.

Exemple

Le gestionnaire suivant serait normalement placé dans un script d'animation dans une fenêtre. Il place le nom de l'utilisateur et le numéro de série dans une zone d'affichage dès l'ouverture de la fenêtre :

```
-- Syntaxe Lingo
on prepareMovie
  chaîneAffichée = _player.userName & RETURN & _player.organizationName \
  & RETURN & _player.serialNumber
  member("Infos utilisateur").text = chaîneAffichée
end

// Syntaxe JavaScript
function prepareMovie() {
  var chaîneAffichée = _player.userName + "\n" + _player.organizationName
  + "\n" + _player.serialNumber;
  member("Infos utilisateur").text = chaîneAffichée;
}
```

Voir aussi

Lecteur

shader

Utilisation

```
member(quelActeur).shader(quelMatériau)
member(quelActeur).shader[index]
member(quelActeur).model(quelModèle).shader
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  face[index].shader
```

Description

Propriété 3D d'élément, de modèle et de face ; objet utilisé pour définir l'apparence de la surface du modèle. Le matériau est la « peau » qui entoure la ressource de modèle utilisée par le modèle.

Le matériau même n'est pas une image. Le composant visible d'un matériau est créé avec un maximum de huit couches de `textures`. Ces huit couches de texture ont été créées à partir d'acteurs bitmap ou d'objets image dans Director ou importées avec des modèles de programmes de modélisation 3D. Pour plus d'informations, consultez `texture`.

Tout modèle dispose d'une liste linéaire de matériaux appelée `shaderList`. Le nombre d'entrées de la liste est égal au nombre de mailles de la ressource utilisée par le modèle. Un seul matériau peut être appliqué à chaque maille.

L'acteur 3D a un matériau par défaut nommé `DefaultShader`, qui ne peut pas être supprimé. Ce matériau est utilisé lorsque aucun matériau n'est affecté à un modèle et lorsqu'un matériau utilisé par un modèle est supprimé.

La syntaxe `member(`*`quelActeur`*`).model(`*`quelModèle`*`).shader` donne accès au premier matériau de la liste des matériaux du modèle et est équivalent à `member(`*`quelActeur`*`).model(`*`quelModèle`*`).shaderList[1]`.

Les matériaux sont créés et supprimés avec les commandes `newShader()` et `deleteShader()`.

Les matériaux sont enregistrés dans la palette des matériaux de l'acteur 3D. Ils peuvent être référencés par nom (*`quelMatériau`*) ou par index de palette (*`indexDeMatériau`*). Un matériau peut être utilisé par n'importe quel nombre de modèles. Les modifications apportées à un matériau apparaissent dans tous les modèles qui l'utilisent.

Il existe quatre types de matériaux :

Les matériaux `#standard` présentent leurs textures de façon réaliste.

Les matériaux `#painter`, `#engraver` et `#newsprint` stylisent leurs textures pour qu'elles aient l'apparence d'une peinture, d'une gravure ou d'un journal. Ils ont des propriétés spéciales en plus des propriétés de matériau `#standard`.

Vous trouverez plus d'informations sur les propriétés des matériaux dans la rubrique Utilisation de Director du panneau d'aide de Director.

Les matériaux utilisés par les faces individuelles des primitives `#mesh` peuvent être définis à l'aide de la syntaxe `member(`*`quelActeur`*`).modelResource(`*`quelleRessourceDeModèle`*`).face[`*`index`*`].shader`. Pour modifier cette propriété, il est nécessaire d'appeler la commande `build()`.

Exemple

L'instruction suivante affecte le matériau surfaceDuMur à la propriété de matériau du modèle Mur.

```
member("Pièce").model("Mur").shader = \
  member("Pièce").shader("surfaceDuMur")
```

Voir aussi

`shaderList`, `newShader`, `deleteShader`, `face`, `texture`

shaderList

Utilisation

```
member(quelActeur).model(quelModèle).shaderList
member(quelActeur).model(quelModèle).shaderList[index]
```

Description

Propriété 3D de modèle ; liste linéaire de `shadowPercentage` appliqué au modèle. Le nombre d'entrées de la liste est égal au nombre de mailles de la ressource utilisée par le modèle. Un seul matériau peut être appliqué à chaque maille.

Définit le matériau à la position d'*index* spécifiée dans `shaderList` à l'aide de cette syntaxe : `member(quelActeur).model(quelModèle).shaderList[index] = référenceDeMatériau`.

Dans le texte 3D, chaque caractère est constitué d'une maille distincte. Définissez la valeur de `index` sur le nombre de caractères du matériau que vous souhaitez définir.

Affecte le même matériau à toutes les positions d'*index* de `shaderList` à l'aide de cette syntaxe (remarquez l'absence de position d'index pour `shaderList`) : `member(quelActeur).model(quelModèle).shaderList = référenceDeMatériau`.

Définit la propriété d'un matériau de `shaderList` à l'aide de cette syntaxe : `member(quelActeur).model(quelModèle).shaderList[index].\quellePropriété = valeurDePropriété`

Affecte la même valeur à tous les matériaux d'un modèle à l'aide de cette syntaxe (remarquez l'absence de position d'index pour `shaderList`) :

```
member(quelActeur).model(quelModèle).shaderList.\
  quellePropriété = valeurDePropriété
```

Exemple

L'instruction suivante donne au second matériau de la liste `shaderList` du modèle pareChocs le matériau Chrome.

```
member("Voiture").model("pareChocs").shaderList[2] = \
  member("Voiture").shader("Chrome")
```

L'instruction suivante donne à tous les matériaux de la liste `shaderList` du modèle pareChocs le matériau Chrome.

```
member("Voiture").model("pareChocs").shaderList = \
  member("Voiture").shader("Chrome")
```

Voir aussi

`shadowPercentage`

shadowPercentage

Utilisation

```
member(quelActeur).model(quelModèle).toon.shadowPercentage
member(quelActeur).model(quelModèle).shader.shadowPercentage
member(quelActeur).shader(quelMatériau).shadowPercentage
```

Description

Propriété 3D de modificateur `toon` et de matériau `#painter` ; indique le pourcentage de couleurs disponibles utilisé dans la région éclairée de la surface du modèle sur laquelle la lumière n'apparaît pas.

La plage de cette propriété s'étend de 0 à 100, la valeur par défaut étant 50.

Le nombre de couleurs utilisées par le modificateur `toon` et le matériau `painter` pour un modèle est déterminé par la propriété `colorSteps` du modificateur `toon` ou du matériau `painter` du modèle.

Exemple

L'instruction suivante donne à la propriété `shadowPercentage` du modificateur `toon` du modèle Théière la valeur `50`. La moitié des couleurs disponibles pour le modificateur `toon` sera utilisée pour la région ombragée de la surface du modèle.

```
member("formes").model("Théière").toon.shadowPercentage = 50
```

Voir aussi

`colorSteps`, `shadowStrength`

shadowStrength

Utilisation

```
member(quelActeur).model(quelModèle).toon.shadowStrength
member(quelActeur).model(quelModèle).shader.shadowStrength
member(quelActeur).shader(quelMatériau).shadowStrength
```

Description

Propriété 3D de modificateur `toon` et de matériau `#painter` ; indique la luminosité de la région de la surface du modèle sur laquelle la lumière n'apparaît pas.

La valeur par défaut de cette propriété est 1.0.

Exemple

L'instruction suivante donne à la propriété `shadowStrength` du modificateur `toon` du modèle Sphère la valeur `0.1`. La partie de la surface du modèle qui n'est pas directement éclairée sera très foncée.

```
member("Formes").model("Sphère").toon.shadowStrength = 0.1
```

shapeType

Utilisation

```
member(quelActeur).shapeType
the shapeType of member quelActeur
```

Description

Propriété d'acteur forme ; indique le type de forme spécifiée. Les types possibles sont `#rect`, `#roundRect`, `#oval` et `#line`. Vous pouvez utiliser cette propriété pour spécifier le type d'un acteur forme après sa création avec Lingo.

Exemple

Les instructions suivantes créent un nouvel acteur forme avec le numéro 100, puis le définissent comme un ovale :

```
new(#shape, member 100)
member(100).shapeType = #oval
```

shiftDown

Syntaxe

```
-- Syntaxe Lingo
_key.shiftDown

// Syntaxe JavaScript
_key.shiftDown;
```

Description

Propriété de touche ; indique si l'utilisateur appuie sur la touche Maj. En lecture seule.

Cette propriété renvoie `TRUE` si l'utilisateur appuie sur la touche Maj. Sinon, elle renvoie `FALSE`.

Cette propriété doit être testée conjointement à une autre touche.

Exemple

L'instruction suivante vérifie si la touche Maj est enfoncée et, le cas échéant, appelle le gestionnaire `MajusculeA` :

```
-- Syntaxe Lingo
if (_key.shiftDown) then
  MajusculeA(_key.key)
end if

// Syntaxe JavaScript
if (_key.shiftDown) {
  MajusculeA(_key.key);
}
```

Voir aussi

`controlDown`, `Touche`, `key`, `keyCode`, `optionDown`

shininess

Utilisation

```
member(quelActeur).shader(quelMatériau).shininess
member(quelActeur).model(quelModèle).shader.shininess
member(quelActeur).model(quelModèle).shaderList\
  [indexDeListeDeMatériaux].shininess
```

Description

Propriété 3D de matériau standard ; permet d'obtenir ou de définir le brillant d'une surface. Le brillant est défini en tant que pourcentage de la surface de matériau réservée aux taches spéculaires. La valeur est un nombre entier compris entre 0 et 100, avec une valeur par défaut de 30.

Tous les matériaux ont accès aux propriétés `#standard` ; en plus de ces propriétés standard, les matériaux de type `#engraver`, `#newsprint` et `#painter` possèdent des propriétés uniques à leur type. Pour plus d'informations, consultez `newShader`.

Exemple

L'instruction suivante donne à la propriété de brillant du premier matériau de la liste des matériaux du modèle gbCyl3 la valeur 60. Soixante pour-cent de la surface du matériau seront dédiés aux reflets.

```
member("Séquence").model("gbCyl3").shader.shininess = 60
```

silhouettes

Utilisation

```
member(quelActeur).model(quelModèle).inker.silhouettes
member(quelActeur).model(quelModèle).toon.silhouettes
```

Description

Propriété 3D de modificateur `toon` et `inker` ; indique la présence (`TRUE`) ou l'absence (`FALSE`) de lignes dessinées par le modificateur sur les bords visibles du modèle.

Des lignes de silhouette sont dessinées autour de l'image 2D du modèle, sur le plan de projection de la caméra. Leur relation avec la maille du modèle n'est pas fixée, à la différence des lignes de plis ou de délimitation qui sont dessinées sur la maille.

Les lignes de silhouette sont similaires aux lignes de contour des images dans un livre de coloriage pour enfant.

La valeur par défaut de cette propriété est `TRUE`.

Exemple

L'instruction suivante donne à la propriété `silhouettes` du modificateur `inker` du modèle Sphère la valeur `FALSE`. Les lignes seront dessinées autour du profil du modèle.

```
member("Formes").model("Sphère").inker.silhouettes = FALSE
```

size

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.size

// Syntaxe JavaScript
réfObjActeur.size;
```

Description

Propriété d'acteur ; renvoie la taille occupée en mémoire, en octets, d'un acteur particulier. En lecture seule.

Divisez les octets par 1024 pour obtenir des kilo-octets.

Exemple

La ligne suivante renvoie la taille de l'acteur Temple dans un champ intitulé Taille :

```
-- Syntaxe Lingo
member("Taille").text = string(member("Temple").size)

// Syntaxe JavaScript
member("Taille").text = member("Temple").size.toString();
```

Voir aussi

Acteur

sizeRange

Utilisation

```
member(quelActeur).modelResource
(quelleRessourceDeModèle).sizeRange.start
référenceDobjetDeRessourceDeModèle.sizeRange.start
member(quelActeur).modelResource
(quelleRessourceDeModèle).sizeRange.end
référenceDobjetDeRessourceDeModèle.sizeRange.end
```

Description

Propriété 3D ; utilisée avec une ressource de modèle de type `#particle`, cette propriété permet d'obtenir ou de définir les propriétés `start` et `end` de `sizeRange` de la ressource de modèle. Les particules sont mesurées en unités d'univers.

La taille des particules du système est interpolée de façon linéaire entre `sizeRange.start` et `sizeRange.end` pendant la durée de vie de chaque particule.

Cette propriété doit être un entier supérieur à 0 et a une valeur par défaut de 1.

Exemple

Dans l'exemple suivant, mrFont est une ressource de modèle de type `#particle`. L'instruction suivante définit les propriétés `sizeRange` de mrFont. La première ligne donne la valeur de départ 4 et la seconde donne la valeur de fin 1. Cette instruction donne aux particules de mrFont une taille de 4 lorsqu'elles apparaissent pour la première fois, puis les réduit petit à petit à une taille de 1.

```
member("fontaine").modelResource("mrFont").sizeRange.start = 4
member("fontaine").modelResource("mrFont").sizeRange.end = 1
```

sizeState

Utilisation

```
-- Syntaxe Lingo
réfObjFenêtre.sizeState

// Syntaxe JavaScript
réfObjFenêtre.sizeState;
```

Description

Propriété de fenêtre ; renvoie l'état de la taille d'une fenêtre. En lecture seule.

L'état de taille renvoyé peut avoir l'une des valeurs suivantes :

Etat de taille	Description
`#minimized`	Indique que la fenêtre est actuellement réduite.
`#maximized`	Indique que la fenêtre est actuellement agrandie.
`#normal`	Indique que la fenêtre n'est actuellement ni réduite ni agrandie.

Exemple

Ces instructions agrandissent la fenêtre intitulée Artistes si elle n'est pas déjà agrandie.

```
-- Syntaxe Lingo
if (window("Artistes").sizeState <> #maximized) then
  window("Artistes").maximize()
end if

// Syntaxe JavaScript
if (window("Artistes").sizeState.toString() != "#maximized") {
  window("Artistes").maximize();
}
```

Voir aussi

`Fenêtre`

skew

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.skew

// Syntaxe JavaScript
réfObjImageObjet.skew;
```

Description

Propriété d'image-objet ; renvoie sous forme de valeur flottante exprimée en centièmes de degré, l'angle d'inclinaison des bords verticaux de l'image-objet d'un point de vue vertical. Lecture/écriture.

Des valeurs négatives signalent une inclinaison vers la gauche, tandis que des valeurs positives indiquent une inclinaison vers la droite. Des valeurs supérieures à 90° renversent l'image verticalement.

Le scénario peut conserver des informations d'inclinaison d'une image de +21 474 836,47° à -21 474 836,48°, ce qui permet 59 652 rotations complètes dans chaque direction.

Lorsque la limite d'inclinaison est atteinte (juste après la 59 652ème rotation), la commande rétablit le réglage sur +116,47° ou -116.48° – et non 0,00°. Cela s'explique par le fait que +21 474 836,47° est égal à +116,47° et -21 474 836,48° est égal à -116,48° (ou +243,52°). Pour éviter ce réglage, vous devez contraindre les angles à ±360° dans chaque direction lorsque vous effectuez une inclinaison continue avec un script.

Exemple

Le comportement suivant incline l'image-objet continuellement de 2 degrés à chaque fois que la tête de lecture avance, tout en limitant l'angle à 360 degrés :

```
-- Syntaxe Lingo
property spriteNum

on prepareFrame me
  sprite(spriteNum).skew = integer(sprite(spriteNum).skew + 2) mod 360
end

// Syntaxe JavaScript
function prepareFrame() {
  sprite(this.spriteNum).skew = parseInt(sprite(this.spriteNum).skew + 2)
  % 360;
}
```

Voir aussi

flipH, flipV, rotation, Image-objet

smoothness

Utilisation

```
member(quelActeurTexte).smoothness
member(quelActeur).modelResource(quelleRessourceDeModèleExtrudeur)\
  .smoothness
```

Description

Propriété 3D d'extrudeur et de texte ; permet d'obtenir ou de définir un nombre entier contrôlant le nombre de segments utilisés pour la création d'un acteur texte 3D. Plus ce nombre est élevé, plus le texte est lisse. La plage de cette propriété s'étend de 1 à 10, la valeur par défaut étant 5.

Pour plus d'informations sur l'utilisation des ressources de modèle d'extrudeur et des acteurs texte, consultez `extrude3D`.

Exemple

Dans l'exemple suivant, l'acteur Logo est un acteur texte. L'instruction suivante définit le lissé de Logo à 8 ; le bord de ses lettres sera très lisse en mode 3D.

```
member("Logo").smoothness = 8
```

Dans l'exemple suivant, la ressource du modèle Slogan est du texte extrudé. L'instruction suivante donne à la propriété de lissé de la ressource de modèle Slogan la valeur `1`, ce qui donne un aspect très anguleux aux lettres de Slogan.

```
member("Séquence").model("Slogan").resource.smoothness = 1
```

Voir aussi

`extrude3D`

sound (acteur)

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.sound

// Syntaxe JavaScript
réfObjActeur.sound;
```

Description

Propriété d'acteur ; permet de définir si le son d'une animation, d'une vidéo numérique ou d'une animation Flash est activé (TRUE, valeur par défaut) ou désactivé (FALSE). Lecture/écriture.

Dans les acteurs Flash, le nouveau paramètre prend effet après la lecture du son courant.

Vous pourrez voir un exemple de `sound` dans une animation en consultant l'animation Sound Control du dossier Learning/Lingo, lui-même dans le dossier de Director.

Exemple

Le gestionnaire suivant accepte une référence à un acteur et fait basculer sa propriété `sound` entre les états activé et désactivé :

```
-- Syntaxe Lingo
on ToggleSound(quelActeur)
  member(quelActeur).sound = not(member(quelActeur).sound)
end

// Syntaxe JavaScript
function ToggleSound(quelActeur) {
  member(quelActeur).sound = !(member(quelActeur).sound);
}
```

Voir aussi

Animation Flash

sound (lecteur)

Utilisation

```
-- Syntaxe Lingo
_player.sound[numPisteSonInt]

// Syntaxe JavaScript
_player.sound[numPisteSonInt];
```

Description

Propriété de lecteur ; fournit un accès indexé à un objet Piste audio en utilisant une propriété de lecteur. En lecture seule.

L'argument *numPisteSonInt* est un entier qui indique le numéro de la piste audio à laquelle accéder.

La fonctionnalité de cette propriété est identique à la méthode `sound()` de niveau supérieur.

Exemple

L'instruction suivante affecte à la variable `monSon` le son utilisé dans la piste audio 3 :

```
-- Syntaxe Lingo
monSon = _player.sound[3]

// Syntaxe JavaScript
var monSon = _player.sound[3];
```

Voir aussi

Lecteur, sound(), Piste audio

soundChannel (SWA)

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.soundChannel

// Syntaxe JavaScript
réfObjActeur.soundChannel;
```

Description

Propriété d'acteur Shockwave Audio (SWA) ; spécifie la piste audio dans laquelle le son SWA doit être lu.

Si aucun numéro de piste n'est spécifié ou si la piste 0 est indiquée, l'acteur SWA en flux continu affecte le son à la piste audio inutilisée dotée du plus haut numéro.

Les sons Shockwave Audio en flux continu peuvent apparaître sous la forme d'images-objets dans les pistes d'images-objets, mais exécutent des sons dans une piste audio. Vous devez faire référence aux images-objets son SWA par leur numéro de piste d'image-objet et non par le numéro de piste audio.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante indique à l'acteur son SWA en flux continu Frank Zappa qu'il doit être lu dans la piste 3 :

```
-- Syntaxe Lingo
member("Frank Zappa").soundChannel = 3

// Syntaxe JavaScript
member("Frank Zappa").soundChannel = 3;
```

soundChannel (RealMedia)

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.soundChannel

// Syntaxe JavaScript
réfObjActeurOuImageObjet.soundChannel;
```

Description

Propriété d'acteur ou image-objet RealMedia ; permet d'obtenir ou de définir la piste audio utilisée pour la lecture de l'audio du flux RealMedia. La définition de cette propriété permet de contrôler l'audio d'un flux RealMedia à l'aide des méthodes et propriétés audio. La définition de cette propriété à une valeur non comprise entre 0 et 8 entraîne une erreur. Cette propriété n'a aucun effet si `realPlayerNativeAudio()` a pour valeur `TRUE`.

Le paramètre par défaut de cette propriété est 0, ce qui signifie que le composant audio RealMedia sera lu sur la piste audio la plus élevée disponible et que la valeur de la propriété changera en cours de lecture, en fonction de la piste utilisée. Lorsque l'acteur RealMedia est en cours de lecture, cette propriété correspond à la piste audio active. Lorsque l'acteur RealMedia s'arrête, la valeur de cette propriété repasse à 0.

Si vous spécifiez une piste (de 1 à 8) pour cette propriété et que des sons sont actuellement en cours de lecture sur cette piste (provenant d'autres parties de l'animation), ils seront interrompus et l'audio RealMedia sera lu dans la piste.

La lecture simultanée d'autres acteurs RealMedia n'est pas prise en charge. Si votre animation contient des acteurs RealMedia qui sont lus simultanément, leurs sons seront lus en même temps sur la même piste audio.

Exemple

Les exemples suivants indiquent que le son du flux RealMedia de l'image-objet 2 et de l'acteur Real sera lu dans la piste audio 2.

```
-- Syntaxe Lingo
put(sprite(2).soundChannel) -- 2
put(member("Real").soundChannel) -- 2

// Syntaxe JavaScript
put(sprite(2).soundChannel); // 2
put(member("Real").soundChannel); // 2
```

Les exemples suivants affectent la piste audio 1 au flux RealMedia de l'image-objet 2 et l'acteur Real.

```
-- Syntaxe Lingo
sprite(2).soundChannel = 1
member("Real").soundChannel = 1

// Syntaxe JavaScript
sprite(2).soundChannel = 1;
member("Real").soundChannel = 1;
```

Voir aussi

`realPlayerNativeAudio()`

soundDevice

Utilisation

```
-- Syntaxe Lingo
_sound.soundDevice

// Syntaxe JavaScript
_sound.soundDevice;
```

Description

Propriété audio ; permet le réglage du dispositif de mixage audio pendant la lecture de l'animation. Lecture/écriture.

Les paramètres possibles pour `soundDevice` sont les dispositifs énumérés dans `soundDeviceList`.

Plusieurs dispositifs audio peuvent être répertoriés. Les différents dispositifs audio pour Windows présentent des avantages distincts.

- MacroMix (Windows) – Plus petit dénominateur commun pour une lecture audio Windows. Ce dispositif fonctionne sur n'importe quel ordinateur Windows, mais avec une latence moins bonne que celle des autres dispositifs.

- QT3Mix (Windows) – Mixe le son avec l'audio QuickTime et parfois d'autres applications si celles-ci utilisent DirectSound. Ce dispositif ne fonctionne que si QuickTime est installé et offre un meilleur temps d'exécution que MacroMix.
- DirectSound (Windows)—Similaire à QT3Mix, mais fournit une latence plus élevée.
- MacSoundManager (Macintosh) – Seul dispositif audio disponible sur Macintosh.

Exemple

L'instruction suivante règle le dispositif audio sur MacroMix pour un ordinateur tournant sous Windows. En cas de défaillance du dispositif nouvellement affecté, la propriété `soundDevice` n'est pas modifiée.

```
-- Syntaxe Lingo
_sound.soundDevice = "MacroMix"

// Syntaxe JavaScript
_sound.soundDevice = "MacroMix";
```

Voir aussi

Son, soundDeviceList

soundDeviceList

Utilisation

```
-- Syntaxe Lingo
_sound.soundDeviceList

// Syntaxe JavaScript
_sound.soundDeviceList;
```

Description

Propriété audio ; crée une liste linéaire des dispositifs audio disponibles sur l'ordinateur utilisé. En lecture seule.

Pour le Macintosh, cette propriété ne recense qu'un dispositif, à savoir MacSoundManager.

Exemple

L'instruction suivante affiche une liste de types de dispositifs audio typique sur un ordinateur Windows :

```
-- Syntaxe Lingo
trace(_sound.soundDeviceList)

// Syntaxe JavaScript
trace(_sound.soundDeviceList);
```

Voir aussi

Son, soundDevice

soundEnabled

Utilisation

```
-- Syntaxe Lingo
_sound.soundEnabled

// Syntaxe JavaScript
_sound.soundEnabled;
```

Description

Propriété audio ; détermine si le son est activé (`TRUE`, valeur par défaut) ou désactivé (`FALSE`). Lecture/écriture.

Lorsque vous donnez à cette propriété la valeur `FALSE`, le son est désactivé, mais le réglage du volume ne change pas.

Exemple

L'instruction suivante inverse le paramètre courant de `soundEnabled` (active le son s'il est désactivé et vice-versa) :

```
-- Syntaxe Lingo
_sound.soundEnabled = not (_sound.soundEnabled)

// Syntaxe JavaScript
_sound.soundEnabled = !(_sound.soundEnabled);
```

Voir aussi

Son

soundKeepDevice

Utilisation

```
-- Syntaxe Lingo
_sound.soundKeepDevice

// Syntaxe JavaScript
_sound.soundKeepDevice;
```

Description

Propriété audio ; pour Windows uniquement, détermine si le pilote audio est vidé et rechargé à chaque fois qu'un son doit être lu. Lecture/écriture.

La valeur par défaut de cette propriété est `TRUE`, ce qui évite au pilote audio d'être vidé et rechargé à chaque fois qu'un son doit être lu.

Vous devrez peut-être régler cette propriété sur `FALSE` avant de lire un son pour vous assurer que le dispositif audio est vidé et disponible pour les autres applications ou processus de l'ordinateur une fois que la lecture du son est terminée.

Le fait de régler cette propriété sur `FALSE` risque de nuire à la performance si le son est lu fréquemment par l'application Director.

Exemple

L'instruction suivante donne à la propriété `soundKeepDevice` la valeur `FALSE` :

```
-- Syntaxe Lingo
_sound.soundKeepDevice = FALSE

// Syntaxe JavaScript
_sound.soundKeepDevice = false;
```

Voir aussi

`Son`

soundLevel

Utilisation

```
-- Syntaxe Lingo
_sound.soundLevel

// Syntaxe JavaScript
_sound.soundLevel;
```

Description

Propriété audio ; renvoie ou définit le niveau de volume du son lu dans le haut-parleur de l'ordinateur. Lecture/écriture.

La plage des valeurs possibles s'étend de 0 (silence) à 7 (valeur maximale, par défaut).

Sous Windows, le réglage audio du système est combiné au contrôle du volume des haut-parleurs externes. Par conséquent, le volume réel obtenu après le réglage du son peut varier. Evitez de régler la propriété `soundLevel` à moins d'être certain que le résultat sera acceptable pour l'utilisateur. Il est préférable de régler individuellement le volume des pistes et des images-objets grâce à la propriété `volume` de l'objet Piste audio.

Ces valeurs correspondent aux paramètres présents dans le tableau de commande Son du Macintosh. L'utilisation de cette propriété permet au script de changer directement le volume du son ou d'effectuer une autre opération lorsque le son est situé à un niveau spécifié.

Vous pourrez voir un exemple de `soundLevel` dans une animation en consultant l'animation Sound Control du dossier Learning/Lingo, lui-même dans le dossier de Director.

Exemple

L'instruction suivante définit la variable `ancienSon` pour qu'elle corresponde au niveau sonore courant :

```
-- Syntaxe Lingo
ancienSon = _sound.soundLevel

// Syntaxe JavaScript
var ancienSon = _sound.soundLevel;
```

L'instruction suivante règle le niveau sonore sur 5 :

```
-- Syntaxe Lingo
_sound.soundLevel = 5

// Syntaxe JavaScript
_sound.soundLevel = 5;
```

Voir aussi

Son, volume (Windows Media)

soundMixMedia

Utilisation

```
-- Syntaxe Lingo
_sound.soundMixMedia

// Syntaxe JavaScript
_sound.soundMixMedia;
```

Description

Propriété audio ; indique si les acteurs Flash mixent leur son avec les sons des pistes audio du scénario. Lecture/écriture.

Cette propriété est définie par défaut sur TRUE pour les animations créées dans Director 7 (et versions suivantes) et sur FALSE dans les versions précédentes. Elle ne fonctionne également que sous Windows.

Lorsque cette propriété a la valeur TRUE, les acteurs Flash mixent leur son avec les sons des pistes audio du scénario. Director prend en charge le mixage et la lecture des sons des acteurs Flash.

Il est possible que de légères différences soient constatées dans la lecture des sons Flash. Pour entendre le rendu exact de ces sons, définissez cette propriété sur FALSE.

Lorsque cette propriété est définie sur FALSE, les sons Flash ne sont pas mixés et doivent être lus séparément.

Voir aussi

Son

source

Utilisation

```
sprite(quelleImageObjet).camera.backdrop[indexDeFond].source
member(quelActeur).camera(quelleCaméra).backdrop
[indexDeFond].source
sprite(quelleImageObjet).camera.overlay[indexDeRecouvrement].source
member(quelActeur).camera(quelleCaméra).overlay
[indexDeRecouvrement].source
```

Description

Propriété 3D de fond et de recouvrement ; permet d'obtenir ou de définir la texture utilisée comme image source du recouvrement ou du fond.

Exemple

L'instruction suivante donne à la source du fond 1 la texture Cèdre.

```
sprite(3).camera.backdrop[1].source =
 sprite(3).member.texture("Cèdre")
```

Voir aussi

bevelDepth, overlay

sourceFileName

Utilisation

quelActeurFlash.sourceFileName

Description

Propriété d'acteur Flash ; spécifie le chemin d'accès du fichier FLA source à utiliser lors des opérations de lancement et d'édition. Cette propriété peut être testée et définie. La valeur par défaut est une chaîne vide.

Exemple

L'instruction Lingo suivante donne à sourceFileName de l'acteur Flash SWF la valeur C:\fichiersFlash\monFichier.fla :

```
member("SWF").sourceFileName = "C:\fichiersFlash\monFichier.fla"
```

sourceRect

Utilisation

```
-- Syntaxe Lingo
rectObjFenêtre.sourceRect

// Syntaxe JavaScript
rectObjFenêtre.sourceRect;
```

Description

Propriété de fenêtre ; détermine les coordonnées initiales de la scène de l'animation exécutée dans une fenêtre. En lecture seule.

Cette propriété est utile pour rétablir la taille et la position d'origine d'une fenêtre lorsque celle-ci a été modifiée par le curseur ou la définition de son rectangle.

Exemple

L'instruction suivante affiche les coordonnées d'origine de la scène intitulée Tableau_de_commande dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(window("Tableau_de_commande").sourceRect)

// Syntaxe JavaScript
put(window("Tableau_de_commande").sourceRect);
```

Voir aussi

Fenêtre

specular (lumière)

Utilisation

```
member(quelActeur).light(quelleLumière).specular
```

Description

Propriété 3D de lumière ; permet de savoir ou de définir si la spécularité est activée (TRUE) ou non (FALSE). La spécularité est la capacité de reflet sur un modèle lorsque la lumière dirigée vers le modèle est reflétée vers la caméra. La brillance du matériau d'un objet détermine sa proportion spéculaire. La valeur de cette propriété est ignorée pour les lumières d'ambiance. La valeur par défaut de cette propriété est TRUE.

Remarque : La désactivation de cette propriété peut améliorer la performance.

Exemple

L'instruction suivante donne à la propriété spéculaire de la lumière omni2 la valeur FALSE. Cette lumière n'entraîne pas de reflets. S'il s'agit de la seule lumière éclairant la scène, il n'y aura aucun reflet spéculaire sur les matériaux de la scène.

```
member("Univers 3D").light("omni2").specular = FALSE
```

Voir aussi

silhouettes, specularLightMap

specular (matériau)

Utilisation

```
member(quelActeur).shader(quelMatériau).specular
```

Description

Propriété 3D de matériau standard ; permet d'obtenir ou de définir la couleur spéculaire d'un matériau donné. La couleur spéculaire est la couleur de la tache spéculaire générée lorsque la spécularité est activée. Pour que cette propriété ait un effet, il faut que la propriété spéculaire de certaines lumières de la scène ait pour valeur TRUE. La couleur spéculaire est influencée par la couleur des lumières qui éclairent l'objet. Si la couleur spéculaire est blanche mais que la couleur d'une lumière est rouge, un reflet rouge spéculaire apparaîtra sur l'objet. La valeur par défaut de cette propriété est rgb(255, 255, 255), qui est le blanc.

Tous les matériaux ont accès aux propriétés #standard ; en plus de ces propriétés standard, les matériaux de type #engraver, #newsprint et #painter possèdent des propriétés uniques à leur type. Pour plus d'informations, consultez newShader.

Exemple

```
put member("séquence").shader("matPluton").specular
-- rgb(11, 11, 11)
```

Voir aussi

silhouettes, specular (lumière), specularColor, emissive

specularColor

Utilisation

```
member(quelActeur).specularColor
```

Description

Propriété 3D d'acteur ; permet d'obtenir ou de définir la valeur RVB de la couleur spéculaire du premier matériau de l'acteur. Le premier matériau de la palette d'un acteur est toujours le matériau par défaut. Comme toutes les autres propriétés d'acteur 3D, cette propriété est enregistrée avec l'acteur et sera restaurée la prochaine fois que vous ouvrirez l'animation. La valeur par défaut de cette propriété est `rgb(255, 255, 255)`, qui est le blanc.

Exemple

L'instruction suivante donne à la couleur spéculaire du premier matériau de l'acteur Séquence la valeur `rgb(255, 0, 0)`. Equivalent de `member("Séquence").shader[1].specular = rgb(255, 0, 0)`. Veuillez toutefois remarquer que la syntaxe précédente n'enregistrera pas la nouvelle valeur avec l'acteur lorsque l'animation sera enregistrée. Seul `member.specularColor` enregistrera la nouvelle valeur de couleur.

```
member("Séquence").specularColor = rgb(255, 0, 0)
```

Voir aussi

`silhouettes`, `specular (lumière)`, `specular (matériau)`

specularLightMap

Utilisation

```
member(quelActeur).shader(quelMatériau).specularLightMap
member(quelActeur).model(quelModèle).shader.specularLightMap
member(quelActeur).model(quelModèle).shaderList\
  [indexDeListeDeMatériaux].specularLightMap
```

Description

Propriété 3D de matériau standard ; permet d'obtenir ou de définir la cinquième couche de texture d'un matériau standard donné. Cette propriété est ignorée si le modificateur `toon` est appliqué à la ressource de modèle.

Les valeurs peuvent être définies comme suit :

- `textureModeList[5] = #specular`
- `blendFunctionList[5] = #add`
- `blendFunctionList[1] = #replace`
- `default = void`

Cette propriété fournit une interface simple pour la définition d'une utilisation commune du placage de lumière spéculaire.

Tous les matériaux ont accès aux propriétés `#standard` ; en plus de ces propriétés standard, les matériaux de type `#engraver`, `#newsprint` et `#painter` possèdent des propriétés uniques à leur type. Pour plus d'informations, consultez `newShader`.

Exemple

L'instruction suivante donne la texture Ovale comme propriété specularLightMap au matériau utilisé par le modèle boîteEnVerre.

```
member("planète3D").model("BoîteEnVerre").shader.specularLightMap = \
  member("planète3D").texture("Ovale")
```

Voir aussi

diffuseLightMap

spotAngle

Utilisation

member(*quelActeur*).light(*quelleLumière*).spotAngle

Description

Propriété 3D ; permet d'obtenir ou de définir l'angle du cône de projection de lumière. La lumière extérieure à l'angle spécifié pour cette propriété n'apporte aucune intensité. Cette propriété accepte les valeurs à virgule flottante comprises entre 0.0 et 180.00, sa valeur par défaut étant 90.0. La valeur flottante que vous spécifiez correspond à la moitié de l'angle. Par exemple, si vous souhaitez spécifier un angle de 90°, vous devez indiquer une valeur de 45.

Exemple

L'instruction suivante donne à la propriété `spotAngle` de la lumière unidirectionnelle la valeur 8. L'angle du cône de projection de la lumière sera de 16°.

```
member("Univers 3D").light("unidirectionnelle").spotAngle = 8
```

spotDecay

Utilisation

member(*quelActeur*).light(*quelleLumière*).spotDecay

Description

Propriété 3D de lumière ; permet de savoir ou de définir si l'intensité d'un projecteur diminue avec l'éloignement de la caméra. La valeur par défaut de cette propriété est `FALSE`.

Exemple

L'instruction suivante donne à la propriété `spotDecay` de la lumière 1 la valeur `TRUE`. Les modèles les plus éloignés de la lumière 1 seront moins éclairés que les modèles les plus proches.

```
member("Séquence").light[1].spotDecay = TRUE
```

sprite (animation)

Utilisation

```
-- Syntaxe Lingo
_movie.sprite[nomOuNumDImageObjet]

// Syntaxe JavaScript
_movie.sprite[nomOuNumDImageObjet];
```

Description

Propriété d'animation ; fournit un accès nommé ou indexé à une image-objet d'animation. En lecture seule.

L'argument *nomOuNumDImageObjet* peut être une chaîne spécifiant le nom de l'image-objet ou un entier spécifiant le numéro de cette image-objet.

Exemple

L'instruction suivante affecte la variable ImageObjetSport à l'image-objet animation 5 :

```
-- Syntaxe Lingo
ImageObjetSport = _movie.sprite[5]

// Syntaxe JavaScript
var ImageObjetSport = _movie.sprite[5];
```

Voir aussi

Animation

sprite (piste d'image-objet)

Utilisation

```
-- Syntaxe Lingo
réfObjPisteDimageObjet.sprite

// Syntaxe JavaScript
réfObjPisteDimageObjet.sprite;
```

Description

Propriété de piste d'image-objet ; renvoie une référence à l'image-objet de l'image courante d'une piste d'image-objet. En lecture seule.

Exemple

L'instruction suivante affecte la variable `monImageObjet` à l'image-objet dans la piste d'images-objets Ruban.

```
-- Syntaxe Lingo
monImageObjet = channel("Ruban").sprite

// Syntaxe JavaScript
var monImageObjet = channel("Ruban").sprite;
```

Voir aussi

Piste d'image-objet

spriteNum

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.spriteNum

// Syntaxe JavaScript
réfObjImageObjet.spriteNum;
```

Description

Propriété d'image-objet ; détermine le numéro de piste dans laquelle l'image-objet d'un comportement se trouve et la rend accessible à n'importe quel comportement. En lecture seule.

Il suffit simplement de déclarer la propriété en haut du comportement, avec toutes les autres propriétés que celui-ci pourra utiliser.

Lorsque vous utilisez un gestionnaire `new()` pour créer une instance du comportement, le gestionnaire `new()` du script doit explicitement affecter la propriété `spriteNum` au numéro de l'image-objet. Cela permet d'identifier l'image-objet à laquelle le script est associé. Le numéro de l'image-objet doit être envoyé au gestionnaire `new()` sous forme d'argument lors de l'appel de ce gestionnaire.

Exemple

Dans le gestionnaire suivant, la propriété `spriteNum` est automatiquement définie pour les instances de script créées par le système :

```
-- Syntaxe Lingo
property spriteNum, pRéfDimageObjet

on mouseDown me
  sprite(spriteNum).member = member("imageEnfoncée")
end

// Syntaxe JavaScript
function mouseDown() {
  sprite(this.spriteNum).member = member("imageEnfoncée");
}
```

Le gestionnaire suivant utilise la valeur automatique insérée dans la propriété `spriteNum` pour affecter la référence de l'image-objet à une nouvelle variable de propriété `pRéfDimageObjet`, dans un but de commodité :

```
-- Syntaxe Lingo
property spriteNum, pRéfDimageObjet

on beginSprite me
  pRéfDimageObjet = sprite(me.spriteNum)
end

// Syntaxe JavaScript
function beginSprite() {
  this.pRéfDimageObjet = sprite(this.spriteNum);
}
```

Cette approche permet d'utiliser la référence `pRéfDimageObjet` dans la suite du script, le gestionnaire utilisant la syntaxe :

```
-- Syntaxe Lingo
acteurCourant = pRéfDimageObjet.member

// Syntaxe JavaScript
var acteurCourant = pRéfDimageObjet.member
```

à la place de la syntaxe suivante, légèrement plus longue :

```
-- Syntaxe Lingo
acteurCourant = sprite(spriteNum).member

// Syntaxe JavaScript
var acteurCourant = sprite(this.spriteNum).member
```

Cette seconde approche existe uniquement dans un but de facilité et n'offre aucune fonction différente.

Voir aussi

new(), Image-objet

stage

Utilisation

```
-- Syntaxe Lingo
_movie.stage

// Syntaxe JavaScript
_movie.stage;
```

Description

Propriété d'animation ; fait référence à l'animation principale. En lecture seule.

Cette propriété est utile lors de l'envoi d'un message à l'animation principale à partir d'une animation enfant.

Exemple

L'instruction suivante affiche les paramètres courants de la scène :

```
-- Syntaxe Lingo
put(_movie.stage.rect)

// Syntaxe JavaScript
put(_movie.stage.rect);
```

Voir aussi

Animation

startAngle

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).
startAngle
référenceDobjetDeRessourceDeModèle.startAngle
```

Description

Propriété 3D ; utilisée avec une ressource de modèle de type `#cylinder` ou `#sphere`, cette commande permet d'obtenir et de définir la propriété `startAngle` de la ressource de modèle référencée, sous la forme d'une valeur à virgule flottante allant de 0.0 à 360.0. La valeur par défaut de cette propriété est 0.0.

La propriété `startAngle` détermine l'angle de démarrage du balayage de la ressource de modèle et fonctionne conjointement avec la propriété `endAngle` pour dessiner des sphères et des cylindres. Par exemple, pour obtenir une demi-sphère, donnez à `startAngle` la valeur 0.0 et à `endAngle` la valeur 180.0.

Exemple

L'instruction suivante donne à la propriété startAngle de la ressource de modèle Sphère01 la valeur 0.0. Si sa propriété `endAngle` a pour valeur 90, seul un quart des modèles utilisant cette ressource de modèle apparaîtra.

```
put member("Univers 3D").modelResource("sphère01").startAngle
-- 0.0
```

Voir aussi

`endAngle`

startFrame

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.startFrame

// Syntaxe JavaScript
réfObjImageObjet.startFrame;
```

Description

Propriété d'image-objet ; renvoie le numéro de l'image de départ de l'étendue d'une image-objet. En lecture seule.

Cette propriété permet de déterminer l'étendue associée à une image-objet précise dans le scénario. Elle est uniquement disponible dans une image contenant l'image-objet et ne peut pas être appliquée à des images-objets dans d'autres images de l'animation.

Exemple

L'instruction suivante affiche l'image de départ de l'image-objet dans la piste 5 dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(sprite(5).startFrame)

// Syntaxe JavaScript
put(sprite(5).startFrame);
```

Voir aussi

endFrame, Image-objet

startTime

Utilisation

```
-- Syntaxe Lingo
réfObjPisteAudio.startTime

// Syntaxe JavaScript
réfObjPisteAudio.startTime;
```

Description

Propriété de piste audio ; indique le moment de départ du son actuellement en cours ou en pause, lorsque ce son était placé en file d'attente. En lecture seule.

Cette propriété ne peut pas être définie après le placement du son en file d'attente. Si aucune valeur n'a été définie au moment du placement du son en file d'attente, cette propriété renvoie automatiquement 0.

Exemple

L'instruction suivante fait démarrer l'image-objet vidéo numérique de la piste 5 à 100 battements :

```
-- Syntaxe Lingo
sprite(5).startTime = 100

// Syntaxe JavaScript
sprite(5).startTime = 100;
```

Voir aussi

Piste audio

startTimeList

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.startTimeList

// Syntaxe JavaScript
réfObjDvd.startTimeList;
```

Description

Propriété DVD ; liste des propriétés qui spécifie l'heure ou le chapitre auquel la lecture commence. Lecture/écriture.

`startTimeList` est une liste de propriétés qui peut être basée sur les chapitres ou sur le temps.

Une liste `startTimeList` basée sur les chapitres contient les propriétés suivantes :

- `title`. Spécifie le titre contenant le chapitre à lire.
- `chapter`. Spécifie le chapitre à lire.

Une liste `startTimeList` basée sur le temps contient les propriétés suivantes :

- `title`. Spécifie le titre.
- `hours`. Spécifie l'heure à laquelle la lecture commence.
- `min`. Spécifie la minute à laquelle la lecture commence.
- `sec`. Spécifie la seconde à laquelle la lecture commence.
- `frames`. Spécifie les images sur lesquelles la lecture commence.

La liste `startTimeList` peut être effacée en la définissant sur 0.

Exemple

Cette liste `startTimeList` commence la lecture au chapitre 4 du titre 1 :

```
[#title:1, #chapter:4]
```

Cette liste `startTimeList` commence la lecture à un moment spécifique du titre 1 :

```
[#title:1, #hours:0, #minutes:55, #seconds:45, #frames:15]
```

Cette liste `startTimeList` ne contient qu'un seul paramètre de temps :

```
[#title:1, #seconds:45]
```

Voir aussi

DVD

state (3D)

Utilisation

```
member(quelActeur).state
```

Description

Propriété 3D ; renvoie l'état courant de l'acteur référencé au cours du chargement et de la lecture en flux continu. Cette propriété indique la quantité chargée du fichier initial ou du dernier fichier demandé.

La propriété `state` de l'acteur détermine, s'il y en a un, le code Lingo 3D à exécuter sur l'acteur.

Cette propriété peut avoir une des valeurs suivantes :

- `0`—indique que l'acteur n'est pas actuellement chargé et que, par conséquent, aucun média 3D n'est disponible. Aucune opération Lingo 3D ne devrait être réalisée sur l'acteur.
- `1`—indique que le chargement du média a commencé.
- `2`—indique que le segment de chargement initial de l'acteur est chargé. Tous les objets dont la propriété de chargement en flux continu est zéro (tel que cela est défini à la création du fichier du modèle) seront alors chargés, car ils font partie du segment de chargement initial. Vous pouvez exécuter la plupart des instructions Lingo 3D associées aux objets dont la priorité de chargement est zéro. N'utilisez pas les commandes `loadFile` et `resetWorld` au cours de cet état.
- `3`—indique que tout média supplémentaire de l'acteur est chargé et décompressé. La plupart des instructions Lingo 3D peuvent alors être exécutées. N'utilisez pas les commandes `loadFile` et `resetWorld` au cours de cet état.
- `4`—indique que tous les médias de l'acteur ont été entièrement chargés et décompressés. Toutes les instructions Lingo 3D peuvent maintenant être exécutées sur l'acteur.
- `-1`—indique qu'une erreur non définie a eu lieu pendant le chargement en flux continu des médias. Etant donné que l'erreur peut avoir eu lieu à n'importe quel moment du chargement, l'état de l'acteur n'est pas fiable.

En règle générale, évitez d'utiliser Lingo avec des acteurs 3D dont l'état courant est inférieur à 3.

Exemple

L'instruction suivante indique que le chargement de l'acteur scèneDeFête est terminé, qu'il a eu lieu sans erreur et que l'acteur est prêt pour la lecture.

```
put member("scèneDeFête").state
-- 4
```

state (Flash, SWA)

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.state

// Syntaxe JavaScript
réfObjActeur.state;
```

Description

Propriété d'acteur ; pour les acteurs Shockwave Audio (SWA) lus en flux continu et les acteurs animation Flash, détermine l'état courant du fichier en flux continu. Les propriétés `streamName`, `URL` et `preLoadTime` ne peuvent être modifiées que lorsque le son SWA est arrêté.

Les propriétés de fichier SWA suivantes ne renvoient des informations utiles qu'une fois la lecture en flux continu commencée : `cuePointNames`, `cuePointTimes`, `currentTime`, `duration`, `percentPlayed`, `percentStreamed`, `bitRate`, `sampleRate` et `numChannels`.

Pour les acteurs SWA lus en flux continu, les valeurs suivantes sont possibles :

- `0` – Lecture en flux continu de l'acteur interrompue.
- `1` – L'acteur est en cours de rechargement.

- `2` – Préchargement terminé avec succès.
- `3` – Lecture de l'acteur.
- `4` – Interruption de l'acteur (pause).
- `5` – Lecture en flux continu de l'acteur terminée.
- `9` – Une erreur est survenue.
- `10` – Espace processeur insuffisant.

Pour les acteurs animation Flash, cette propriété renvoie une valeur valable uniquement lorsque l'animation Director est exécutée. Les valeurs suivantes sont possibles :

- `0` – L'acteur n'est pas en mémoire.
- `1` – L'en-tête est en cours de chargement.
- `2` – Le téléchargement de l'en-tête est terminé.
- `3` – Le média de l'acteur est en cours de chargement.
- `4` – Le chargement du média de l'acteur est terminé.
- `-1` – Une erreur est survenue.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante émet une alerte si une erreur est détectée pour l'acteur SWA lu en flux continu :

```
-- Syntaxe Lingo
on mouseDown
  if member("Ella Fitzgerald").state = 9 then
    _player.alert("Aucun fichier audio n'a pu être trouvé.")
  end if
end

// Syntaxe JavaScript
function mouseDown() {
  var ellaSt = member("Ella Fitzgerald").state;
  if (ellaSt == 9) {
    _player.alert("Aucun fichier audio n'a pu être trouvé.");
  }
}
```

Exemple

Le script d'image suivant vérifie si une animation Flash intitulée Intro a été complètement transférée en mémoire. Dans la négative, le script affiche l'état courant de l'acteur dans la fenêtre Messages et maintient la tête de lecture en boucle sur l'image courante jusqu'à ce que le chargement de l'animation en mémoire soit terminé.

```
-- Syntaxe Lingo
on exitFrame
  if member("Intro").percentStreamed < 100 then
    put("Etat de téléchargement :" && member("Intro").state)
    _movie.go(_movie.frame)
  end if
end
```

```
// Syntaxe JavaScript
function exitFrame() {
  var intSt = member("Intro").percentStreamed;
  if (intSt < 100) {
    put("Etat de téléchargement : " + member("Intro").state);
    _movie.go(_movie.frame);
  }
}
```

Voir aussi

clearError(), getError() (Flash, SWA)

state (RealMedia)

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.state

// Syntaxe JavaScript
réfObjActeurOuImageObjet.state;
```

Description

Propriété d'acteur ou image-objet RealMedia ; renvoie l'état courant du flux RealMedia, exprimé par un nombre entier compris entre 1 et 4. Chaque valeur d'état correspond à un point spécifique dans le processus de lecture en flux continu. Cette propriété est dynamique en cours de lecture et peut être testée, mais pas définie.

Le processus de lecture en flux continu est initialisé lors de l'entrée de la tête de lecture sur l'image-objet RealMedia dans le scénario, lors de l'appel de la méthode `play` dans une image-objet ou un acteur RealMedia, ou lorsqu'un utilisateur clique sur le bouton Lire dans la fenêtre RealMedia. L'appel de cette propriété renvoie une valeur numérique indiquant l'état du processus de lecture en flux continu de l'acteur RealMedia. A chaque état correspond une ou plusieurs valeur(s) de propriété `mediaStatus (RealMedia, Windows Media)` et chaque valeur `mediaStatus` n'est observée que dans un état. Par exemple, les valeurs `#seeking` et `#buffering` de la propriété `mediaStatus` ne sont présentes que lorsque la valeur de `state` est 3.

La valeur de la propriété `state` fournit d'importantes informations utiles à l'exécution du code sur un acteur. Si la valeur de `member.state` est inférieure à 2, il se peut que certaines propriétés soient incorrectes et, par conséquent, que les commandes dépendant de ces propriétés le soient également. Si la valeur `member.state` est supérieure ou égale à 2 et inférieure à 4, l'acteur RealMedia ne s'affiche pas, mais toutes les propriétés et méthodes auront des valeurs bien définies qui pourront être utilisées pour effectuer des opérations sur l'acteur.

Lors du démarrage du processus de flux continu, la propriété `state` passe par les états suivants, à moins qu'une erreur (-1) ne survienne et empêche le démarrage de la lecture en flux continu :

-1 (erreur) indique qu'un problème est survenu, par exemple une erreur due à des éléments restants du flux RealMedia précédent. Vous obtiendrez des détails supplémentaires en examinant la propriété `lastError`. Cet état est l'équivalent de `#error` pour la propriété `mediaStatus`.

0 (fermé) indique que le flux n'est pas démarré, que les propriétés d'acteur sont toujours dans leur état initial ou qu'elles sont constituées de copies provenant d'une lecture antérieure de l'acteur. Cet état est l'équivalent de `#closed` pour la propriété `mediaStatus`.

1 (connexion) indique que le flux a démarré mais se trouve dans les premières étapes de connexion au serveur, et que les informations disponibles sont encore insuffisantes pour un traitement quelconque de l'acteur. Cet état est l'équivalent de `#connecting` pour la propriété `mediaStatus`.

2 (ouvert) indique que les propriétés ont été actualisées à l'aide du flux actuel. Lorsque la valeur de `state` est supérieure ou égale à 2, les propriétés `height`, `width` et `duration` du flux RealMedia sont connues. Cet état est transitoire et passe rapidement à l'état 3. Cet état est l'équivalent de `#opened` pour la propriété `mediaStatus`.

3 (recherche ou mise en tampon) indique que toutes les propriétés d'acteur RealMedia sont actives, mais que la lecture de l'acteur n'est pas prête à démarrer. Les fenêtres Scène ou RealMedia affichent un rectangle noir ou le logo RealNetworks. Si cet état est le résultat d'une mise en tampon due à une congestion réseau, la valeur de `state` repasse rapidement à `4` (lecture). Cet état est l'équivalent de `#buffering` ou `#seeking` pour la propriété `mediaStatus`.

4 (lecture) indique que le flux RealMedia est en cours de lecture (ou en pause) et qu'aucune erreur ni problème n'est détecté. Il s'agit de l'état correspondant à une lecture normale. Cet état est l'équivalent de `#playing` ou `#paused` pour la propriété `mediaStatus`.

Exemple

Les exemples suivants indiquent que l'état des flux de l'image-objet 2 et de l'acteur Real est 0, qui indique la fermeture.

```
-- Syntaxe Lingo
put(sprite(2).state) -- 0
put(member("Real").state) -- 0

// Syntaxe JavaScript
put(sprite(2).state); // 0
put(member("Real").state); // 0
```

Voir aussi

`mediaStatus (RealMedia, Windows Media)`, `percentBuffered`, `lastError`

static

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.static

// Syntaxe JavaScript
réfObjActeurOuImageObjet.static;
```

Description

Propriété d'acteur et d'image-objet ; associe la performance de lecture d'une image-objet animation Flash à la présence ou non d'éléments d'animation. Si l'animation contient des éléments d'animation (`FALSE`, valeur par défaut) la propriété rafraîchit l'image-objet pour chaque image ; dans le cas contraire (`TRUE`), la propriété ne rafraîchit l'image-objet que lorsqu'elle se déplace ou change de taille.

Cette propriété peut être testée et définie.

Remarque : Vous ne devez régler la propriété `static` sur `TRUE` que lorsque l'image-objet d'animation Flash ne croise pas d'autres images-objets Director en mouvement. Si l'animation Flash croise des images-objets Director en mouvement, elle risque de ne pas être rafraîchie correctement.

Exemple

Le script d'image-objet suivant affiche dans la fenêtre Messages le numéro de piste d'une image-objet animation Flash et indique si celle-ci contient des éléments d'animation :

```
-- Syntaxe Lingo
property spriteNum
  on beginSprite me
  if sprite(spriteNum).static then
    typeDanimation = "ne contient pas d'éléments d'animation."
  else
    typeDanimation = "contient des éléments d'animation."
  end if
  put("L'animation Flash de la piste" && spriteNum && typeDanimation)
end

// Syntaxe JavaScript
function beginSprite() {
  var st = sprite(this.spriteNum).static;
  if (st == 1) {
    typeDanimation = "ne contient pas d'éléments d'animation.";
  } else {
    typeDanimation = "contient des éléments d'animation.";
  }
  trace("L'animation Flash dans la piste " + this.spriteNum + typeDanimation);
}
```

staticQuality

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.staticQuality

// Syntaxe JavaScript
réfObjImageObjet.staticQuality;
```

Description

Propriété d'image-objet QuickTime VR ; indique la qualité de codec utilisée lorsque l'image panoramique est statique. Les valeurs possibles sont `#minQuality`, `#maxQuality` et `#normalQuality`.

Cette propriété peut être testée et définie.

status

Utilisation

```
-- Syntaxe Lingo
réfObjPisteAudio.status

// Syntaxe JavaScript
réfObjPisteAudio.status;
```

Description

Propriété de piste audio ; indique l'état d'une piste audio. En lecture seule.

Les valeurs possibles sont :

Etat	Nom	Signification
0	Idle	Aucun son n'est lu ni placé en file d'attente.
1	Loading	Un son en attente est préchargé mais n'est pas encore lu.
2	Queued	La piste audio a terminé le préchargement d'un son en attente mais ne lit pas encore le son.
3	Playing	Un son est en cours de lecture.
4	Paused	Un son est en pause.

Exemple

L'instruction suivante affiche l'état courant de la piste d'image-objet 2 dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(sound(2).status)

// Syntaxe JavaScript
put(sound(2).status);
```

Voir aussi

Piste audio

stillDown

Utilisation

```
-- Syntaxe Lingo
_mouse.stillDown

// Syntaxe JavaScript
_mouse.stillDown;
```

Description

Propriété de souris ; indique si l'utilisateur appuie sur le bouton de la souris (`TRUE`) ou non (`FALSE`). En lecture seule.

Cette fonction est utile dans les scripts `mouseDown` pour ne déclencher certains événements qu'après l'exécution de la fonction `mouseUp`.

Le script ne peut pas tester `stillDown` si cette fonction est utilisée dans une boucle. Utilisez plutôt la fonction `mouseDown` dans une boucle.

Exemple

L'instruction suivante vérifie si le bouton de la souris est enfoncé et, le cas échéant, appelle le gestionnaire `procédureDeGlissement` :

```
-- Syntaxe Lingo
if (_mouse.stillDown) then
  procédureDeGlissement
end if

// Syntaxe JavaScript
if (_mouse.stillDown) {
  procédureDeGlissement();
}
```

Voir aussi

`Souris`, `mouseDown`, `mouseUp`

stopTime

Utilisation

```
sprite(quelleImageObjet).stopTime
the stopTime of sprite quelleImageObjet
```

Description

Propriété d'image-objet ; détermine le moment auquel l'image-objet vidéo numérique spécifiée s'arrête. La valeur de `stopTime` est exprimée en battements.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante arrête l'image-objet vidéo numérique de la piste 5 à 100 battements :

```
sprite(5).stopTime = 100
```

stopTimeList

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.stopTimeList

// Syntaxe JavaScript
réfObjDvd.stopTimeList;
```

Description

Propriété DVD ; liste de propriétés qui spécifie l'heure ou le chapitre auquel la lecture s'arrête. Lecture/écriture.

`stopTimeList` est une liste de propriétés qui peut être basée sur les chapitres ou sur le temps.

Une liste `stopTimeList` basée sur les chapitres contient les propriétés suivantes :

- `title`. Spécifie le titre.
- `chapter`. Spécifie le chapitre. La lecture s'arrête après la lecture de ce chapitre.

Une liste `stopTimeList` basée sur le temps contient les propriétés suivantes :

- `title`. Spécifie le titre.
- `hours`. Spécifie l'heure à laquelle la lecture s'arrête.
- `min`. Spécifie la minute à laquelle la lecture s'arrête.
- `sec`. Spécifie la seconde à laquelle la lecture s'arrête.
- `frames`. Spécifie les images sur lesquelles la lecture s'arrête.

La liste `stopTimeList` peut être effacée en la définissant sur 0.

Exemple

Cette liste `stopTimeList` arrête la lecture au chapitre 4 du titre 1 :

```
[#title:1, #chapter:4]
```

Cette liste `stopTimeList` arrête la lecture à un moment spécifique du titre 1 :

```
[#title:1, #hours:0, #minutes:55, #seconds:45, #frames:15]
```

Cette liste `stopTimeList` ne contient qu'un seul paramètre de temps :

```
[#title:1, #seconds:45]
```

Voir aussi

`DVD`

streamMode

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.streamMode

// Syntaxe JavaScript
réfObjActeur.streamMode;
```

Description

Propriété d'acteur Flash ; contrôle le mode de transfert en mémoire d'un acteur animation Flash liée, de la manière suivante :

- `#frame` (valeur par défaut) – Transfère une partie de l'acteur à chaque fois que l'image Director avance pendant que l'image-objet est sur la scène.
- `#idle` – Transfère une partie de l'acteur à chaque fois qu'un événement `idle` est généré ou au moins une fois par image Director pendant que l'image-objet est sur la scène.
- `#manual` – Transfère une partie de l'acteur en mémoire uniquement lorsque la commande `stream` est émise pour cet acteur.

Cette propriété peut être testée et définie.

Exemple

Le script `startMovie` suivant effectue une recherche d'acteurs animation Flash dans la distribution interne et donne à leur propriété `streamMode` la valeur `#manual` :

```
-- Syntaxe Lingo
on startMovie
  repeat with i = 1 to castLib(1).member.count
    if member(i, 1).type = #flash then
      member(i, 1).streamMode = #manual
    end if
  end repeat
end

// Syntaxe JavaScript
function startMovie() {
  i = 1;
  while( i < (castLib(quelleDistribution).member.count) + 1) {
    var tp = member(i, quelleDistribution).toString();
    if (tp = "#flash") {
      member(i, 1).streamMode = symbol("manual");
      i++;
    }
  }
}
```

streamName

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.streamName

// Syntaxe JavaScript
réfObjActeur.streamName;
```

Description

Propriété d'acteur Shockwave Audio (SWA) ; spécifie une adresse URL ou un nom de fichier pour un acteur SWA lu en flux continu. Cette propriété fonctionne comme la propriété d'acteur `URL`.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante lie le fichier Orchestre.swa à un acteur SWA lu en flux continu. Le fichier lié se trouve sur le disque nommé monDisque dans le dossier Sons.

```
-- Syntaxe Lingo
member("fluxSWA").streamName = "monDisque/Sons/Orchestre.swa"

// Syntaxe JavaScript
member("fluxSWA").streamName = "monDisque/Sons/Orchestre.swa";
```

streamSize

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.streamSize

// Syntaxe JavaScript
réfObjActeur.streamSize;
```

Description

Propriété d'acteur ; indique un nombre entier correspondant au nombre total d'octets à transférer pour l'acteur spécifié. La propriété `streamSize` ne renvoie une valeur que lorsque l'animation Director est en cours de lecture.

Cette propriété peut être testée, mais pas définie.

Exemple

Le script d'image suivant vérifie si une animation Flash intitulée Intro a été complètement transférée en mémoire. Dans la négative, le script met à jour un acteur champ pour indiquer le nombre d'octets déjà chargés (à l'aide de la propriété d'acteur `bytesStreamed`) et le nombre total d'octets de l'acteur (grâce à la propriété d'acteur `streamSize`). Le script contraint la tête de lecture à passer en boucle sur l'image courante jusqu'à ce que l'animation soit complètement chargée en mémoire.

```
-- Syntaxe Lingo
on exitFrame
  if member("Intro").percentStreamed < 100 then
    member("message").text = \
  string(member("Intro").bytesStreamed) && "sur" && \
  string(member("Intro").streamSize) && \
  "octets ont été téléchargés."
    _movie.go(_movie.frame)
  end if
end

// Syntaxe JavaScript
function exitFrame() {
  var pctStm = member("Intro").percentStreamed;
  var strSs = new String(member("Intro").streamSize);
  var strIm = new String(member("Intro").bytesStreamed);
  if (pctStm < 100) {
    member("message").text = strStm + " sur " + strSS  +
  " octets ont été téléchargés.";
  _ movie.go(_movie.frame);
  }
}
```

streamSize (3D)

Utilisation

```
member(quelActeur).streamSize
```

Description

Propriété 3D ; permet d'obtenir la taille du flux de données à télécharger, de 0 au maximum. Cette commande fait référence à l'importation du fichier initial ou à la dernière commande `loadFile()` demandée.

Exemple

L'instruction suivante indique que le dernier chargement de fichier associé à l'acteur Séquence a une taille totale de 325300 octets.

```
put member("Séquence").streamSize
-- 325300
```

Voir aussi

`bytesStreamed (3D)`, `percentStreamed (3D)`, `state (3D)`, `preLoad (3D)`

strokeColor

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.strokeColor

// Syntaxe JavaScript
réfObjActeur.strokeColor;
```

Description

Propriété d'acteur forme vectorielle ; indique la couleur RVB du trait formant les contours de la forme.

Vous pourrez voir un exemple de `strokeColor` dans une animation en consultant l'animation Vector Shapes du dossier Learning/Lingo, lui-même dans le dossier de Director.

Exemple

L'instruction suivante donne à la propriété strokeColor de l'acteur Trait la couleur rouge :

```
-- Syntaxe Lingo
member("Trait").strokeColor = color(255, 0, 0)

// Syntaxe JavaScript
member("Trait").strokeColor = color(255, 0, 0);
```

Voir aussi

`color()`, `fillColor`, `endColor`, `backgroundColor`

strokeWidth

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.strokeWidth

// Syntaxe JavaScript
réfObjActeur.strokeWidth;
```

Description

Propriété d'acteur forme vectorielle ; indique l'épaisseur, en pixels, du trait formant les contours de la forme.

Cette valeur est un nombre à virgule flottante compris entre 0 et 100 et peut être testée et définie.

Vous pourrez voir un exemple de `strokeWidth` dans une animation en consultant l'animation Vector Shapes du dossier Learning/Lingo, lui-même dans le dossier de Director.

Exemple

L'instruction suivante donne à la propriété `strokeWidth` de l'acteur Trait la valeur de 10 pixels :

```
-- Syntaxe Lingo
member("Trait").strokeWidth = 10

// Syntaxe JavaScript
member("trait").strokeWidth = 10;
```

style

Utilisation

```
member(quelActeur).model(quelModèle).toon.style
member(quelActeur).model(quelModèle).shader.style
member(quelActeur).shader(quelMatériau).style
```

Description

Propriété 3D de modificateur `toon` et de matériau `painter` ; indique la façon dont le modificateur `toon` et le matériau `painter` appliquent la couleur à un modèle. Les valeurs possibles sont :

- `#toon` utilise des transitions aiguës entre les couleurs.
- `#gradient` utilise des transitions fluides entre les couleurs. C'est la valeur par défaut.
- `#blackAndWhite` utilise les couleurs noir et blanc.

Le nombre de couleurs utilisées par le modificateur `toon` et le matériau `painter` est défini avec la propriété `colorSteps` du modificateur ou du matériau.

Exemple

L'instruction suivante donne à la propriété `style` du modificateur `toon` du modèle Théière la valeur `#blackAndWhite`. Le modèle sera rendu en noir et blanc.

```
member("Formes").model("Théière").toon.style = #blackAndWhite
```

subdivision

Utilisation

```
member(quelActeur).model(quelModèle).sds.subdivision
```

Description

Propriété 3D de modificateur `sds` ; permet d'obtenir ou de définir le mode opérationnel du modificateur de fractionnement. Les valeurs possibles sont :

- `#uniform` spécifie que la maille est mise à l'échelle en détail de manière uniforme, chaque face étant fractionnée le même nombre de fois.
- `#adaptive` spécifie que des détails supplémentaires ne sont ajoutés qu'en présence d'un changement d'orientation majeur et seulement aux zones actuellement visibles de la maille.

Le modificateur `sds` ne peut pas être utilisé avec les modificateurs `inker` ou `toon` ; vous devrez également faire preuve de prudence lors de l'utilisation du modificateur `sds` avec le modificateur `lod`. Pour plus d'informations, consultez l'entrée du modificateur `sds`.

Exemple

L'instruction suivante donne à la propriété `subdivision` du modificateur `sds` du modèle Bébé la valeur `#adaptive`. La géométrie de Bébé ne sera pas modifiée de façon uniforme.

```
member("Séquence").model("Bébé").sds.subdivision = #adaptive
```

Voir aussi

`sds (modificateur)`, `error`, `enabled (sds)`, `depth (3D)`, `tension`

subPicture

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.subPicture

// Syntaxe JavaScript
réfObjDvd.subPicture;
```

Description

Propriété DVD. Détermine la sous-image courante, si elle existe. Lecture/écriture.

La valeur de `subPicture` est un nombre entier. La valeur 0 désactive `subPicture`.

Voir aussi

`DVD`

subPictureCount

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.subPictureCount

// Syntaxe JavaScript
réfObjDvd.subPictureCount;
```

Description

Propriété DVD. Renvoie le nombre de sous-images disponibles. En lecture seule.

Voir aussi

DVD

suspendUpdates

Utilisation

```
sprite(quelleImageObjet3D).suspendUpdates
```

Description

Propriété 3D d'image-objet ; lorsque définie sur TRUE, empêche la mise à jour de l'image-objet dans le cadre des opérations de rafraîchissement normales de l'écran. Ceci peut améliorer les performances de lecture de l'animation. Certains types d'actualisation d'écran pourront encore affecter l'image-objet, par exemple ceux qui sont engendrés par le glissement d'une autre fenêtre sur l'image-objet. Lorsque la propriété suspendUpdates est définie sur FALSE, l'image-objet est redessinée normalement.

Il est fondamental que la propriété suspendUpdates garde la valeur FALSE lors de l'animation de tout élément dans l'image-objet 3D.

switchColorDepth

Utilisation

```
-- Syntaxe Lingo
_player.switchColorDepth

// Syntaxe JavaScript
_player.switchColorDepth;
```

Description

Propriété de lecteur ; détermine si Director change le codage des couleurs du moniteur que la scène occupe conformément au codage des couleurs de l'animation en cours de chargement (TRUE) ou laisse le codage des couleurs du moniteur inchangé lors du chargement de l'animation (FALSE). Lecture/écriture.

Si la propriété switchColorDepth a la valeur TRUE, rien ne se passe tant qu'une nouvelle animation n'est pas chargée.

Il est recommandé d'adapter le codage des couleurs du moniteur à celui de l'animation.

- Si le réglage du codage des couleurs du moniteur est inférieur à celui de l'animation, le fait de le faire passer aux paramètres définis pour l'animation (en supposant que le moniteur puisse fournir ce type de codage) permet de conserver l'aspect initial de l'animation.

- Si le codage des couleurs du moniteur est supérieur à celui de l'animation, la réduction du codage permet d'utiliser moins de mémoire pour exécuter les animations, de charger les acteurs de manière plus efficace et d'accélérer l'animation.

La valeur de cette propriété peut également être définie avec l'option Adapter le moniteur aux couleurs de l'animation dans la boîte de dialogue Préférences générales.

Exemple

L'instruction suivante affecte à la variable intitulée `interrupteur` le paramètre courant de `switchColorDepth` :

```
-- Syntaxe Lingo
interrupteur = _player.switchColorDepth

// Syntaxe JavaScript
var interrupteur = _player.switchColorDepth;
```

L'instruction suivante vérifie si le codage des couleurs courant est de 8 bits et, le cas échéant, active la propriété `switchColorDepth` :

```
-- Syntaxe Lingo
if (_system.colorDepth = 8) then
  _player.switchColorDepth = TRUE
end if

// Syntaxe JavaScript
if (_system.colorDepth == 8) {
  _player.switchColorDepth = true;
}
```

Voir aussi

`colorDepth`, `Lecteur`

systemTrayIcon

Utilisation

```
-- Syntaxe Lingo
_movie.displayTemplate.systemTrayIcon
réfObjFenêtre.systemTrayIcon

// Syntaxe JavaScript
_movie.displayTemplate.systemTrayIcon;
réfObjFenêtre.systemTrayIcon;
```

Description

Propriété d'animation et de fenêtre (Microsoft Windows uniquement). Détermine si une icône est associée à une fenêtre dans la barre d'état système du bureau d'un utilisateur. Lecture/écriture.

Si `systemTrayIcon` a la valeur `TRUE`, une icône de fenêtre est placée dans la barre d'état système.

Si `systemTrayIcon` a la valeur `FALSE`, aucune icône n'apparaît dans la barre d'état système.

Voir aussi

`displayTemplate`, `Animation`, `systemTrayTooltip`, `Fenêtre`

systemTrayTooltip

Utilisation

```
-- Syntaxe Lingo
_movie.displayTemplate.systemTrayTooltip
réfObjFenêtre.systemTrayTooltip

// Syntaxe JavaScript
_movie.displayTemplate.systemTrayTooltip;
réfObjFenêtre.systemTrayTooltip;
```

Description

Propriété d'animation et de fenêtre (Microsoft Windows uniquement). Détermine la chaîne qui s'affiche dans l'info-bulle de l'icône de la barre d'état système. Lecture/écriture.

Cette propriété n'est applicable que si `systemTrayIcon` a pour valeur `TRUE`. Si `systemTrayIcon` a la valeur `TRUE`, l'info-bulle apparaîtra lorsqu'un utilisateur déplacera la souris sur la barre d'état système.

La valeur par défaut de `systemTrayTooltip` est le titre de la fenêtre.

Voir aussi

`displayTemplate`, `Animation`, `systemTrayIcon`, `Fenêtre`

tabCount

Utilisation

`expressionSousChaîne``.tabCount`

Description

Propriété d'acteur texte ; indique le nombre d'arrêts de tabulation uniques présents dans l'expression de sous-chaîne de l'acteur texte.

La valeur est un nombre entier égal ou supérieur à 0 et peut être testée, mais pas définie.

tabs

Utilisation

`member(`*`quelActeurTexte`*`).tabs`

Description

Propriété d'acteur texte ; contient une liste de tous les arrêts de tabulations définis dans l'acteur texte.

Chaque élément de la liste comporte des informations concernant ces tabulations pour l'acteur. Les propriétés possibles de la liste sont les suivantes :

`#type`	Peut correspondre à `#left` (gauche), `#center` (centre), `#right` (droite) ou `#decimal` (décimale).
`#position`	Valeur entière indiquant la position de la tabulation en points.

Cette propriété peut être testée et définie. Lorsque la propriété `tabs` est définie, la propriété `type` devient facultative. Si la propriété `type` n'est pas définie, le type de tabulation passe par défaut à `#left` (gauche).

Exemple

L'instruction suivante récupère toutes les tabulations de l'acteur texte Crédits et les affiche dans la fenêtre Messages :

```
put member("Crédits").tabs
-- [[#type: #left, #position: 36], [#type: #Decimal, #position: 141],  \
  [#type: #right, #position: 216]]
```

target

Utilisation

objetDeTemporisation.target

Description

Propriété d'objet de temporisation ; indique l'objet enfant auquel l'*objetDeTemporisation* donné enverra ses événements de temporisation. Les objets de temporisation dont la propriété cible est `VOID` enverront leurs événements à un gestionnaire dans un script d'animation.

Cette propriété s'avère pratique pour le débogage des comportements et des scripts parents utilisant les objets de temporisation.

Exemple

L'instruction suivante affiche le nom de l'objet enfant qui recevra les événements de temporisation de l'objet de temporisation `minuteur1` dans la fenêtre Messages :

```
put timeout("minuteur1").target
```

Voir aussi

`name (temporisation)`, `timeout()`, `timeoutHandler`, `timeoutList`

targetFrameRate

Utilisation

sprite(*quelleImageObjet3D*).targetFrameRate

Description

Propriété 3D d'image-objet ; détermine le nombre d'images par seconde à utiliser lors du rendu d'une image-objet 3D. La valeur par défaut est de 30 images par seconde. La propriété `targetFrameRate` n'est utilisée que si la propriété `useTargetFrameRate` est définie sur `TRUE`. Si la propriété `useTargetFrameRate` a pour valeur `TRUE`, le nombre de polygones des modèles de l'image-objet est réduit pour atteindre le taux d'images spécifié ciblé.

Exemple

Ces instructions donnent à la propriété `targetFrameRate` de l'image-objet 3 la valeur 45 et appliquent la cadence d'image en donnant à la propriété `useTargetFrameRate` de l'image-objet la valeur `TRUE` :

```
sprite(3).targetFrameRate = 45
sprite(3).useTargetFrameRate = TRUE
```

Voir aussi

useTargetFrameRate

tension

Utilisation

```
member(quelActeur).model(quelModèle).sds.tension
```

Description

Propriété 3D de modificateur `sds` ; permet d'obtenir ou de définir un pourcentage à virgule flottante compris entre 0.0 et 100.0 pour déterminer à quel point la surface nouvellement générée doit refléter la surface d'origine. Plus cette valeur est élevée, plus les surfaces fractionnées correspondent à la surface d'origine. La valeur par défaut est 65,0.

Exemple

L'instruction suivante donne à la propriété `tension` du modificateur `sds` du modèle Bébé la valeur 35. Si le paramètre `error` du modificateur `sds` est trop faible et que son paramètre `depth` est trop élevé, cette instruction a un effet très prononcé sur la géométrie de Bébé.

```
member("Séquence").model("Bébé").sds.tension = 35
```

Voir aussi

sds (modificateur), error, depth (3D)

text

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.text

// Syntaxe JavaScript
réfObjActeur.text;
```

Description

Propriété d'acteur texte ; détermine la chaîne de caractère dans l'acteur champ spécifié par *quelActeur*.

La propriété d'acteur `text` est utile pour afficher des messages et enregistrer ce que saisit l'utilisateur.

Cette propriété peut être testée et définie.

Les changements apportés par Lingo ou la syntaxe JavaScript au texte d'un acteur suppriment tout formatage que vous auriez pu appliquer aux mots ou lignes. La modification de la propriété d'acteur `text` applique à nouveau le formatage global. Pour changer des portions de texte particulières, vous devez faire référence aux lignes, mots ou éléments qu'il contient.

Lorsque l'animation est lue sous forme d'applet, la valeur de cette propriété est `""` (une chaîne vide) pour un acteur champ dont le texte n'a pas encore été lu en flux continu.

Vous pourrez voir un exemple de `text` dans une animation en consultant les animations Forms and Post et Text du dossier Learning/Lingo, lui-même dans le dossier de Director.

Exemple

L'instruction suivante place Merci dans l'acteur vide Réponse :

```
-- Syntaxe Lingo
if (member("Réponse").text = EMPTY) then
  member("Réponse").text = "Merci."
end if

// Syntaxe JavaScript
if (member("Réponse").text == " ") {
  member("Réponse").text = "Merci.";
}
```

L'instruction suivante affecte à l'acteur Remarque la phrase « Vous avez pris la bonne décision ! » :

```
-- Syntaxe Lingo
member("Remarque").text = "Vous avez pris la bonne décision !"

// Syntaxe JavaScript
member("Remarque").text = "Vous avez pris la bonne décision!";
```

Voir aussi

`selEnd`, `selStart`

texture

Utilisation

```
member(quelActeur).texture(quelleTexture)
member(quelActeur).texture[index]
member(quelActeur).shader(quelMatériau).texture
member(quelActeur).model(quelModèle).shader.texture
member(quelActeur).model(quelModèle).shaderList.texture
member(quelActeur).model(quelModèle).shaderList[index].texture
member(quelActeur).modelResource(quelModèleDeSystèmeDeParticules\
  Ressource).texture
```

Description

Propriété 3D d'élément et de matériau ; objet image utilisé par un matériau pour définir l'apparence de la surface d'un modèle. L'image est enrobée sur la géométrie du modèle par le matériau.

Le composant visible d'un matériau est créé avec un maximum de huit couches de textures. Ces huit couches de texture ont été créées à partir d'acteurs bitmap ou d'objets image dans Director ou importées avec des modèles de programmes de modélisation 3D.

Créez et supprimez des textures avec les commandes `newTexture()` et `deleteTexture()`.

Les textures sont enregistrées dans la palette des textures de l'acteur 3D. Elles peuvent être référencées par nom (*`quelleTexture`*) ou par index de palette (*`indexDeTexture`*). Une texture peut être utilisée par n'importe quel nombre de matériaux. Les modifications apportées à une texture apparaissent dans tous les matériaux qui l'utilisent.

Il existe trois types de textures :

`#fromCastmember` ; la texture est créée à partir d'un acteur bitmap avec la commande `newTexture()`.

`#fromImageObject` ; la texture est créée à partir d'un objet image Lingo avec la commande `newTexture()`.

`#importedFromFile` ; la texture est importée avec un modèle à partir d'un programme de modélisation 3D.

Vous trouverez plus d'informations sur les propriétés des textures dans la rubrique Utilisation de Director du panneau d'aide de Director.

La texture d'un système de particules est une propriété de la ressource de modèle, dont le type est `#particle`.

Exemple

L'instruction suivante donne à la propriété `texture` du matériau surfaceDuMur la propriété peintureBleue.

```
member("Pièce").shader("surfaceDuMur").texture = \
  member("Pièce").texture("peintureBleue")
```

Voir aussi

`newTexture`, `deleteTexture`

textureCoordinateList

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).
textureCoordinateList
référenceDobjetDeRessourceDeModèle.textureCoordinateList
```

Description

Propriété 3D ; utilisée avec une ressource de modèle de type `#mesh`, ou avec un modificateur `meshDeform` associé à un modèle, cette propriété permet d'obtenir ou de définir la propriété `textureCoordinateList` de la ressource de modèle.

La propriété `textureCoordinateList` est une liste de sous-listes identifiant les positions à utiliser dans une image pour le placage de texture sur un triangle. Chaque sous-liste est composée de deux valeurs indiquant une position dans une texture. Ces valeurs doivent être comprises entre 0.0 et 1.0 pour pouvoir être redimensionnées en textures de taille arbitraire. La valeur par défaut est une liste vide.

Manipulez `modelResource.face[index].textureCoordinates` ou `model.meshdeform.mesh[index].face[index]` pour changer la correspondance entre `textureCoordinates` et les coins de la face de la maille.

Exemple

```
put member(5,2).modelResource("carré de maille").\
  textureCoordinateList
--[ [0.1, 0.1], [0.2, 0.1], [0.3, 0.1], [0.1, 0.2], [0.2, 0.2], \
  [0.3, 0.2], [0.1, 0.3], [0.2, 0.3], [0.3, 0.3] ]
```

Voir aussi

face, texture, meshDeform (modificateur)

textureCoordinates

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  face[indexDeFaces].textureCoordinates
objetDeRessourceDeModèle.face[indexDeFace].textureCoordinates
```

Description

Propriété 3D ; identifie les éléments de la `textureCoordinateList` utilisés pour la face de *indexDeFace*. Cette propriété doit être une liste de trois entiers spécifiant les index de la `textureCoordinateList`, correspondant aux coordonnées de texture à utiliser pour chaque coin de la face de la maille.

Voir aussi

face, textureCoordinateList

textureLayer

Utilisation

```
member(quelActeur).model(quelModèle).meshDeform.mesh[index].\
  textureLayer.count
member(quelActeur).model(quelModèle).meshdeform.mesh[index].\
  texturelayer.add()
member(quelActeur).model(quelModèle).meshdeform.mesh[index].\
  texturelayer[index].textureCoordinateList.
```

Description

Propriétés 3D de modificateur `meshdeform` ; ces propriétés permettent d'obtenir et de définir les informations concernant les couches de texture d'une maille spécifiée.

Un matériau peut contenir jusqu'à huit couches de texture, chaque couche ne pouvant contenir qu'une seule texture, la même texture pouvant être spécifiée pour plus d'une couche. Les couches de texture sont celles utilisées par les matériaux.

Utilisez les propriétés suivantes pour accéder aux couches de texture et les manipuler :

`meshdeform.mesh[index].texturelayer.count` renvoie le nombre de couches de texture pour la maille spécifiée.

`model.meshdeform.mesh[index].texturelayer.add()` ajoute une couche de texture vide à la maille spécifiée.

`model.meshdeform.mesh[index].texturelayer[index].texturecoordinatelist` permet d'obtenir ou de définir une liste de coordonnées de texture pour une couche de la maille spécifiée. Vous pouvez également utiliser cette propriété pour copier les coordonnées de texture entre différentes couches, tel qu'indiqué ci-dessous :

```
model.meshdeform.texturelayer[a].texturecoordinatelist = \
  model.meshdeform.texturelayer[b].texturecoordinatelist
```

Voir aussi

meshDeform (modificateur), mesh (propriété), textureCoordinateList, add (texture 3D), count, texture, textureModeList

textureList

Utilisation

```
member(quelActeur).model(quelModèle).shader(quelMatériau).textureList
member(quelActeur).model(quelModèle).shader(quelMatériau).textureList[index]
```

Description

Propriété 3D de matériau ; détermine la liste des textures appliquées au matériau. Un matériau peut utiliser jusqu'à huit couches de texture. Lorsque testée, cette propriété renvoie une liste linéaire d'objets de texture. Lorsque définie sans spécifier d'index, cette propriété spécifie un objet de texture à appliquer à toutes les couches. La définition de la propriété `textureList` sur `VOID` désactive l'application de textures pour toutes les couches. La valeur par défaut est `VOID`.

Pour tester ou définir l'objet de texture d'une couche de texture spécifique, vous devez inclure une valeur d'index.

Exemple

L'instruction suivante définit la troisième couche de texture du matériau appelé SurfaceDuMur sur la texture appelée PeintureBleue dans l'acteur Pièce.

```
member(3).model("Voiture").shader("surfaceDuMur").textureList[3] = \
  member("Pièce").texture("peintureBleue")
```

Voir aussi

textureModeList

textureMember

Utilisation

```
member(quelActeur).textureMember
```

Description

Propriété 3D d'acteur ; indique le nom de l'acteur bitmap utilisé comme source de la texture par défaut pour l'acteur 3D.

La propriété `textureType` de l'acteur 3D doit avoir pour valeur `#member` afin d'activer la propriété `textureMember`.

Exemple

L'instruction suivante donne à la propriété `textureMember` de l'acteur scèneDeJardin la valeur Haie. Si la propriété `textureType` de scèneDeJardin a pour valeur `#member`, l'acteur Haie devient le bitmap source pour la texture par défaut de scèneDeJardin.

```
member("scèneDeJardin").textureMember = "Haie"
```

Voir aussi

`textureType`

textureMode

Utilisation

```
member(quelActeur).shader(quelMatériau).textureMode
member(quelActeur).model(quelModèle).shader.textureMode
member(quelActeur).model(quelModèle).shaderList{[index]}.\
  textureMode
```

Description

Propriété 3D de matériau `#standard` ; spécifie la façon dont la première couche de texture est plaquée sur la surface du modèle. Utilisez la propriété `textureModeList` pour spécifier les textures des couches autres que la première. Cette propriété est ignorée si le modificateur `#toon` est appliqué à la ressource de modèle.

Les valeurs possibles de cette propriété sont `#none`, `#wrapPlanar`, `#wrapCylindrical`, `#wrapSpherical`, `#reflection`, `#diffuseLight` et `#specularLight`. Une description de ces termes est donnée dans la section `textureModeList`.

Exemple

L'instruction suivante donne à la propriété `textureMode` de la première couche de texture du matériau du modèle Balle la valeur `#wrapSpherical`.

```
member("Séquence").model("Balle").shader.textureMode = #wrapSpherical
```

Voir aussi

`textureModeList`

textureModeList

Utilisation

```
member(quelActeur).shader(quelMatériau).textureModeList
member(quelActeur).shader(quelMatériau).
textureModeList[indexDeCoucheDeTexture]
member(quelActeur).model(quelModèle).shader.textureModeList
member(quelActeur).model(quelModèle).shader.
textureModeList[indexDeCoucheDeTexture]
```

Description

Propriété 3D de matériau standard ; permet de changer la façon dont une couche de texture est plaquée sur la surface d'un modèle. Cette propriété est ignorée si le modificateur `#toon` est appliqué à la ressource de modèle.

Les valeurs possibles sont :

- `#none` utilise les valeurs de coordonnées de texture définies à l'origine pour la ressource de modèle. Ce paramètre désactive `wrapTransform` et `wrapTransformList [indexDeCoucheDeTexture]`.
- `#wrapPlanar` enrobe la surface du modèle avec la texture comme s'il s'agissait d'une rétroprojection. La `wrapTransformList[indexDeCoucheDeTexture]` du matériau est appliquée à l'espace de placage avant la génération des coordonnées de texture. Avec une `wrapTransformList[indexDeCoucheDeTexture]` d'identité (la valeur par défaut), le placage planaire est orienté de façon à ce que la texture soit extrudée le long de l'axe des z, avec le haut de la texture le long de l'axe des y.
- `#wrapCylindrical` enrobe la texture autour de la surface comme si la surface était placée au milieu de la texture avant d'être enroulée autour de la surface pour former un cylindre. La `wrapTransformList[textureLayerIndex]` est appliquée à l'espace de placage avant la génération des coordonnées de texture. Avec une `wrapTransformList [indexDeCoucheDeTexture]` d'identité (la valeur par défaut), le placage cylindrique est orienté de façon à ce que la texture soit enrobée depuis l'axe des y négatif, en commençant du bord gauche de la texture et en allant vers l'axe des x positif autour de l'axe des z. Le haut de la texture suit l'axe des z positif.
- `#wrapSpherical` enrobe la texture autour de la surface comme si la surface était placée au milieu de la texture avant de voir ses quatre coins tirés pour se rencontrer en haut. La `wrapTransformList[textureLayerIndex]` est appliquée à l'espace de placage avant la génération des coordonnées de texture. Avec une `wrapTransformList [indexDeCoucheDeTexture]` d'identité, le placage sphérique est placé à l'origine de l'espace du modèle et est orienté de façon à ce que la texture soit enrobée depuis l'axe des y négatif, en commençant du bord gauche de la texture et en allant vers l'axe des x autour de l'axe des z. Le haut de la texture suit l'axe des z positif.
- `#reflection` est similaire à `#wrapSpherical`, à l'exception du fait que les nouvelles coordonnées de textures sont continuellement projetées sur la surface à partir d'un point fixe. Les coordonnées de texture ne pivotent pas en même temps que le modèle. Simule la lumière réfléchie sur un objet par son environnement. Ce paramètre désactive `wrapTransform`.
- `#diffuseLight` génère des valeurs de coordonnées de texture de lumière diffuse, une par sommet, et enregistre les résultats dans la maille référencée. Ce paramètre désactive `wrapTransform`.
- `#specularLight` génère des valeurs de coordonnées de texture de lumière spéculaire, une par sommet, et enregistre les résultats dans la maille référencée. Ce paramètre désactive `wrapTransform`.

Exemple

Dans l'exemple suivant, un matériau est configuré pour simuler une balle de jardin avec des reflets. La première couche de texture du matériau est un placage sphérique et la troisième couche utilise un placage de style `#reflection`. L'entrée `textureList[3]` semblera être reflétée à partir de l'environnement sur tous les modèles qui utilisent le matériau.

```
member("Séquence").shader("balleDeJardin).textureList[1] = \
  member("Séquence").texture("balleBrillante")
member("Séquence").shader("balleDeJardin").textureModeList[1] = \
  #wrapSpherical
member("Séquence").shader("balleDeJardin").textureList[3] = \
  member("Séquence").texture("environnementDeJardin")
member("Séquence").shader("balleDeJardin").textureModeList[3] = \
  #reflection
```

Voir aussi

`textureTransformList`, `wrapTransform`

textureRenderFormat

Utilisation

```
getRenderServices().textureRenderFormat
```

Description

Propriété 3D de `rendererServices` ; permet d'obtenir ou de définir le format binaire utilisé par toutes les textures des acteurs 3D. Utilisez la propriété `textureRenderFormat` pour annuler ce paramètre pour certaines textures uniquement. Les formats binaires plus réduits (par exemple, les variantes 16 bits telles que `#rgba5551`) utilisent une quantité plus réduite de RAM vidéo de l'accélérateur matériel, ce qui permet d'utiliser un plus grand nombre de textures avant d'être forcé de passer au rendu logiciel. Les formats binaires plus élevés (par exemple, les variantes 32 bits telles que `#rgba8888`) offrent généralement un meilleur résultat. Pour utiliser la transparence alpha dans une texture, le dernier bit ne doit pas être nul. Pour obtenir un bon effet de transparence, le canal alpha doit avoir une précision supérieure à un bit.

Les formats de pixels comptent chacun quatre chiffres, chaque chiffre indiquant le degré de précision pour le rouge, le vert, le bleu et l'alpha. La valeur choisie détermine la précision des couleurs (la précision du canal alpha) et la quantité de mémoire utilisée par le tampon des textures du matériel. Vous pouvez choisir une valeur qui améliore la fidélité des couleurs ou qui vous permet de mettre plus de textures sur la carte. Vous pouvez mettre deux fois plus de textures 16 bits que de textures 32 bits dans le même espace. Director passe au rendu `#software` lorsqu'une animation utilise plus de textures que la carte ne peut en contenir.

Vous pouvez spécifier n'importe laquelle des valeurs suivantes pour `textureRenderFormat` :

- `#rgba8888` : mode de couleur 32 bits avec 8 bits chaque pour rouge, vert, bleu et alpha.
- `#rgba8880` : comme ci-dessus, sans valeur alpha.
- `#rgba5650` : mode de couleur 16 bits sans alpha ; 5 bits pour rouge, 6 bits pour vert et 5 bits pour bleu.
- `#rgba5550` : mode de couleur 16 bits sans alpha ; 5 bits chaque pour rouge, vert et bleu, sans mesure alpha.
- `#rgba5551` : mode de couleur 16 bits avec 5 bits chaque pour rouge, vert et bleu ; 1 bit pour alpha.

- `#rgba4444` : mode de couleur 16 bits avec 4 bits chaque pour rouge, vert, bleu et alpha.

La valeur par défaut est `#rgba5551`.

Exemple

L'instruction suivante donne à la propriété globale `textureRenderFormat` de l'acteur 3D la valeur `#rgba8888`. Chaque texture de cette animation sera rendue avec des couleurs 32 bits à moins que sa propriété `textureRenderFormat` ait une valeur autre que `#default`.

```
getRendererServices().textureRenderFormat = #rgba8888
```

Voir aussi

`renderer`, `preferred3dRenderer`, `renderFormat`, `getRendererServices()`

textureRepeat

Utilisation

```
member(quelActeur).shader(quelMatériau).textureRepeat
member(quelActeur).model(quelModèle).shader.textureRepeat
member(quelActeur).model(quelModèle).shaderList{[index]}.\
  textureRepeat
```

Description

Propriété 3D de matériau `#standard` ; contrôle le comportement de verrouillage de la première couche de texture du matériau. Utilisez la propriété `textureRepeatList` pour contrôler cette propriété pour les couches de texture autres que la première.

Lorsque `textureRepeat` a pour valeur `TRUE` et que la valeur des composants x et/ou y de `référenceDeMatériau.textureTransform.scale` est inférieure à 1, la texture est juxtaposée sur la surface du modèle.

Lorsque `textureRepeat` a pour valeur `FALSE`, la texture n'est pas juxtaposée. Si la valeur des composants x et/ou y de `référenceDeMatériau.textureTransform.scale` est inférieure à 1, les régions du modèle qui ne sont pas recouvertes par la texture sont noires. Si la valeur des composants x et/ou y de `référenceDeMatériau.textureTransform.scale` est supérieure à 1, les portions de la texture dépassant la plage de coordonnées sont rognées.

La valeur par défaut de cette propriété est `TRUE`. Cette propriété est toujours `TRUE` avec le moteur de rendu `#software`.

Exemple

L'instruction suivante donne à la propriété `textureRepeat` du premier matériau utilisé par le modèle gbCyl3 la valeur `TRUE`. La première texture de ce matériau sera juxtaposée si la valeur du composant x ou y de sa propriété `textureTransform` ou `textureTransformList` est inférieure à 1.

```
member("séquence").model("gbCyl3").shader.textureRepeat = TRUE
```

Voir aussi

`textureTransform`, `textureTransformList`

textureRepeatList

Utilisation

référence
eListeDeMatériaux]. textureRepeatList[*indexDeCoucheDeTexture*]

Description

Propriété 3D de matériau standard ; permet d'obtenir ou de définir le comportement de verrouillage de texture de toute couche de texture. Lorsque `TRUE`, la valeur par défaut, la texture de `indexDeCoucheDeTexture` peut être juxtaposée plusieurs fois sur les surfaces du modèle. Cela peut être obtenu en donnant à `référenceDeMatériau.textureTransform[indexDeCoucheDeTexture].scale` une valeur inférieure à 1 dans x ou y. Lorsque la valeur est FALSE, la texture est appliquée à une portion plus réduite des surfaces plutôt que juxtaposée lorsque `référenceDeMatériau.textureTransform[indexDeCoucheDeTexture].scale` est inférieure à 1 dans x ou y. Une illustration de cet effet serait la réduction de l'image source dans le cadre de l'image d'origine et le remplissage de l'espace en noir. De même, si `référenceDeMatériau.textureTransform[indexDeCoucheDeTexture].scale` est supérieure à 1 dans x ou y, les portions de la texture dépassant la plage de coordonnées sont rognées.

Exemple

Le code suivant donnera à une sphère une texture granitée répétée quatre fois sur toute la surface, ainsi qu'un logo qui ne couvrira qu'un quart de la surface.

```
m = member(2).model("maSphère")
f = member(2).newTexture("granite", #fromCastmember, \
  member("granite"))
g = member(2).newTexture("logo", #fromCastmember, member("logo"))
s = member(2).newShader("s", #standard)
s.textureList[1] = g
s.textureList[2] = f
s.textureRepeatList[2] = false
s.textureRepeatList[1] = true
s.textureTransformList[1].scale(0.5,0.5,1.0)
s.textureTransformList[2].scale(0.5,0.5,1.0)
s.textureModeList[2] = #wrapPlanar
s.blendFunctionList[2] = #add
m.shaderList = s
```

textureTransform

Utilisation

```
member(quelActeur).shader(quelMatériau).textureTransform
member(quelActeur).model(quelModèle).shader.textureTransform
member(quelActeur).model(quelModèle).shaderList[[index]].\
  textureTransform
```

Description

Propriété 3D de matériau `#standard` ; donne accès à une transformation qui modifie les coordonnées de la première couche de texture du matériau. Manipulez ces transformations pour juxtaposer, faire pivoter ou déplacer la texture avant de l'appliquer à la surface du modèle. La texture même n'est pas affectée, la transformation ne modifiant que la façon dont le matériau applique la texture. La propriété `textureTransform` est appliquée à toutes les coordonnées de texture, quelle que soit la valeur de la propriété `textureMode`. Il s'agit de la dernière modification des coordonnées de la texture avant leur envoi au moteur de rendu. La propriété `textureTransform` est une matrice qui opère sur la texture de l'espace `textureImage`. L'espace `TextureImage` est défini pour exister uniquement sur le plan x,y.

Pour juxtaposer l'image deux fois le long de son axe horizontal, utilisez `référenceDeMatériau.textureTransform.scale(0.5, 1.0, 1.0)`. Le redimensionnement sur l'axe des z est ignoré.

Pour décaler l'image de `point(positionX,positionY)`, utilisez `référenceDeMatériau.textureTransform.translate(positionX,positionY,0.0)`. La translation en fonction d'entiers lorsque la propriété `textureRepeat` a pour valeur `TRUE` n'a aucun effet étant donné que la largeur et la hauteur de la texture auront une valeur comprise entre 0.0 et 1.0 dans ce cas.

Pour appliquer une rotation à une couche de texture, utilisez `référenceDeMatériau.textureTransform.rotate(0,0,angle)`. Les rotations sur l'axe des z sont effectuées autour du second point (0,0) qui correspond au coin supérieur gauche de la texture. Les rotations sur les axes des x et des y sont ignorées.

Tout comme pour les transformations de modèle, les modifications `textureTransform` peuvent être multicouches. Pour faire pivoter la texture autour de `point(positionX,positionY)` au lieu de `point(0,0)`, effectuez d'abord une translation vers `point(0 - positionX, 0 - positionY)`, faites pivoter, puis effectuez encore une translation vers `point(positionX, positionY)`. `textureTransform` est similaire à la propriété `wrapTransform` du matériau, avec les exceptions suivantes. Elle est appliquée dans un espace image 2D plutôt que dans un espace d'univers 3D. Seules les rotations autour de l'axe des z et les translations et redimensionnements sur les axes des x et y ont un résultat. La transformation est appliquée quel que soit le paramètre `référenceDeMatériau.textureMode`. `wrapTransform`, en comparaison, n'est efficace que lorsque `textureMode` a pour valeur `#wrapPlanar`, `#wrapCylindrical` ou `#wrapSpherical`.

Exemple

L'instruction suivante indique la `textureTransform` de la première texture du premier matériau utilisé par le modèle gbCyl3.

```
put member("Séquence").model("gbCyl3").shader.textureTransform
-- transform(1.0000, 0.0000, 0.0000,0.0000, 0.0000, 1.0000, \
  0.0000, 0.0000, 0.0000, 0.0000, 1.0000, 0.0000, 0.0000, 0.0000, \
  0.0000, 1.0000)
```

L'instruction suivante divise par deux la hauteur et la largeur de la première texture utilisée par le matériau gbCyl3. Si la propriété `textureRepeat` de gbCyl3 a pour valeur `TRUE`, quatre copies de la texture sont juxtaposées sur le matériau.

```
member("Séquence").shader("gbCyl3").textureTransform.scale = \
  vector(0.5, 0.5, 1)
```

Cette instruction fait pivoter la première texture utilisée par le matériau gbCyl3 de 90° à partir de `vector(0, 0, 0)`.

```
member("Séquence").shader("gbCyl3").textureTransform.rotation = \
  vector(0, 0, 90)
```

textureTransformList

Utilisation

```
référenceDeMatériau.textureTransformList[indexDeCoucheDeTexture]
member(quelActeur).shader(NomDeMatériau).textureTransformList\
  [indexDeCoucheDeTexture]
member(quelActeur).shader[indexDeListeDeMatériaux].texture\
  TransformList[indexDeCoucheDeTexture]
member(quelActeur).model(nomDeModèle).shader.texture\
  TransformList[indexDeCoucheDeTexture]
member(quelActeur).model(nomDeModèle).shaderList\
  [indexDeListeDeMatériaux]. textureTransformList[indexDeCoucheDeTexture]
```

Description

Propriété 3D de matériau standard ; donne accès à une transformation qui modifie les coordonnées de texture d'une couche de texture spécifiée. Manipulez ces transformations pour juxtaposer, faire pivoter ou déplacer une image de texture avant de l'appliquer à la surface d'un modèle. La texture même n'est pas affectée, la transformation ne modifiant que la façon dont le matériau applique la texture.

Pour juxtaposer l'image deux fois le long de son axe horizontal, utilisez `textureTransformList [quelleCoucheDeTexture].scale(0.5, 1.0, 1.0)`. Les redimensionnements sur l'axe des z seront ignorés étant donné que les images sont en 2D. Les échelles 0.0 devront être évitées (même dans z) afin de ne pas annuler l'effet de la texture tout entière.

Pour décaler l'image de point(positionX,positionY), utilisez `textureTransformList [whichTextureLayer].translate(positionX,positionY,0.0)`. La translation en fonction d'entiers lorsque la propriété `textureRepeat` de cette couche de texture a pour valeur `TRUE` n'a aucun effet étant donné que la largeur et la hauteur de la texture auront une valeur comprise entre 0.0 et 1.0 dans ce cas.

Pour appliquer une rotation à une couche de texture, utilisez `textureTransformList [quelleCoucheDeTexture].rotate(0,0,angle)`. Les rotations sur l'axe des z sont effectuées autour du second point (0,0) qui correspond au coin supérieur gauche de la texture. Les rotations sur les axes des x et des y sont ignorées étant donné que les images sont en 2D par nature.

Tout comme pour les transformations de modèle, les modifications `textureTransform` peuvent être multicouches. Pour faire pivoter l'image autour de `point(positionX,positionY)` au lieu de `point(0,0)`, effectuez d'abord une translation vers `point(0 - positionX, 0 - positionY)`, faites pivoter, puis effectuez encore une translation vers `point(positionX, positionY)`.

`textureTransformList` est similaire à la propriété `wrapTransformList` du matériau, avec les exceptions suivantes.

Elle est appliquée dans un espace image 2D plutôt que dans un espace d'univers 3D. Seules les rotations autour de l'axe des z et les translations et redimensionnements sur les axes des x et y ont un résultat.

La transformation est appliquée quel que soit le paramètre `référenceDeMatériau.textureModeList[index]`. `wrapTransform`, en comparaison, n'est efficace que lorsque `textureMode` a pour valeur `#wrapPlanar`, `#wrapCylindrical` ou `#wrapSpherical`.

Exemple

L'instruction suivante indique la transformation de texture de la troisième texture du premier matériau utilisé par le modèle gbCyl3.

```
put member("séquence").model("gbCyl3").shader.textureTransformList[3]
-- transform(1.0000, 0.0000, 0.0000, 0.0000, 0.0000, 1.0000, \
  0.0000, 0.0000, 0.0000, 0.0000, 1.0000, 0.0000, 0.0000, 0.0000, \
  0.0000, 1.0000)
```

L'instruction suivante divise par deux la hauteur et la largeur de la cinquième texture utilisée par le matériau gbCyl3. Si la valeur `textureRepeatList[5]` de gbCyl3 est `TRUE`, quatre copies de la texture sont juxtaposées sur le matériau.

```
member("Séquence").shader("gbCyl3").textureTransformList[5].scale = \
  vector(0.5, 0.5, 1)
```

Cette instruction fait pivoter la quatrième texture utilisée par le matériau gbCyl3 de 90° à partir de `vector(0, 0, 0)`.

```
member("Séquence").shader("gbCyl3").textureTransformList[4].rotation \
  = vector(0, 0, 90)
```

Les instructions suivantes font pivoter la troisième texture utilisée par le matériau gbCyl3 de 90° autour de son centre, en prenant pour base une texture de 128x128.

```
s = member("Séquence").shader("gbCyl3")
s.textureTransformList[3].translate(-64,-64,0)
s.textureTransformList[3].rotate(0,0,90)
s.textureTransformList[3].translate(64,64,0)
```

textureType

Utilisation

```
member(quelActeur).textureType
```

Description

Propriété 3D de texture ; permet d'obtenir ou de définir le type de la texture par défaut. Les valeurs possibles sont :

- `#none` ne spécifie aucun type de texture.
- `#default` utilise la texture du matériau d'origine.
- `#member` utilise l'image de l'acteur spécifié.

La valeur par défaut de cette propriété est `#default`. Vous devez spécifier `#member` pour cette propriété de façon à pouvoir utiliser la propriété `textureMember`.

Exemple

L'instruction suivante donne à la propriété `textureType` de l'acteur Séquence la valeur `#member`.

```
member("Séquence").textureType = #member
```

Cela permet d'utiliser un acteur bitmap comme source de texture par défaut en définissant la propriété `textureMember`. L'acteur bitmap est appelé « herbe ».

```
member("Séquence").textureMember = "herbe"
```

Voir aussi

`textureMember`

thumbNail

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.thumbNail

// Syntaxe JavaScript
réfObjActeur.thumbNail;
```

Description

Propriété d'acteur ; contient l'image utilisée pour afficher l'aperçu d'un acteur dans la fenêtre Distribution. Lecture/écriture pendant la programmation uniquement.

L'image peut être personnalisée pour n'importe quel acteur.

Exemple

L'exemple suivant indique comment utiliser un acteur servant de repère d'emplacement pour afficher une miniature sur la scène. Cet acteur est placé sur la scène, puis son image est définie sur la miniature de l'acteur 10. Cela permet d'afficher une image réduite sans devoir procéder à une mise à l'échelle ou à une manipulation du graphique :

```
-- Syntaxe Lingo
member("Repère d'emplacement").picture = member(10).thumbNail

// Syntaxe JavaScript
member("Repère d'emplacement").picture = member(10).thumbNail;
```

Voir aussi

`Acteur`

tilt

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.tilt

// Syntaxe JavaScript
réfObjImageObjet.tilt;
```

Description

Propriété d'image-objet QuickTime VR ; inclinaison courante, en degrés, de l'animation QuickTime VR.

Cette propriété peut être testée et définie.

time (objet de temporisation)

Utilisation

```
objetDeTemporisation.time
```

Description

Propriété d'objet de temporisation ; heure système, exprimée en millisecondes, de l'envoi du prochain événement de temporisation par l'*objetDeTemporisation* donné.

Il ne s'agit pas du temps restant jusqu'au prochain événement, mais de la position temporelle absolue du prochain événement de temporisation.

Exemple

Le gestionnaire suivant détermine le temps restant jusqu'à l'envoi du prochain événement de temporisation par l'objet de temporisation `Mise à jour`, en calculant la différence entre sa propriété `time` et la valeur courante des millisecondes, et affiche le résultat dans le champ Temps restant :

```
on prepareFrame
  msAvantLaMiseAJour = timeout("Mise à jour").time - the milliseconds
  secondesAvantLaMiseAJour = msAvantLaMiseAJour / 1000
  minAvantLaMAJ = secondesAvantLaMiseAJour / 60
  member("Temps restant").text = string(minAvantLaMAJ ) && "minutes avant \
  la fin du prochain délai"
end
```

Voir aussi

`milliseconds`, `period`, `persistent`, `target`, `timeout()`, `timeoutHandler`

timeoutHandler

Utilisation

objetDeTemporisation.timeoutHandler

Description

Propriété système ; représente le nom du gestionnaire qui recevra les messages de temporisation de l'*objetDeTemporisation* donné. Sa valeur est un symbole, tel que #délaiDépassé. Le timeoutHandler est toujours un gestionnaire compris dans l'objet cible de l'objet de temporisation, ou dans un script d'animation, si aucune cible n'a été définie pour l'objet de temporisation.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante affiche le timeoutHandler de l'objet Minuterie du jeu dans la fenêtre Messages :

```
put timeout("Minuterie du jeu").timeoutHandler
```

Voir aussi

target, timeout(), timeoutList

timeoutList

Utilisation

```
-- Syntaxe Lingo
_movie.timeoutList

// Syntaxe JavaScript
_movie.timeoutList;
```

Description

Propriété d'animation ; liste linéaire contenant tous les objets de temporisation actuellement actifs. En lecture seule.

Utilisez la méthode forget() pour supprimer un objet de temporisation.

Les objets de temporisations sont ajoutés à la liste timeoutList avec la méthode new().

Exemple

L'instruction suivante supprime le troisième objet de temporisation de la liste timeoutList :

```
-- Syntaxe Lingo
_movie.timeoutList[3].forget()

// Syntaxe JavaScript
_movie.timeoutList[3].forget();
```

Voir aussi

forget() (fenêtre), Animation, new()

timeScale

Utilisation

```
member(quelActeur).timeScale
the timeScale of member quelActeur
```

Description

Propriété d'acteur ; renvoie l'unité temporelle par seconde sur laquelle les images de la vidéo numérique sont basées. Par exemple, une vidéo numérique QuickTime est mesurée en 1/600ème de seconde.

Cette propriété peut être testée, mais pas définie.

Voir aussi

digitalVideoTimeScale

title (DVD)

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.title

// Syntaxe JavaScript
réfObjDvd.title;
```

Description

Propriété DVD ; spécifie le titre courant. Lecture/écriture.

Cette propriété renvoie un nombre entier qui spécifie le numéro du titre courant.

Exemple

Cette instruction renvoie le titre courant :

```
-- Syntaxe Lingo
trace (member(1).title) -- 1

// Syntaxe JavaScript
trace (member(1).title); // 1
```

Voir aussi

DVD

title (fenêtre)

Utilisation

```
-- Syntaxe Lingo
réfObjFenêtre.title

// Syntaxe JavaScript
réfObjFenêtre.title;
```

Description

Propriété de fenêtre ; affecte un titre à une fenêtre. Lecture/écriture.

Exemple

Cette instruction affecte le titre Planètes à la cinquième fenêtre :

```
-- Syntaxe Lingo
_player.windowList[5].title = "Planètes"

// Syntaxe JavaScript
_player.windowList[5].title = "Planètes";
```

Voir aussi

Fenêtre

titlebarOptions

Utilisation

```
-- Syntaxe Lingo
réfObjFenêtre.titlebarOptions

// Syntaxe JavaScript
réfObjFenêtre.titlebarOptions;
```

Description

Propriété de fenêtre; indique une liste de propriétés qui répertorie les options de barre de titre d'une fenêtre. Lecture/écriture.

La liste de propriétés contient les propriétés suivantes :

Propriété	Description
#icon	Indique l'icône d'acteur à utiliser dans la barre de titre. Cette propriété n'est disponible que si la barre de titre est visible (la propriété #visible est définie sur TRUE).
#visible	Indique si la barre de titre est visible. Si sa valeur est FALSE, la barre de titre n'est pas visible et toutes les autres propriétés de barres de titre et de fenêtres ne sont pas affectées. Si sa valeur est TRUE, la barre de titre est visible et la fenêtre conserve l'état de toutes les autres propriétés de barres de titre et de fenêtres. La valeur par défaut est TRUE.
#closebox	Indique si une case de fermeture apparaît dans le coin supérieur droit de la fenêtre. Si sa valeur est TRUE, une case de fermeture apparaît. Si sa valeur est FALSE, aucune case de fermeture n'apparaît. La valeur par défaut est TRUE.
#minimizebox	Indique si une case de réduction apparaît dans le coin supérieur droit de la fenêtre. Si sa valeur est TRUE, une case de réduction apparaît. Si sa valeur est FALSE, aucune case de réduction n'apparaît. La valeur par défaut est TRUE.
#maximizebox	Indique si une case d'agrandissement apparaît dans le coin supérieur droit de la fenêtre. Si sa valeur est TRUE, une case d'agrandissement apparaît. Si sa valeur est FALSE, aucune case d'agrandissement n'apparaît. La valeur par défaut est TRUE.
#sideTitlebar	(Macintosh uniquement) Indique si la barre de titre doit apparaître sur le côté de la fenêtre. Si sa valeur est TRUE, la barre de titre apparaît sur le côté de la fenêtre. Si sa valeur est FALSE, la barre de titre n'apparaît pas sur le côté de la fenêtre. La valeur par défaut est FALSE.

Il est également possible d'accéder à ces propriétés à l'aide de la propriété `displayTemplate` de l'objet d'animation.

Exemple

Cette instruction affiche toutes les options de barre de titre de la fenêtre intitulée Eléments, dans la fenêtre Messages :

```
-- Syntaxe Lingo
trace(window("Eléments").titlebarOptions)

// Syntaxe JavaScript
trace(window("Eléments").titlebarOptions);
```

Ces instructions affectent la propriété `icon` à l'acteur bitmap `smallIcon`:

```
-- Syntaxe Lingo
window("Eléments").titlebarOptions.icon = member("smallIcon")

// Syntaxe JavaScript
window("Eléments").titlebarOptions.icon = member("smallIcon");
```

Voir aussi

`appearanceOptions`, `displayTemplate`, `Fenêtre`

titleCount

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.titleCount

// Syntaxe JavaScript
réfObjDvd.titleCount;
```

Description

Propriété DVD ; renvoie le nombre de titres disponibles. En lecture seule.

Le nombre de titres disponibles est compris entre 1 et 99.

Voir aussi

`DVD`

toolXtraList

Utilisation

```
-- Syntaxe Lingo
_player.toolXtraList

// Syntaxe JavaScript
_player.toolXtraList;
```

Description

Propriété de lecteur ; renvoie une liste linéaire de tous les Xtras d'outils disponibles sur le lecteur Director. En lecture seule.

Exemple

L'instruction suivante affiche, dans la fenêtre Messages, tous les Xtras d'outils disponibles.

```
-- Syntaxe Lingo
put(_player.toolXtraList)

// Syntaxe JavaScript
put(_player.toolXtraList);
```

Voir aussi

mediaXtraList, Lecteur, scriptingXtraList, transitionXtraList, xtraList (lecteur)

toon (modificateur)

Utilisation

member(*quelActeur*).model(*quelModèle*).toon.*propriétéDeModificateurToon*

Description

Modificateur 3D ; vous pouvez, une fois le modificateur #toon ajouté à un modèle, en obtenir et définir les propriétés.

Le modificateur #toon dessine un modèle en n'utilisant qu'un nombre réduit de couleurs, ce qui permet d'obtenir un style de dessin animé pour la surface du modèle. Lorsque le modificateur #toon est appliqué, les propriétés texture, reflectionMap, diffuseLightMap, specularLightMap et glossMap du matériau du modèle sont ignorées.

Lorsque le modificateur #toon est utilisé en conjonction avec le modificateur #inker, le rendu est cumulatif et varie en fonction du premier modificateur appliqué. La liste des modificateurs renvoyée par la propriété modifier indiquera #inker ou #toon (en fonction du premier qui aura été ajouté), mais pas les deux. Le modificateur #toon ne peut pas être utilisé en conjonction avec le modificateur #sds.

Le modificateur #toon a les propriétés suivantes :

Remarque : Pour plus d'informations, consultez les entrées des différentes propriétés.

- style permet d'obtenir ou de définir le style appliqué aux transitions des couleurs. Les valeurs possibles sont les suivantes :

  #toon donne des transitions aiguës entre les couleurs disponibles.

  #gradient donne des transitions fluides entre les couleurs disponibles.

  #blackAndWhite donne des transitions aiguës entre le noir et le blanc.
- colorSteps permet d'obtenir ou de définir le nombre de couleurs utilisées pour le calcul de l'éclairage. Cette valeur est arrondie à la puissance de 2 inférieure la plus proche. Les valeurs permises sont 2, 4, 8 et 16 ; la valeur par défaut est 2.
- shadowPercentage permet d'obtenir ou de définir le pourcentage des couleurs (colorSteps) définies pour l'éclairage utilisé pour le rendu de la portion ombragée de la surface du modèle. Les valeurs possibles s'étalent de 0 à 100. La valeur par défaut est 50.
- shadowStrength permet d'obtenir ou de définir le niveau d'obscurité appliqué à la portion ombragée de la surface du modèle. Les valeurs possibles sont n'importe quel nombre à virgule flottante non négatif. La valeur par défaut est 1.0.

- `highlightPercentage` permet d'obtenir ou de définir le pourcentage des couleurs (`colorSteps`) définies pour l'éclairage utilisé pour le rendu de la portion éclairée de la surface du modèle. Les valeurs possibles s'étalent de 0 à 100. La valeur par défaut est 50.
- `highlightStrength` permet d'obtenir ou de définir le niveau de luminosité appliqué à la portion éclairée de la surface du modèle. Les valeurs possibles sont n'importe quel nombre à virgule flottante non négatif. La valeur par défaut est 1.0.
- `lineColor` permet d'obtenir ou de définir la couleur des lignes dessinées par le modificateur `#inker`. Les valeurs possibles sont n'importe quel objet de couleur Lingo. La valeur par défaut est `rgb(0, 0, 0)`, qui est le noir.
- `creases` permet de savoir ou de définir si les lignes sont dessinées avec des plis. Il s'agit d'une valeur booléenne ; la valeur par défaut est `TRUE`.
- `creaseAngle` si `creases` a pour valeur `TRUE`, permet de savoir ou de définir la façon dont la fonction des lignes du modificateur `#toon` répond à la présence des plis.
- `boundary` permet de savoir ou de définir si les lignes sont dessinées autour de la limite de la surface. Il s'agit d'une valeur booléenne ; la valeur par défaut est `TRUE`.
- `lineOffset` permet de savoir ou de définir si les lignes sont dessinées en fonction de la surface et de la caméra. Les valeurs négatives déplacent les lignes vers la caméra. Les valeurs positives éloignent les lignes de la caméra. Les valeurs possibles sont des nombres à virgule flottante compris entre -100.0 et 100.0. La valeur par défaut est -2.0.
- `useLineOffset` permet de savoir ou de définir si `lineOffset` est activé ou désactivé. Il s'agit d'une valeur booléenne ; la valeur par défaut est `FALSE`.
- `silhouettes` permet de savoir ou de définir si les lignes sont dessinées avec les bords le long de la bordure d'un modèle, en soulignant la forme. Il s'agit d'une valeur booléenne ; la valeur par défaut est `TRUE`.

Voir aussi

`addModifier`, `modifiers`, `sds (modificateur)`, `inker (modificateur)`

top

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.top

// Syntaxe JavaScript
réfObjImageObjet.top;
```

Description

Propriété d'image-objet ; renvoie ou définit la coordonnée verticale du bord supérieur du rectangle délimitant une image-objet, en pixels à partir du coin supérieur gauche de la scène. Lecture/écriture.

Exemple

L'instruction suivante vérifie si le haut de l'image-objet 3 dépasse le haut de la scène et, le cas échéant, appelle le gestionnaire `débordementEnHaut` :

```
-- Syntaxe Lingo
if (sprite(3).top < 0) then
  débordementEnHaut()
end if

// Syntaxe JavaScript
if (sprite(3).top < 0) {
  débordementEnHaut();
}
```

Voir aussi

bottom, height, left, locH, locV, right, Image-objet, width

topSpacing

Utilisation

expressionSousChaîne.topSpacing

Description

Propriété d'acteur texte ; permet de spécifier tout espacement supplémentaire applicable en haut de chaque paragraphe dans la partie *expressionSousChaîne* de l'acteur texte.

La valeur est un entier, qui indique un espacement moindre entre les paragraphes s'il est inférieur à 0 et un espacement plus important s'il est supérieur à 0.

La valeur par défaut est 0 ; elle correspond à l'espacement par défaut entre les paragraphes.

Exemple

L'instruction suivante définit la propriété `topSpacing` du second paragraphe de l'acteur texte `monTexte` sur 20 :

```
member(1).paragraph[2].topSpacing = 20
```

Voir aussi

bottomSpacing

traceLoad

Utilisation

```
-- Syntaxe Lingo
_movie.traceLoad

// Syntaxe JavaScript
_movie.traceLoad;
```

Description

Propriété d'animation ; indique la quantité d'informations sur les acteurs affichée pendant leur chargement. Lecture/écriture.

Les valeurs correctes de `traceLoad` sont les suivantes.

- 0—N'affiche aucune information (valeur par défaut).
- 1—Affiche les noms des acteurs.
- 2—Affiche les noms des acteurs, le numéro de l'image courante, le nom de l'animation et le décalage de recherche du fichier (la quantité relative de déplacement que le lecteur doit effectuer pour charger les médias).

Exemple

L'instruction suivante entraîne l'affichage des noms des acteurs lorsqu'ils sont chargés :

```
-- Syntaxe Lingo
_movie.traceLoad = 1

// Syntaxe JavaScript
_movie.traceLoad = 1;
```

Voir aussi

`Animation`

traceLogFile

Utilisation

```
-- Syntaxe Lingo
_movie.traceLogFile

// Syntaxe JavaScript
_movie.traceLogFile;
```

Description

Propriété d'animation ; spécifie le nom du fichier dans lequel le contenu de la fenêtre Messages est enregistré. Lecture/écriture.

Vous pouvez fermer ce fichier en affectant à la propriété `traceLogFile` la valeur `EMPTY` (Lingo) ou une chaîne vide " " (syntaxe JavaScript). Tout élément généré devant apparaître dans la fenêtre Messages est écrit dans ce fichier. Cette propriété est utile pour le processus de débogage lors de l'exécution d'une animation dans une projection et dans le cadre de la programmation.

Exemple

L'instruction suivante demande à Lingo ou la syntaxe JavaScript d'écrire le contenu de la fenêtre Messages dans le fichier Messages.txt dans le dossier de l'animation courante :

```
-- Syntaxe Lingo
_movie.traceLogFile = _movie.path & "Messages.txt"

// Syntaxe JavaScript
_movie.traceLogFile = _movie.path + "Messages.txt";
```

L'instruction suivante ferme le fichier dans lequel est écrit le contenu de la fenêtre Messages :

```
-- Syntaxe Lingo
_movie.traceLogFile = ""

// Syntaxe JavaScript
_movie.traceLogFile = "";
```

Voir aussi

Animation

traceScript

Utilisation

```
-- Syntaxe Lingo
_movie.traceScript

// Syntaxe JavaScript
_movie.traceScript;
```

Description

Propriété d'animation ; indique si la fonction de suivi de l'animation est activée (`TRUE`) ou non (`FALSE`).
Lecture/écriture.

Lorsque la fonction `traceScript` est activée, la fenêtre Messages affiche chaque ligne de script exécutée.

Exemple

L'instruction suivante active la propriété `traceScript`.

```
-- Syntaxe Lingo
_movie.traceScript = TRUE

// Syntaxe JavaScript
_movie.traceScript = true;
```

Voir aussi

Animation

trackCount (acteur)

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.trackCount()

// Syntaxe JavaScript
réfObjActeur.trackCount();
```

Description

Propriété d'acteur vidéo numérique ; renvoie le nombre de pistes dans l'acteur vidéo numérique spécifié.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante détermine le nombre de pistes de l'acteur vidéo numérique Chronique jazz et affiche le résultat dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(member("Chronique Jazz").trackCount())

// Syntaxe JavaScript
trace(member("Chronique Jazz").trackCount());
```

trackCount (image-objet)

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.trackCount()

// Syntaxe JavaScript
réfObjImageObjet.trackCount();
```

Description

Propriété d'image-objet vidéo numérique ; renvoie le nombre de pistes dans l'image-objet vidéo numérique spécifiée.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante détermine le nombre de pistes de l'image-objet vidéo numérique affectée à la piste 10 et affiche le résultat dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(sprite(10).trackCount())

// Syntaxe JavaScript
trace(sprite(10).trackCount());
```

trackEnabled

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.trackEnabled(quellePiste)

// Syntaxe JavaScript
réfObjImageObjet.trackEnabled(quellePiste);
```

Description

Propriété d'image-objet vidéo numérique ; indique l'état de la piste spécifiée d'une vidéo numérique. Cette propriété a la valeur `TRUE` si la piste est activée et en cours d'exécution. Cette propriété a la valeur `FALSE` si la piste est désactivée et n'est plus en cours d'exécution ou n'est pas mise à jour.

Cette propriété ne peut pas être définie. Vous devez utiliser la propriété `setTrackEnabled`.

Exemple

L'instruction suivante vérifie si la piste 2 de l'image-objet vidéo numérique 1 est activée :

```
-- Syntaxe Lingo
put(sprite(1).trackEnabled(2))

// Syntaxe JavaScript
put(sprite(1).trackEnabled(2));
```

Voir aussi

setTrackEnabled()

trackNextKeyTime

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.trackNextKeyTime(quellePiste);\

// Syntaxe JavaScript
réfObjImageObjet.trackNextKeyTime(quellePiste);
```

Description

Propriété d'image-objet vidéo numérique ; indique la position temporelle de l'image-clé suivant la position temporelle actuelle de la piste vidéo numérique spécifiée.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante détermine la position temporelle de l'image-clé suivant la position temporelle de la piste 5 de la vidéo numérique affectée à la piste d'image-objet 15 et l'affiche dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(sprite(15).trackNextKeyTime(5))

// Syntaxe JavaScript
put(sprite(15).trackNextKeyTime(5));
```

trackNextSampleTime

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.trackNextSampleTime(quellePiste)

// Syntaxe JavaScript
réfObjImageObjet.trackNextSampleTime(quellePiste);
```

Description

Propriété d'image-objet vidéo numérique ; indique la position temporelle de l'image-clé qui suit la position temporelle actuelle de la vidéo numérique. Cette propriété est pratique pour localiser les pistes texte dans une vidéo numérique.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante détermine la position temporelle de l'image-clé qui suit la position temporelle dans la piste 5 de la vidéo numérique affectée à l'image-objet 15 :

```
-- Syntaxe Lingo
put(sprite(15).trackNextSampleTime(5))

// Syntaxe JavaScript
put(sprite(15).trackNextSampleTime(5));
```

trackPreviousKeyTime

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.trackPreviousKeyTime(quellePiste)

// Syntaxe JavaScript
réfObjImageObjet.trackPreviousKeyTime(quellePiste);
```

Description

Propriété d'image-objet vidéo numérique ; renvoie la position temporelle de l'image-clé qui précède la position temporelle actuelle.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante détermine la position temporelle de l'image-clé de la piste 5 qui précède la position temporelle actuelle de l'image-objet vidéo numérique affectée à la piste 15 et affiche le résultat dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(sprite(15).trackPreviousKeyTime(5))

// Syntaxe JavaScript
put(sprite(15).trackPreviousKeyTime(5));
```

trackPreviousSampleTime

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.trackPreviousSampleTime(quellePiste)

// Syntaxe JavaScript
réfObjImageObjet.trackPreviousSampleTime(quellePiste);
```

Description

Propriété d'image-objet vidéo numérique ; indique la position temporelle de l'échantillon qui précède la position temporelle actuelle de la vidéo numérique. Cette propriété est pratique pour localiser les pistes texte dans une vidéo numérique.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante détermine la position temporelle de l'échantillon de la piste 5 qui précède la position temporelle actuelle de l'image-objet vidéo numérique affectée à la piste 15 et affiche le résultat dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(sprite(15).trackPreviousSampleTime(5))

// Syntaxe JavaScript
put(sprite(15).trackPreviousSampleTime(5));
```

trackStartTime (acteur)

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.trackStartTime(quellePiste)

// Syntaxe JavaScript
réfObjActeur.trackStartTime(quellePiste);
```

Description

Propriété d'acteur vidéo numérique ; renvoie la position temporelle du début de la piste de l'acteur vidéo numérique spécifié.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante détermine la position temporelle de début de la lecture de la piste 5 de l'acteur vidéo numérique Chronique Jazz et affiche le résultat dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(member("Chronique Jazz").trackStartTime(5))

// Syntaxe JavaScript
put(member("Chronique Jazz").trackStartTime(5));
```

trackStartTime (image-objet)

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.trackStartTime(quellePiste)

// Syntaxe JavaScript
réfObjImageObjet.trackStartTime(quellePiste);
```

Description

Propriété d'image-objet vidéo numérique ; définit la position temporelle de début d'une animation vidéo numérique dans la piste d'image-objet spécifiée. La valeur de `trackStartTime` est mesurée en battements.

Cette propriété peut être testée, mais pas définie.

Exemple

Dans la fenêtre Messages, l'instruction suivante signale le début de la lecture de la piste 5 de la piste d'image-objet 10. La position temporelle de début est à 120 battements (2 secondes).

```
-- Syntaxe Lingo
put(sprite(10).trackStartTime(5))

// Syntaxe JavaScript
put(sprite(10).trackStartTime(5));
```

Voir aussi

duration (acteur), playRate (QuickTime, AVI), currentTime (QuickTime, AVI)

trackStopTime (acteur)

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.trackStopTime(quellePiste)

// Syntaxe JavaScript
réfObjActeur.trackStopTime(quellePiste);
```

Description

Propriété d'acteur vidéo numérique ; renvoie la position temporelle d'arrêt de la piste de l'acteur vidéo numérique spécifié. Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante détermine la position temporelle d'arrêt de la piste 5 de l'acteur vidéo numérique Chronique Jazz et affiche le résultat dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(member("Chronique Jazz").trackStopTime(5))

// Syntaxe JavaScript
put(member("Chronique Jazz").trackStopTime(5));
```

trackStopTime (image-objet)

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.trackStopTime(quellePiste)

// Syntaxe JavaScript
réfObjImageObjet.trackStopTime(quellePiste);
```

Description

Propriété d'image-objet vidéo numérique ; renvoie la position temporelle d'arrêt de la piste de l'image-objet vidéo numérique spécifiée.

Lors de la lecture d'une animation vidéo numérique, la propriété `trackStopTime` correspond à l'endroit auquel la lecture s'arrête ou effectue une boucle si la propriété `loop` est activée.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante détermine la position temporelle d'arrêt de la piste 5 de la vidéo numérique affectée à l'image-objet 6 et affiche le résultat dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(sprite(6).trackStopTime(5))

// Syntaxe JavaScript
put(sprite(6).trackStopTime(5));
```

Voir aussi

playRate (QuickTime, AVI), currentTime (QuickTime, AVI), trackStartTime (acteur)

trackText

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.trackText(quellePiste)

// Syntaxe JavaScript
réfObjImageObjet.trackText(quellePiste);
```

Description

Propriété d'image-objet vidéo numérique ; renvoie le texte situé dans la piste spécifiée de la vidéo numérique à la position temporelle actuelle. Le résultat est une chaîne de caractères, qui peut avoir une longueur de 32 Ko. Cette propriété ne s'applique qu'aux pistes de texte.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante affecte le texte de la piste 5 de la vidéo numérique affectée à la position temporelle actuelle à l'image-objet 20 à l'acteur champ Archives :

```
-- Syntaxe Lingo
member("Archives").text = string(sprite(20).trackText(5))

// Syntaxe JavaScript
member("Archives").text = sprite(20).trackText(5).toString();
```

trackType (acteur)

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.trackType(quellePiste)

// Syntaxe JavaScript
réfObjActeur.trackType(quellePiste);
```

Description

Propriété d'acteur vidéo numérique ; indique le type de média qui se trouve dans la piste spécifiée de l'acteur indiqué. Les valeurs possibles sont #video, #sound, #text et #music.

Cette propriété peut être testée, mais pas définie.

Exemple

Le gestionnaire suivant vérifie si la piste 5 de l'acteur vidéo numérique Nouvelles du jour est une piste texte et, le cas échéant, exécute le gestionnaire `vérifDeTexte` :

```
-- Syntaxe Lingo
on vérifDeTexte
  if member("Nouvelles du jour").trackType(5) = #text then
    textFormat
  end if
end

// Syntaxe JavaScript
function vérifDeTexte() {
  var tt = member("Nouvelles du jour").trackType(5);
  if (tt == "text") {
    textFormat();
  }
}
```

trackType (image-objet)

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.trackType(quellePiste)

// Syntaxe JavaScript
réfObjImageObjet.trackType(quellePiste);
```

Description

Propriété d'image-objet vidéo numérique ; renvoie le type de média qui se trouve dans la piste spécifiée de l'image-objet indiquée. Les valeurs possibles sont `#video`, `#sound`, `#text` et `#music`.

Cette propriété peut être testée, mais pas définie.

Exemple

Le gestionnaire suivant vérifie si la piste 5 de l'image-objet vidéo numérique affectée à la piste 10 est une piste texte et, le cas échéant, exécute le gestionnaire `vérifDeTexte` :

```
-- Syntaxe Lingo
on vérifDeTexte
  if sprite(10).trackType(5) = #text then
    textFormat
  end if
end

// Syntaxe JavaScript
function vérifDeTexte() {
  var tt = sprite(10).trackType(5);
  if (tt == "text") {
    textFormat();
  }
}
```

trails

Utilisation

```
sprite(quelleImageObjet).trails
the trails of sprite quelleImageObjet
```

Description

Propriété d'image-objet ; active (1 ou `TRUE`) ou désactive (0 ou `FALSE`) l'effet des traces pour l'image-objet spécifiée par *`quelleImageObjet`*. Pour que la valeur définie par Lingo dure au-delà de l'image-objet courante, un script doit être affecté à l'image-objet.

Pour supprimer les traces, animez une autre image-objet sur ces pixels ou utilisez une transition.

Exemple

L'instruction suivante active les traces pour l'image-objet 7 :

```
sprite(7).trails = 1
```

Voir aussi

`directToStage`

transform (propriété)

Utilisation

```
member(quelActeur).node(quelNœud).transform
member(quelActeur).node(quelNœud).transform.transformation\
  Propriété
member(quelActeur).model(quelModèle).bonesPlayer.\
  bone[iDdeSegment].transform
member(quelActeur).model(quelModèle).bonesPlayer.\
  bone[iDdeSegment].transform.propriétéDeTransformation
```

Description

Propriété et commande 3D ; permet d'obtenir ou de définir la transformation associée à un nœud ou segment particulier au sein d'un modèle utilisant le modificateur `bonesPlayer`. En tant que commande, `transform` donne accès aux différentes commandes et propriétés de l'objet de transformation. Un nœud peut être un objet de caméra, groupe, lumière ou modèle.

Pour les objets de nœud, cette propriété est, par défaut, la transformation d'identité. Une transformation de nœud définit la position, la rotation, et l'échelle du nœud en fonction de son objet parent. Lorsque le parent d'un nœud est l'objet groupe World, la propriété `transform` du nœud a la même valeur que celle renvoyée par la commande `getWorldTransform()`.

Pour les segments des modèles utilisant le modificateur `bonesPlayer`, cette propriété prend comme valeur par défaut la valeur de la transformation affectée au segment à la création du fichier du modèle. La transformation d'un segment représente la rotation du segment en fonction de son segment parent et sa position en fonction de la position d'origine de l'articulation. La position d'articulation d'origine est déterminée au moment de la création du fichier du modèle.

Vous pouvez utiliser les commandes et propriétés de transformation suivantes avec la propriété `transform` des objets nœuds :

Remarque : Cette section ne contenant qu'un récapitulatif, vous trouverez de plus amples informations en consultant les entrées correspondantes.

- `preScale` applique une mise à l'échelle avant les décalages de position, rotation et d'échelle de la transformation.
- `preTranslate` applique une translation avant les décalages de position, rotation et d'échelle de la transformation.
- `preRotate` applique une rotation avant les décalages de position, rotation et d'échelle de la transformation.
- `scale` (commande) applique une mise à l'échelle après les décalages de position, rotation et d'échelle de la transformation.
- `scale` (transformation) permet d'obtenir ou de définir le degré de redimensionnement de la transformation.
- `translate` applique une translation après les décalages de position, rotation et d'échelle de la transformation.
- `rotate` applique une rotation après les décalages de position, rotation et d'échelle de la transformation.
- `position` (transformation) permet d'obtenir ou de définir le décalage de position de la transformation.
- `rotation` (transformation) permet d'obtenir ou de définir le décalage de rotation de la transformation.

Pour modifier la propriété `transform` d'un segment au sein d'un modèle, vous devrez enregistrer une copie de la transformation d'origine du segment, modifier la copie enregistrée à l'aide des commandes et propriétés indiquées ci-dessus, puis réinitialiser la propriété `transform` du segment de façon à la rendre égale à la transformation modifiée. Par exemple :

```
t = member("personnage").model("bipède").bonesPlayer.bone[38].\
  transform.duplicate()
t.translate(25,0,-3)
member("personnage").model("bipède").bonesPlayer.bone[38].\
  transform = t
```

Paramètres

Aucun.

Exemple

L'instruction suivante indique la transformation du modèle Boîte, suivie des propriétés de position et de rotation de la transformation.

```
put member("Univers 3D").model("Boîte").transform
-- transform(1.000000,0.000000,0.000000,0.000000, \
  0.000000,1.000000,0.000000,0.000000, \
  0.000000,0.000000,1.000000,0.000000, -\
  94.144844,119.012825,0.000000,1.000000)
put member("Univers 3D").model("Boîte").transform.position
-- vector( 119.0128, -94.1448, 0.0000 )
put member("Univers 3D").model("Boîte").transform.rotation
-- vector( 0.0000, 0.0000, 0.0000 )
```

Voir aussi

interpolateTo(), scale (transformation), rotation (transformation), position (transformation), bone, worldTransform, preRotate, preScale(), preTranslate()

transitionType

Utilisation

```
member(quelActeur).transitionType
the transitionType of member quelActeur
```

Description

Propriété d'acteur transition ; détermine un type de transition, donné sous la forme d'un nombre. Les valeurs possibles sont les mêmes que les numéros de code affectés aux transitions avec la commande `puppetTransition`.

Exemple

L'instruction suivante affecte le type d'acteur de transition 51 à l'acteur 3, qui est un acteur de type fondu pixels :

```
member(3).transitionType = 51
```

transitionXtraList

Utilisation

```
-- Syntaxe Lingo
_player.transitionXtraList

// Syntaxe JavaScript
_player.transitionXtraList;
```

Description

Propriété de lecteur ; renvoie une liste linéaire de tous les Xtras de transition disponibles sur le lecteur Director. En lecture seule.

Exemple

L'instruction suivante affiche, dans la fenêtre Messages, tous les Xtras de transition disponibles.

```
-- Syntaxe Lingo
put(_player.transitionXtraList)

// Syntaxe JavaScript
put(_player.transitionXtraList);
```

Voir aussi

mediaXtraList, Lecteur, scriptingXtraList, toolXtraList, xtraList (lecteur)

translation

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.translation

// Syntaxe JavaScript
réfObjActeurOuImageObjet.translation;
```

Description

Propriété d'acteur et d'image-objet QuickTime ; contrôle le décalage d'une image d'image-objet QuickTime à l'intérieur du cadre de délimitation de l'image-objet.

Ce décalage est exprimé par rapport à l'emplacement par défaut de l'image-objet tel qu'il est défini par sa propriété `center`. Lorsque celle-ci a pour valeur `TRUE`, l'image-objet est décalée par rapport au centre du rectangle de délimitation ; lorsque la propriété `center` a pour valeur `FALSE`, l'image-objet est décalée par rapport au coin supérieur gauche du rectangle de délimitation.

Le décalage, indiqué en pixels sous forme de nombres entiers positifs ou négatifs, est défini comme une liste Director : [*transX, transY*]. Le paramètre *transX* indique le décalage horizontal à partir de l'emplacement par défaut de l'image-objet, tandis que le paramètre *transY* indique le décalage vertical. La valeur par défaut est `[0,0]`.

Lorsque la propriété `crop` de l'image-objet est réglée sur `TRUE`, la propriété `translation` peut être utilisée pour masquer des parties de l'animation QuickTime en les déplaçant à l'extérieur du cadre de délimitation. Lorsque la propriété `crop` est réglée sur `FALSE`, la propriété `translation` n'est pas prise en compte et l'image-objet est toujours placée dans le coin supérieur gauche du cadre de délimitation.

Cette propriété peut être testée et définie.

Exemple

Le script d'image suivant suppose que la propriété `center` d'une image-objet QuickTime d'une largeur de 320 pixels placée dans la piste 5 est réglée sur `FALSE` et que sa propriété `crop` est réglée sur `TRUE`. Elle garde la tête de lecture dans l'image courante jusqu'à ce que le point de translation horizontal de l'animation se soit déplacé vers le bord droit de l'image-objet, par incréments de 10 pixels. Cela crée un effet de balayage vers la droite, qui met l'image-objet hors de vue en la déplaçant vers la droite. Lorsque l'image-objet ne figure plus à l'écran, la tête de lecture passe à l'image suivante.

```
-- Syntaxe Lingo
on exitFrame
  positionHorizontale = sprite(5).translation[1]
  if positionHorizontale < 320 then
    sprite(5).translation = sprite(5).translation + [10, 0]
    _movie.go(_movie.frame)
  end if
end

// Syntaxe JavaScript
function exitFrame() {
  var positionHorizontale = sprite(5).translation[1];
  if (positionHorizontale < 320 ) {
    sprite(5).translation = sprite(5).translation + list(10, 0);
    _movie.go(_movie.frame);
  }
}
```

transparent

Utilisation

```
member(quelActeur).shader(quelMatériau).transparent
member(quelActeur).model(quelModèle).shader.transparent
member(quelActeur).model(quelModèle).shaderList\
  [indexDeListeDeMatériaux].transparent
```

Description

Propriété 3D de matériau standard ; permet de savoir ou de définir si l'opacité du modèle est réalisée à l'aide de valeurs alpha (`TRUE`) ou s'il s'agit d'un rendu opaque (`FALSE`). La valeur par défaut de cette propriété est `TRUE` (avec alpha).

La fonctionnalité `shader.blend` dépend de cette propriété.

Tous les matériaux ont accès aux propriétés `#standard` ; en plus de ces propriétés standard, les matériaux de type `#engraver`, `#newsprint` et `#painter` possèdent des propriétés uniques à leur type. Pour plus d'informations, consultez `newShader`.

Exemple

L'instruction suivante entraîne un rendu opaque du modèle Pluton. Le paramètre de la propriété `blend` du matériau de ce modèle n'a aucun effet.

```
member("Séquence").model("Pluton").shader.transparent = FALSE
```

Voir aussi

`blendFactor`, `blend (3D)`

triggerCallback

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.triggerCallback

// Syntaxe JavaScript
réfObjImageObjet.triggerCallback;
```

Description

Propriété d'image-objet QuickTime VR ; contient le nom du gestionnaire exécuté lorsque l'utilisateur clique sur une zone référencée dans une animation QuickTime VR. Le gestionnaire reçoit deux arguments : le paramètre `me` et l'identifiant de la zone référencée sur laquelle l'utilisateur a cliqué.

La valeur que le gestionnaire renvoie détermine la façon dont la zone référencée est gérée par l'animation. Si le gestionnaire renvoie la valeur `#continue`, l'image-objet QuickTime VR continue à traiter la zone référencée normalement. S'il renvoie la valeur `#cancel`, le comportement par défaut de la zone référencée est annulé.

Cette propriété doit être réglée sur 0 pour effacer l'instruction d'appel.

L'image-objet QuickTime VR reçoit le message en premier.

Pour des performances optimales, ne définissez de propriété `triggerCallback` que lorsque absolument nécessaire.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante affecte le gestionnaire `MonAppelDeZoneRéférencée` au gestionnaire d'appel d'une image-objet QuickTime VR dès que la tête de lecture entre dans l'étendue de l'image-objet. Le gestionnaire `MonAppelDeZoneRéférencée` est exécuté à chaque fois que la zone référencée est déclenchée. L'appel est annulé lorsque la tête de lecture quitte l'étendue de l'image-objet.

```
-- Syntaxe Lingo
property pMonNuméroDimageObjet, spriteNum

on beginSprite (me)
  pMonNuméroDimageObjet = numéroDimageObjet
  sprite(pMonNuméroDimageObjet).triggerCallback = #MonAppelDeZoneRéférencée
end

on MonAppelDeZoneRéférencée(me, IddeZoneRéférencée)
  put "La zone référencée" && IDdeZoneRéférencée && "vient d'être déclenchée"
end

on endSprite me
  sprite(pMonNuméroDimageObjet).triggerCallback = 0
end
```

```
// Syntaxe JavaScript
function beginSprite() {
  pMonNuméroDimageObjet = this.spriteNum;
  sprite(this.pMonNuméroDimageObjet).triggerCallback =
  symbol("MonAppelDeZoneRéférencée");
}

function MonAppelDeZoneRéférencée(IddeZoneRéférencée) {
  trace("La zone référencée" + IDdeZoneRéférencée + "vient d'être
  déclenchée");
}

function endSprite() {
  sprite(pMonNuméroDimageObjet).triggerCallback = 0;
}
```

trimWhiteSpace

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.trimWhiteSpace

// Syntaxe JavaScript
réfObjActeur.trimWhiteSpace;
```

Description

Propriété d'acteur ; détermine si les pixels blancs situés autour du bord d'un acteur bitmap sont supprimés ou laissés en place. Cette propriété est définie à l'importation de l'acteur. Elle peut être modifiée dans Lingo ou la syntax JavaScript ou dans le volet Bitmap de l'inspecteur des propriétés.

tunnelDepth

Utilisation

```
member(quelActeurTexte).tunnelDepth
member(quelActeur).modelResource(quelModèleDextrudeur\
  Ressource).tunnelDepth
```

Description

Propriété 3D de ressource de modèle d'extrudeur et d'acteur texte. Cette propriété permet d'obtenir ou de définir la profondeur d'extrusion (la distance séparant les faces avant et arrière) d'une ressource de modèle 3D. Les valeurs possibles sont des nombres à virgule flottante compris entre 1.0 et 100.0. La valeur par défaut est 50,0.

Pour plus d'informations sur l'utilisation des ressources de modèle d'extrudeur et des acteurs texte, consultez l'entrée `extrudeToMember`.

Exemple

Dans l'exemple suivant, l'acteur Logo est un acteur texte. L'instruction suivante définit la profondeur du tunnel du logo à 5 ; la profondeur de ses lettres sera très réduite lorsque affichées en mode 3D.

```
member("Logo").tunnelDepth = 5
```

Dans l'exemple suivant, la ressource du modèle Slogan est du texte extrudé. L'instruction suivante donne à la propriété `tunnelDepth` de la ressource de modèle Slogan la valeur 1000 ; ses lettres seront très profondes.

```
member("Séquence").model("Slogan").resource.tunnelDepth = 1000
```

Voir aussi

`extrude3D`

tweened

Utilisation

```
sprite(quelleImageObjet).tweened
the tweened of sprite quelleImageObjet
```

Description

Propriété d'image-objet ; détermine si seule la première image de la nouvelle image-objet est une image-clé (`TRUE`) ou si toutes les images de la nouvelle image-objet sont des images-clés (`FALSE`).

Cette propriété n'affecte pas la lecture et n'est utile que lors de l'enregistrement du scénario.

Cette propriété peut être testée et définie.

Exemple

Lorsque l'instruction suivante est émise, les nouvelles images-objets créées dans la piste 25 n'ont une image-clé que dans la première image de l'étendue de l'image-objet :

```
sprite(25).tweened = 1
```

tweenMode

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).tweenMode
référenceDobjetDeRessourceDeModèle.tweenMode
```

Description

Propriété 3D de particule ; permet de savoir ou de définir si la couleur d'une particule varie en fonction de sa vitesse ou de son âge. La propriété `tweenMode` peut avoir les valeurs suivantes :

- `#velocity` modifie la couleur de la particule entre `colorRange.start` et `colorRange.end` en fonction de sa vitesse.
- `#age` modifie la couleur de la particule en effectuant une interpolation linéaire de la couleur entre `colorRange.start` et `colorRange.end` sur la durée de vie de la particule. Il s'agit de la valeur par défaut de cette propriété.

Exemple

Dans l'exemple suivant, systèmeThermique est une ressource de modèle de type `#particle`. Cette instruction donne à la propriété `tweenMode` de systèmeThermique la valeur `#velocity`, de façon à ce que les particules plus lentes n'atteignent pas la couleur spécifiée par `colorRange.end`, alors que les particules plus rapides l'atteindront.

```
member(8,2).modelResource("systèmeThermique").tweenMode = \
  #velocitytype (lumière)
```

type (lumière)

Utilisation

```
member(quelActeur).light(quelleLumière).type
```

Description

Propriété 3D de lumière ; type de la lumière référencée. Les valeurs possibles de cette propriété sont :

- `#ambient` entraîne une lumière uniforme sur toutes les surfaces. L'intensité des lumières ambiantes n'est pas affectée par la distance les séparant de la source lumineuse.
- `#directional` entraîne une lumière qui semble dirigée dans une direction particulière, sans pour autant être aussi précis que les lumières de type `#spot`. L'intensité des lumières directionnelles diminue avec la distance les séparant de la source lumineuse.
- `#point` entraîne une lumière éclairant dans toutes les directions à partir d'un emplacement spécifique de l'univers 3D. L'effet est semblable à celui produit par une ampoule. L'intensité des lumières `#point` diminue avec la distance les séparant de la source lumineuse.
- `#spot` entraîne une lumière partant d'un point spécifique, dans le cône défini par la direction « en avant » de la lumière et la propriété `spotAngle`. L'intensité de ces lumières décline avec la distance de la source en fonction des valeurs définies dans la propriété `attenuation` de la lumière.

Exemple

L'instruction suivante affiche la propriété de type de la lumière lumièrePrincipale.

```
put member("3D").motion("lumièrePrincipale").type
-- #spot
```

Voir aussi

`spotAngle`, `attenuation`

type (acteur)

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.type

// Syntaxe JavaScript
réfObjActeur.type;
```

Description

Propriété d'acteur ; indique le type d'un acteur. En lecture seule.

La propriété `type` peut avoir l'une des valeurs suivantes :

`#animgif`	`#ole`
`#bitmap`	`#palette`
`#button`	`#picture`
`#cursor`	`#QuickTimeMedia`
`#digitalVideo`	`#realMedia`

#DVD	#script
#empty	#shape
#field	#shockwave3D
#filmLoop	#sound
#flash	#swa
#flashcomponent	#text
#font	#transition
#havok	#vectorShape
#movie	#windowsMedia

Cette liste comprend les types d'acteurs disponibles dans Director et les Xtras l'accompagnant. Vous pouvez également définir des types d'acteurs spéciaux correspondants pour des acteurs personnalisés.

Pour les animations créées sous Director 5 et 6, la propriété `type` renvoie `#field` pour les acteurs champ et `#richText` pour les acteurs texte. Cependant, les acteurs champ créés sous Director 4 renvoient `#text` comme type d'acteur, ce qui offre une compatibilité en amont avec les animations créées sous Director 4.

Exemple

Le gestionnaire suivant vérifie si l'acteur Nouvelles du jour est un acteur champ et, dans la négative, affiche un message d'alerte :

```
-- Syntaxe Lingo
on vérifDuFormat
  if member("Nouvelles du jour").type <> #field then
    _player.alert("Désolé, cet acteur doit être un champ.")
  end if
end

// Syntaxe JavaScript
function vérifDuFormat() {
  if (member("Nouvelles du jour").toString() != "#field") {
    _player.alert("Désolé, cet acteur doit être un champ.");
  }
}
```

Voir aussi

Acteur

type (ressource de modèle)

Utilisation

`member(`*`quelActeur`*`).modelResource(`*`quelleRessourceDeModèle`*`).type`

Description

Propriété 3D de la ressource de modèle ; type de la ressource de modèle référencée. Les valeurs possibles de cette propriété sont :

- `#box` indique que cette ressource de modèle est une ressource de boîte primitive créée avec la commande `newModelResource`.

- `#cylinder` indique que cette ressource de modèle est une ressource de cylindre primitive créée avec la commande `newModelResource`.
- `#extruder` indique que cette ressource de modèle est une ressource d'extrudeur de texte primitive créée avec la commande `extrude3d`.
- `#mesh` indique que cette ressource de modèle est une ressource de générateur de maille primitive créée avec la commande `newMesh`.
- `#particle` indique que cette ressource de modèle est une ressource de système de particules primitive créée avec la commande `newModelResource`.
- `#plane` indique que cette ressource de modèle est une ressource de plan primitive créée avec la commande `newModelResource`.
- `#sphere` indique que cette ressource de modèle est une ressource de sphère primitive créée avec la commande `newModelResource`.
- `#fromFile` indique que cette ressource de modèle a été créée dans un programme autre que Director et chargée à partir d'un fichier ou acteur externe.

Exemple

L'instruction suivante affiche la propriété `type` de la ressource de modèle Hélice.

```
put member("modèles d'hélices").modelResource("Hélice").type
-- #fromFile
```

Voir aussi

`newModelResource`, `newMesh`, `extrude3D`

type (mouvement)

Utilisation

`member(`*`quelActeur`*`).motion(`*`quelMouvement`*`).type`

Description

Propriété 3D de mouvement ; type du mouvement référencé. Les valeurs possibles de cette propriété sont :

- `#bonesPlayer` indique que ce mouvement est une animation basée sur segments qui a besoin du modificateur `#bonesPlayer`.
- `#keyFramePlayer` indique que ce mouvement est une animation basée sur images-clés qui a besoin du modificateur `#keyFramePlayer`.
- `#none` indique que ce mouvement n'a aucun mouvement correspondant et peut être lu avec le modificateur `#bonesPlayer` ou `#keyFramePlayer`. Les mouvements par défaut des acteurs 3D sont de ce type.

Exemple

L'instruction suivante affiche la propriété `type` du mouvement Course.

```
put member("Séquence").motion("Course").type
-- #bonesPlayer
```

L'instruction suivante affiche la propriété `type` du mouvement mouvementParDéfaut.

```
put member("Séquence").motion("mouvementParDéfaut").type
-- #none
```

Voir aussi

`bonesPlayer (modificateur)`, `keyframePlayer (modificateur)`

type (matériau)

Utilisation

```
member(quelActeur).shader(quelMatériau).type
```

Description

Propriété 3D de matériau ; type de matériau du matériau référencé. Les valeurs possibles de cette propriété sont :

- `#standard` indique qu'il s'agit d'un matériau standard.
- `#painter` indique qu'il s'agit d'un matériau peintre.
- `#newsprint` indique qu'il s'agit d'un matériau journal.
- `#engraver` indique qu'il s'agit d'un matériau gravure.

Exemple

L'instruction suivante indique que le matériau utilisé par le modèle boîte2 est un matériau peintre.

```
put member("séquence").model("boîte2").shader.type
-- #painter
```

Voir aussi

`newShader`

type (image-objet)

Utilisation

```
sprite(quelleImageObjet).type
the type of sprite quelleImageObjet
```

Description

Propriété d'image-objet ; libère des pistes d'images-objets pendant l'enregistrement du scénario en donnant à la propriété d'image-objet `type` une valeur de 0 pour ces pistes.

Remarque : L'acteur d'une image-objet doit être remplacé uniquement par un autre acteur du même type, afin d'éviter toute modification des propriétés de l'image-objet lors de l'échange.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante libère la piste d'image-objet 1 lors d'une session d'enregistrement de scénario :

```
sprite(1).type = 0
```

type (texture)

Utilisation

```
member(quelActeur).shader(quelMatériau).type
```

Description

Propriété 3D de texture ; type de texture de la texture référencée. Les valeurs possibles de cette propriété sont :

- `#fromCastMember` indique qu'il s'agit d'une texture créée à partir d'un acteur Director supportant la propriété `image` à l'aide de la commande `newTexture`.
- `#fromImageObject` indique qu'il s'agit d'une texture créée à partir d'un objet image à l'aide de la commande `newTexture`.
- `#importedFromFile` indique qu'il s'agit d'une texture créée dans un programme autre que Director et créée au moment de l'importation d'un fichier ou du chargement d'un acteur.

Exemple

L'instruction suivante indique que la texture utilisée par le matériau du modèle Pluton a été créée à partir d'un objet image.

```
put member("Séquence").model("Pluton").shader.texture.type
-- #fromImageObject
```

Voir aussi

`newTexture`

type (fenêtre)

Utilisation

```
-- Syntaxe Lingo
réfObjFenêtre.type

// Syntaxe JavaScript
réfObjFenêtre.type;
```

Description

Propriété de fenêtre ; spécifie le type de fenêtre. Lecture/écriture.

Si la propriété `type` est définie, toutes les propriétés appartenant à la nouvelle fenêtre sont définies en conséquence.

Cette propriété peut avoir l'une des valeurs suivantes :

Propriété	Description
`#document`	Indique que la fenêtre apparaîtra avec une barre de titre standard, une case de fermeture, une case de réduction et une case d'agrandissement. Ces types de fenêtres peuvent être déplacées.

Propriété	Description
#tool	Indique que la fenêtre apparaîtra avec une barre de titre réduite et seulement une petite case de fermeture dans le coin supérieur droit. Ces types de fenêtres ne reçoivent plus d'événements d'activation ou de désactivation, car les fenêtres #tool sont toujours actives. Ces types de fenêtres seront toujours liées et apparaîtront au-dessus des fenêtres #document.
#dialog	Indique que la fenêtre apparaîtra avec une barre de titre standard, une case de fermeture et sans icône. Ces types de fenêtres sont modales et apparaissent toujours au-dessus des autres fenêtres.

Il est également possible d'accéder à ces propriétés à l'aide de la propriété displayTemplate de l'objet d'animation.

Les comportements des fenêtres dépendent également des valeurs de la propriété type ainsi que de la propriété dockingEnabled de l'objet Animation.

- Si dockingEnabled a pour valeur TRUE et que type a pour valeur #document, l'aspect de la fenêtre MIAW, tout comme son comportement, sera similaire à celui d'une fenêtre de type Document de Director. La fenêtre apparaîtra dans la zone de « l'espace principal » et sera ancrable avec la fenêtre Scène, Scénario et Distribution, les éditeurs de médias et les fenêtres Messages. Elle ne pourra cependant être regroupée avec aucune de ces fenêtres.
- Si dockingEnabled a pour valeur TRUE et que type a pour valeur #tool, l'aspect de la fenêtre MIAW, tout comme son comportement, sera similaire à celui d'une fenêtre de type Outil de Director. La fenêtre pourra se regrouper avec toutes les fenêtres de type Outil, à l'exception de l'inspecteur des propriétés et de la palette des outils.
- Si dockingEnabled a pour valeur TRUE et que type a pour valeur #fullscreen ou #dialog, le type sera ignoré et la fenêtre sera une fenêtre de programmation.

Exemple

Cette instruction affecte le type de fenêtre intitulée Planètes à #tool.

```
-- Syntaxe Lingo
window("Planètes").type = #tool

// Syntaxe JavaScript
window("Planètes").type = symbol("tool");
```

Voir aussi

appearanceOptions, displayTemplate, dockingEnabled, titlebarOptions, Fenêtre

updateLock

Utilisation

```
-- Syntaxe Lingo
_movie.updateLock

// Syntaxe JavaScript
_movie.updateLock;
```

Description

Propriété d'animation ; détermine si la scène est mise à jour pendant l'enregistrement du scénario (FALSE) ou non (TRUE). Lecture/écriture.

Vous pouvez éviter la modification de l'affichage de la scène pendant une session d'enregistrement du scénario en donnant la valeur `TRUE` à `updateLock` avant que le script ne mette à jour le scénario. Si `updateLock` a la valeur `FALSE`, la scène est mise à jour et affiche une nouvelle image à chaque fois que la commande atteint une nouvelle image.

Vous pouvez également utiliser `updateLock` pour empêcher des mises à jour indésirées du scénario à la sortie d'une image, comme lorsque vous sortez temporairement d'une image pour examiner les propriétés d'une autre image.

Bien que cette propriété puisse être utilisée pour masquer les changements apportés à une image pendant l'exécution, il faut savoir que les acteurs champ porteront la marque de leurs changements dès la modification de leur contenu, contrairement aux modifications apportées aux emplacements ou aux acteurs avec d'autres images-objets, qui ne sont mis à jour que lorsque cette propriété est désactivée.

Voir aussi

`Animation`

updateMovieEnabled

Utilisation

```
the updateMovieEnabled
```

Description

Propriété système ; indique si les changements apportés à l'animation courante sont enregistrés automatiquement (`TRUE`) ou non (`FALSE`, par défaut) lorsque celle-ci passe à une autre animation.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante demande à Director d'enregistrer les changements de l'animation courante à chaque fois qu'elle passe à une autre animation :

```
the updateMovieEnabled = TRUE
```

URL

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.URL

// Syntaxe JavaScript
réfObjActeur.URL;
```

Description

Propriété d'acteur ; spécifie l'URL des acteurs Shockwave Audio (SWA) et acteurs animation Flash.

Pour les acteurs animation Flash, cette propriété est identique à la propriété d'acteur `pathName`.

Cette propriété `URL` peut être testée et définie. Pour les acteurs SWA, cette propriété ne peut être définie que lorsque l'acteur SWA lu en flux continu est arrêté.

Exemple

L'instruction suivante fait d'un fichier sur un serveur Internet l'adresse URL de l'acteur SWA Benny G. :

```
-- Syntaxe Lingo
on mouseDown
  member("Benny G.").URL = \
  "http://audio.macromedia.com/samples/classic.swa"
end

// Syntaxe JavaScript
function mouseDown() {
  member("Benny G.").URL =
  "http://audio.macromedia.com/samples/classic.swa"
}
```

useAlpha

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.useAlpha
réfObjImage.useAlpha

// Syntaxe JavaScript
réfObjActeur.useAlpha;
réfObjImage.useAlpha;
```

Description

Propriété d'acteur bitmap et d'objet image ; pour les objets image et acteurs 32 bits comportant des données de couches alpha, détermine si Director utilise ces données à la composition de l'image sur la scène (`TRUE`) ou si Director n'en tient pas compte (`FALSE`).

Exemple

L'exemple suivant bascule l'état d'activation de la couche alpha de l'acteur Premier plan.

```
-- Syntaxe Lingo
member("Premier plan").useAlpha = not(member("Premier plan").useAlpha)

// Syntaxe JavaScript
switch(member("Premier plan").useAlpha) {
  cas 0 :
    member("Premier plan").useAlpha = 1;
    break;
  cas 1 :
    member("Premier plan").useAlpha = 0;
    break;
}
```

useDiffuseWithTexture

Utilisation

```
member(quelActeur).shader(quelMatériau).useDiffuseWithTexture
```

Description

Propriété 3D de matériau standard ; permet de savoir ou de définir si la couleur diffuse est utilisée pour moduler la texture (TRUE) ou non (FALSE).

Lorsque TRUE, cette propriété fonctionne en conjonction avec les propriétés blendFunction et blendConstant : lorsque blendFunction a pour valeur #blend, la couleur diffuse est équilibrée avec la couleur de la texture afin de déterminer la couleur finale. Par exemple, si blendFunction a pour valeur #blend et blendConstant a pour valeur 100.0, la couleur finale est la couleur pure de la texture. Si vous donnez à blendConstant la valeur 0.0, la couleur finale est la couleur diffuse. Si vous donnez à blendConstant la valeur 10.0, la couleur finale est 10 % de la couleur de la texture et 90 % de la couleur diffuse.

La valeur par défaut de cette propriété est FALSE.

Tous les matériaux ont accès aux propriétés #standard ; en plus de ces propriétés standard, les matériaux de type #engraver, #newsprint et #painter possèdent des propriétés uniques à leur type. Pour plus d'informations, consultez newShader.

Exemple

Dans l'exemple suivant, la propriété shaderList du modèle Mystère contient six matériaux. Chaque matériau possède une liste de textures pouvant contenir jusqu'à huit textures. La propriété diffuseColor de l'acteur (Niveau2) est rgb(255, 0, 0). La propriété blendFunction des six matériaux est #blend et leur propriété blendConstant est 80. L'instruction suivante donne à la propriété useDiffuseWithTexture de tous les matériaux utilisés par Mystère la valeur TRUE. Un peu de rouge sera mélangé à la surface du modèle Cette propriété est affectée par les paramètres des propriétés blendFunction, blendFunctionList, blendSource, blendSourceList, blendConstant et blendConstantList.

```
member("Niveau2").model("Mystère").shaderlist.useDiffuseWith\
  Texture = TRUE
```

Voir aussi

blendFunction, blendConstant

useFastQuads

Utilisation

```
-- Syntaxe Lingo
_movie.useFastQuads

// Syntaxe JavaScript
_movie.useFastQuads;
```

Description

Propriété d'animation ; indique s'il faut utiliser des opérations de calcul plus rapide de quadrilatères (TRUE) ou de calcul plus lent (FALSE, valeur par défaut). Lecture/écriture.

Lorsque définie sur TRUE, Director utilise une méthode de calcul plus rapide et moins précise pour les opérations de quadrilatères. Les calculs rapides de quadrilatères sont suffisants pour les effets simples de rotation et d'inclinaison des images-objets.

Lorsque définie sur FALSE, Director utilise la méthode de calcul de quadrilatères par défaut de Director, plus lente, et qui fournit des résultats visuellement plus attractifs, lorsque vous utilisez les quadrilatères pour des effets de distorsion et d'autres effets aléatoires.

Indépendamment de ce paramètre, les opérations de simple rotation et d'inclinaison d'images-objets utilisent toujours la méthode de calcul rapide de quadrilatères. Si vous définissez useFastQuads sur TRUE, la vitesse de ces simples opérations ne sera pas accélérée.

Exemple

L'instruction suivante entraîne Director à utiliser son code de calcul rapide de quadrilatères pour toutes les opérations de quadrilatères de l'animation :

```
-- Syntaxe Lingo
_movie.useFastQuads = TRUE

// Syntaxe JavaScript
_movie.useFastQuads = true;
```

Voir aussi

Animation

useHypertextStyles

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.useHypertextStyles

// Syntaxe JavaScript
réfObjActeur.useHypertextStyles;
```

Description

Propriété d'acteur texte ; contrôle l'affichage des liens hypertexte dans l'acteur texte spécifié.

Si useHypertextStyles est TRUE, tous les liens apparaissent automatiquement en bleu et sont soulignés, et le curseur prend la forme d'un doigt lorsqu'il est placé sur ces liens.

Si cette propriété a la valeur FALSE, le formatage automatique et le changement de forme du curseur sont désactivés.

Exemple

Le comportement suivant bascule l'état d'activation du formatage hypertexte dans l'acteur texte monTexte :

```
-- Syntaxe Lingo
on mouseUp
  member("monTexte").usehypertextStyles = \
  not(member("monTexte").usehypertextStyles)
end

// Syntaxe JavaScript
function mouseUp() {
  member("monTexte").usehypertextStyles =
  !(member("monTexte").usehypertextStyles)
}
```

useLineOffset

Utilisation

```
member(quelActeur).model(quelModèle).toon.useLineOffset
member(quelActeur).model(quelModèle).inker.useLineOffset
```

Description

Propriété 3D de modificateur `toon` et `inker` ; indique si la propriété `lineOffset` du modificateur est utilisée par le modificateur lorsqu'il dessine des lignes sur la surface du modèle.

La valeur par défaut de cette propriété est `FALSE`.

Exemple

L'instruction suivante donne à la propriété `useLineOffset` du modificateur `toon` du modèle Théière la valeur `FALSE`. La propriété `lineOffset` du modificateur `toon` n'aura aucun effet.

```
member("tp").model("Théière").toon.useLineOffset = FALSE
```

Voir aussi

`lineOffset`

userData

Utilisation

```
member(quelActeur).model(quelModèle).userData
member(quelActeur).light(quelleLumière).userData
member(quelActeur).camera(quelleCaméra).userData
member(quelActeur).group(quelleCaméra).userData
```

Description

Propriété 3D ; renvoie la liste de propriétés `userData` d'un modèle, d'un groupe, d'une caméra ou d'une lumière. La valeur par défaut de cette propriété pour un objet créé dans un programme autre que Director est une liste de toutes les propriétés affectées à la propriété `userData` du modèle dans le programme de modélisation 3D. La valeur par défaut de cette propriété pour un objet créé dans Director est une liste de propriétés vide [:], à moins que l'objet n'ait été créé à l'aide de commandes de clonage. Lorsqu'une commande de clonage a été utilisée pour créer l'objet, la propriété `userData` du nouvel objet prend une valeur égale à celle de l'objet source d'origine.

Pour modifier les éléments de cette liste, vous devrez utiliser les commandes `addProp` et `deleteProp` documentées dans le dictionnaire Lingo principal.

Exemple

L'instruction suivante affiche la propriété `userData` du modèle nouvelleCarrosserie.

```
put member("Voiture").model("nouvelleCarrosserie").userData
-- [#chauffeur: "Robert", #dommages: 34]
```

L'instruction suivante ajoute la propriété `#santé` avec une valeur de `100` à la liste de propriétés `userData` du modèle Lecteur.

```
member("Séquence").model("Lecteur").userData.addProp(#santé,100)
```

userName

Utilisation

```
-- Syntaxe Lingo
_player.userName

// Syntaxe JavaScript
_player.userName;
```

Description

Propriété de lecteur ; renvoie une chaîne comprenant le nom d'utilisateur saisi pendant l'installation de Director. En lecture seule.

Cette propriété est uniquement disponible dans l'environnement de programmation. Elle peut être utilisée dans une animation dans une fenêtre personnalisée pour afficher des informations utilisateur.

Exemple

Le gestionnaire suivant place le nom de l'utilisateur et le numéro de série dans une zone d'affichage dès l'ouverture de la fenêtre. Un script d'animation dans une fenêtre est l'endroit idéal pour ce gestionnaire.

```
-- Syntaxe Lingo
on prepareMovie
  chaîneAffichée = _player.userName & RETURN & _player.organizationName \
  & RETURN & _player.serialNumber
  member("Infos utilisateur").text = chaîneAffichée
end

// Syntaxe JavaScript
function prepareMovie() {
  var chaîneAffichée = _player.userName + "\n" + _player.organizationName
  + "\n" + _player.serialNumber;
  member("Infos utilisateur").text = chaîneAffichée;
}
```

Voir aussi

Lecteur

userName (RealMedia)

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.userName

// Syntaxe JavaScript
réfObjActeurOuImageObjet.userName;
```

Description

Propriété d'acteur et image-objet RealMedia ; permet de définir le nom d'utilisateur nécessaire à l'accès à un flux RealMedia protégé. Vous ne pouvez pas utiliser cette propriété pour récupérer un nom d'utilisateur spécifié auparavant. Si aucun nom d'utilisateur n'a encore été défini, la valeur de cette propriété est la chaîne `"********"`. La valeur par défaut de cette propriété est une chaîne vide, ce qui signifie qu'aucun nom d'utilisateur n'a été spécifié.

Exemple

Les exemples suivants indiquent que le nom d'utilisateur pour le flux RealMedia de l'acteur Real ou de l'image-objet 2 a été défini.

```
-- Syntaxe Lingo
put(sprite(2).userName) -- "********"
put(member("Real").userName) -- "********"

// Syntaxe JavaScript
put(sprite(2).userName); // "********"
put(member("Real").userName); // "********"
```

Les exemples suivants indiquent que le nom d'utilisateur pour le flux RealMedia de l'acteur Real ou de l'image-objet 2 n'a jamais été défini.

```
-- Syntaxe Lingo
put(sprite(2).userName) -- ""
put(member("Real").userName) -- ""

// Syntaxe JavaScript
put(sprite(2).userName); // ""
put(member("Real").userName); // ""
```

Les exemples suivants indiquent que le nom d'utilisateur pour le flux RealMedia de l'acteur Real et de l'image-objet 2 est Marcel.

```
-- Syntaxe Lingo
member("Real").userName = "Marcel"
sprite(2).userName = "Marcel"

// Syntaxe JavaScript
member("Real").userName = "Marcel";
sprite(2).userName = "Marcel";
```

Voir aussi

password

useTargetFrameRate

Utilisation

```
sprite(quelleImageObjet3D).useTargetFrameRate
```

Description

Propriété 3D d'image-objet ; détermine si la propriété `targetFrameRate` de l'image-objet est appliquée. Si la propriété `useTargetFrameRate` a pour valeur `TRUE`, le nombre de polygones des modèles de l'image-objet est réduit pour atteindre le taux d'images spécifié ciblé.

Exemple

Ces instructions donnent à la propriété `targetFrameRate` de l'image-objet 3 la valeur 45 et appliquent la cadence d'image en donnant à la propriété `useTargetFrameRate` de l'image-objet la valeur `TRUE` :

```
sprite(3).targetFrameRate = 45
sprite(3).useTargetFrameRate = TRUE
```

Voir aussi

`targetFrameRate`

vertex

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.vertex[quellePositionDeSommet]

// Syntaxe JavaScript
réfObjActeur.vertex[quellePositionDeSommet];
```

Description

Expression de sous-chaîne ; permet d'accéder directement à certaines parties d'une liste de sommets d'un acteur forme vectorielle.

Utilisez cette sous-chaîne pour éviter de devoir analyser différentes sous-chaînes de la liste de sommets. L'utilisation de ce type d'expression de sous-chaîne permet à la fois de tester et de définir les valeurs de la liste de sommets.

Exemple

Le code suivant présente la définition du nombre de points de sommet dans un acteur :

```
-- Syntaxe Lingo
put(member("Archie").vertex.count) -- 2

// Syntaxe JavaScript
put(member("Archie").vertex.count); // 2
```

Pour obtenir le second sommet de l'acteur, vous pouvez utiliser la syntaxe suivante :

```
-- Syntaxe Lingo
put(member("Archie").vertex[2]) -- point(66.0000, -5.0000)

// Syntaxe JavaScript
put(member("Archie").vertex[2]) -- point(66.0000, -5.0000)
```

Vous pouvez également définir la valeur d'une poignée de contrôle :

```
-- Syntaxe Lingo
member("Archie").vertex[2].handle1 = point(-63.0000, -16.0000)

// Syntaxe JavaScript
member("Archie").vertex[2].handle1 = point(-63.0000, -16.0000);
```

Voir aussi

vertexList

vertexList

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.vertexList

// Syntaxe JavaScript
réfObjActeur.vertexList;
```

Description

Propriété d'acteur ; renvoie une liste linéaire contenant des listes de propriétés, une pour chaque sommet d'une forme vectorielle. La liste de propriétés contient l'emplacement du sommet et la poignée de contrôle. Si l'emplacement a la valeur (0,0), il n'y a pas de poignée de contrôle.

Chaque sommet peut avoir deux poignées de contrôle déterminant la courbe entre ce sommet et les sommets adjacents. Dans `vertexList`, les coordonnées des poignées de contrôle d'un sommet ont des valeurs relatives au sommet, plutôt que des valeurs absolues dans le système des coordonnées de la forme. Si la première poignée de contrôle d'un sommet est située 10 pixels à gauche de ce sommet, son emplacement est enregistré sous la valeur (-10, 0). Par conséquent, lorsque l'emplacement d'un sommet est modifié avec Lingo ou la syntaxe JavaScript, les poignées de contrôle se déplacent avec le sommet et n'ont pas besoin d'être mises à jour (sauf si l'utilisateur veut absolument changer l'emplacement ou la taille de la poignée).

En cas de modification de cette propriété, sachez que vous devez réinitialiser le contenu de la liste après avoir changé l'une des valeurs. En effet, lorsque vous affectez une variable à la valeur de la propriété, vous placez une copie de la liste, et non la liste elle-même, dans la variable. Pour appliquer un changement, utilisez une syntaxe comme :

```
-- accéder au contenu de la propriété courante
listeDesSommetsCourante = member(1).vertexList
-- ajouter 25 pixels aux positions horizontale et verticale du premier sommet
  de la liste
listeDesSommetsCourante [1] .vertex = listeDesSommetsCourante[1] .vertex +
  point(25, 25)
-- réinitialiser la propriété réelle sur la nouvelle position calculée
member(1).vertexList = listeDesSommetsCourante
```

Exemple

L'instruction suivante affiche la valeur `vertexList` pour une ligne arquée comportant deux sommets :

```
-- Syntaxe Lingo
put(member("Archie").vertexList)
  -- [[#vertex: point(-66.0000, 37.0000),  \
  #handle1: point(-70,0000, -36,0000),  \
  #handle2: point(-62,0000, 110,0000)], [#vertex: point(66,0000, -5,0000),  \
  #handle1: point(121,0000, 56,0000), #handle2: point(11,0000, -66,0000)]]

// Syntaxe JavaScript
put(member("Archie").vertexList);
  //[[#vertex: point(-66,0000, 37,0000), #handle1: point(-70.0000, -36.0000),
  #handle2: point(-62,0000, 110,0000)], [#vertex: point(66.0000, -5.0000),
  #handle1: point(121,0000, 56,0000), #handle2: point(11,0000, -66,0000)]]
```

Voir aussi

`addVertex()`, `count()`, `deleteVertex()`, `moveVertex()`, `moveVertexHandle()`, `originMode`, `vertex`

vertexList (générateur de maille)

Utilisation

`member(`*`quelActeur`*`).modelResource(`*`quelleRessourceDeModèle`*`).vertexList`

Description

Propriété 3D ; utilisée avec une ressource de modèle de type `#mesh`, cette propriété permet d'obtenir ou de définir la propriété `vertexList` de la ressource de modèle.

La `vertexList` est une liste linéaire de chaque sommet utilisé dans la maille. Un seul sommet peut être partagé par plusieurs faces de la maille. Vous pouvez spécifier une liste de n'importe quelle taille pour cette propriété, qui n'enregistrera cependant que le nombre d'éléments spécifié lors de l'utilisation de la commande `newMesh()` pour créer la ressource de modèle `#mesh`.

Exemple

L'instruction suivante définit la `vertexList` de la ressource de modèle Triangle.

```
member("Formes").modelResource("Triangle").vertexList = \
  [vector(0,0,0), vector(20,0,0), vector(20, 20, 0)]
```

Voir aussi

`newMesh`, `face`, `vertices`

vertexList (déformation de maille)

Utilisation

`member(`*`quelActeur`*`).model(`*`quelModèle`*`).meshDeform.mesh\`
`  [`*`index`*`].vertexList`

Description

Propriété 3D ; utilisée avec un modèle associé au modificateur `#meshDeform`, cette propriété permet d'obtenir ou de définir la propriété `vertexList` de la maille spécifiée du modèle référencé.

La `vertexList` est une liste linéaire de chaque sommet utilisé dans la maille spécifiée. Un seul sommet peut être partagé par plusieurs faces de la maille.

Lorsqu'un modèle utilise les modificateurs `#sds` ou `#lod` en plus du modificateur `#meshDeform`, il est important de ne pas oublier que la valeur de cette propriété change en fonction des modificateurs `#sds` ou `#lod`.

Exemple

L'instruction suivante affiche la `vertexList` du modificateur `#meshDeform` pour la première maille du modèle Triangle.

```
put member("Formes").model("Triangle").meshDeform.mesh[1].vertexList
-- [vector(0,0,0), vector(20,0,0), vector(20, 20, 0)]
```

Voir aussi

face, vertices, mesh (propriété)

vertices

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).\
  face[indexDeFaces].vertices
```

Description

Propriété 3D de face ; utilisée avec une ressource de modèle de type `#mesh`, cette propriété permet d'obtenir ou de définir les sommets de la propriété `vertexList` de la ressource à utiliser pour la face de la maille spécifiée par *indexDeFace*.

Cette propriété est une liste linéaire de trois entiers correspondant aux positions d'index des trois sommets, de la propriété `vertexList` de la maille, qui comprend la face spécifiée.

Les sommets doivent être spécifiés dans la liste dans un ordre antihoraire de façon à obtenir une normale de surface pointant vers l'extérieur.

Si vous apportez des modifications à cette propriété ou utilisez la commande `generateNormals()`, vous devrez appeler la commande `build()` pour reconstruire la maille.

Exemple

L'exemple suivant affiche la `vertexList` de la ressource de modèle de maille carréSimple, puis affiche la propriété `vertices` pour la seconde face de cette maille.

```
put member("3D").modelResource("CarréSimple").vertexList
-- [vector( 0.0000, 0.0000, 0.0000), vector( 0.0000, 5.0000, \
  0.0000), vector( 5.0000, 0.0000, 0.0000), vector( 5.0000, \
  5.0000, 0.0000)]
put member("3D").modelResource("CarréSimple").face[1].vertices
-- [3, 4, 1]
```

Voir aussi

face, vertexList (déformation de maille), generateNormals()

video (QuickTime, AVI)

Utilisation

```
member(quelActeur).video
the video of member quelActeur
```

Description

Propriété d'acteur vidéo numérique ; détermine si l'image graphique de l'acteur vidéo numérique spécifié est lue (TRUE ou 1) ou non (FALSE ou 0).

Seul l'élément visuel de l'acteur vidéo numérique est affecté. Par exemple, lorsque la propriété video a la valeur FALSE, la lecture de la piste audio de la vidéo (s'il en existe une) continue.

Exemple

L'instruction suivante désactive la vidéo associée à l'acteur Entrevue :

```
member("Entrevue").video = FALSE
```

Voir aussi

setTrackEnabled(), trackEnabled

video (RealMedia, Windows Media)

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.video

// Syntaxe JavaScript
réfObjActeurOuImageObjet.video;
```

Description

Propriété RealMedia et Windows Media ; permet de savoir ou de définir si l'image-objet ou l'acteur rend la vidéo (TRUE) ou seulement l'audio (FALSE). Lecture/écriture.

Les valeurs entières autres que 1 ou 0 sont traitées comme TRUE.

Utilisez cette propriété pour faire disparaître la vidéo lors de la lecture d'un composant audio d'un acteur RealMedia ou Windows Media, ou pour activer/désactiver la vidéo pendant la lecture.

Exemple

Les exemples suivants indiquent que la propriété vidéo de l'image-objet 2 et de l'acteur Real a pour valeur TRUE.

```
-- Syntaxe Lingo
put(sprite(2).video) -- 1
put(member("Real").video) -- 1

// Syntaxe JavaScript
put(sprite(2).video); // 1
put(member("Real").video); // 1
```

Les exemples suivants donnent à la propriété `video` de l'image-objet 2 et de l'acteur Real la valeur `FALSE`.

```
-- Syntaxe Lingo
sprite(2).video = FALSE
member("Real").video = FALSE

// Syntaxe JavaScript
sprite(2).video = 0;
member("Real").video = 0;
```

videoFormat

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.videoFormat

// Syntaxe JavaScript
réfObjDvd.videoFormat;
```

Description

Propriété DVD. Renvoie un symbole qui indique le format vidéo. En lecture seule.

Les symboles possibles sont les suivants :

Symbole	Description
#MPEG1	Le format vidéo est MPEG-1.
#MPEG2	Le format vidéo est MPEG-2.
#unknown	Le format vidéo est inconnu.

Voir aussi

DVD

videoForWindowsPresent

Utilisation

```
the videoForWindowsPresent
```

Description

Propriété système ; indique si un logiciel AVI est installé sur l'ordinateur.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante vérifie si Vidéo pour Windows est présent et, dans la négative, fait passer la tête de lecture au repère Autre séquence :

```
if the videoForWindowsPresent= FALSE then go to "Autre séquence"
```

Voir aussi

quickTimeVersion()

viewH

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.viewH

// Syntaxe JavaScript
réfObjActeurOuImageObjet.viewH;
```

Description

Propriété d'acteur et d'image-objet ; contrôle la coordonnée horizontale d'une animation Flash et le point de vue d'une forme vectorielle, exprimés en pixels. Les valeurs peuvent être des nombres à virgule flottante. La valeur par défaut est 0.

Le point de vue d'une animation Flash est défini par rapport à son point d'origine.

La définition d'une valeur positive pour `viewH` décale l'animation vers la gauche à l'intérieur de l'image-objet ; si une valeur négative est choisie, l'animation est décalée vers la droite. Par conséquent, la modification de la propriété `viewH` peut entraîner le recadrage de l'animation, voire son retrait total de l'écran.

Cette propriété peut être testée et définie.

Remarque : Cette propriété doit avoir la valeur par défaut si la propriété `scaleMode` a pour valeur `#autoSize`. Sinon, l'image-objet n'est pas correctement affichée.

Exemple

Le gestionnaire suivant accepte une référence d'image-objet comme paramètre et déplace la vue d'une image-objet animation Flash de la gauche vers la droite, à l'intérieur du rectangle de délimitation de cette image-objet :

```
-- Syntaxe Lingo
on panoramiqueDroite quelleImageObjet
  repeat with i = 120 down to -120
    sprite(quelleImageObjet).viewH = i
    _movie.updateStage()
  end repeat
end

// Syntaxe JavaScript
function panoramiqueDroite(quelleImageObjet) {
  var i = 120;
  while (i > -121) {
    sprite(quelleImageObjet).viewH = i;
    _movie.updateStage();
    i--;
  }
}
```

Voir aussi

`scaleMode`, `viewV`, `viewPoint`, `viewScale`

viewPoint

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.viewPoint

// Syntaxe JavaScript
réfObjActeurOuImageObjet.viewPoint;
```

Description

Propriété d'acteur et d'image-objet ; contrôle le point présent dans une animation Flash ou une forme vectorielle affiché au centre du rectangle de délimitation de l'image-objet, en pixels. Ces valeurs sont des entiers.

La modification du point de vue d'un acteur ne fait que modifier l'affichage d'une animation dans le rectangle de délimitation de l'image-objet et non l'emplacement de l'image-objet sur la scène. Le point de vue correspond à la coordonnée d'un acteur affiché au centre du rectangle de délimitation de l'image-objet et est toujours exprimé par rapport au point d'origine de l'animation (tel que défini par les propriétés `originPoint`, `originH` et `originV`). Par exemple, si vous définissez le point de vue d'une animation Flash sur point(100,100), le centre de l'image-objet est le point de l'animation Flash situé à 100 pixels (unités d'animation Flash) vers la droite et 100 pixels (unités d'animation Flash) vers le bas à partir du point d'origine, où que le point d'origine soit déplacé.

La propriété `viewPoint` est spécifiée sous la forme d'une valeur de point Director : par exemple, point (100,200). La définition du point de vue d'une animation Flash avec la propriété `viewPoint` équivaut à un réglage séparé des propriétés `viewH` et `viewV`. Par exemple, la définition de la propriété `viewPoint` sur point (50,75) équivaut au réglage de la propriété `viewH` sur 50 et de la propriété `viewV` sur 75.

Les valeurs de point Director spécifiées pour la propriété `viewPoint` sont limitées aux nombres entiers, alors que les propriétés `viewH` et `viewV` peuvent être définies avec des nombres à virgule flottante. Lorsque vous testez la propriété `viewPoint`, les valeurs de point sont tronquées pour donner des nombres entiers. En règle générale, utilisez les propriétés `viewH` et `viewV` pour obtenir plus de précision ; utilisez la propriété `originPoint` pour plus de rapidité et de facilité.

Cette propriété peut être testée et définie. La valeur par défaut est point (0,0).

Remarque : Cette propriété doit avoir la valeur par défaut si la propriété `scaleMode` a pour valeur `#autoSize`. Sinon, l'image-objet n'est pas correctement affichée.

Exemple

Le gestionnaire suivant a pour effet de déplacer une image-objet animation Flash vers le bas et vers la droite par incréments de cinq pixels en unités d'animation Flash :

```
-- Syntaxe Lingo
on panoramiqueTransversal(quelleImageObjet)
  repeat with i = 1 to 10
    sprite(quelleImageObjet).viewPoint = sprite(quelleImageObjet).viewPoint +
  \
  point(i * -5, i * -5)
    _movie.updateStage()
  end repeat
end
```

```
// Syntaxe JavaScript
function panoramiqueTransversal(quelleImageObjet) {
  var i = 1;
  while (i < 11) {
    sprite(quelleImageObjet).viewPoint = sprite(quelleImageObjet).viewPoint +
  point(i * -5, i * -5);
    _movie.updateStage();
    i++
  }
}
```

Voir aussi

scaleMode, viewV, viewH, viewScale

viewScale

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.viewScale

// Syntaxe JavaScript
réfObjActeurOuImageObjet.viewScale;
```

Description

Propriété d'acteur et d'image-objet ; définit la valeur d'ensemble permettant de mettre à l'échelle l'affichage d'une image-objet animation Flash ou forme vectorielle à l'intérieur du rectangle de délimitation de l'image-objet. Vous spécifiez la valeur en degrés sous la forme d'un nombre à virgule flottante. La valeur par défaut est 100.

Le rectangle de l'image-objet n'est pas mis à l'échelle ; seul l'affichage de l'acteur dans le rectangle l'est. La définition de la propriété `viewScale` d'une image-objet est similaire au choix d'un objectif pour un appareil-photo. Au fur et à mesure de la diminution de la valeur `viewScale`, la taille apparente de l'animation dans l'image-objet augmente, et vice versa. Par exemple, la définition de la propriété `viewScale` sur 200 % signifie que l'intérieur de l'image-objet affiché sera le double de ce qu'il était et que l'acteur placé dans l'image-objet sera réduit de moitié par rapport à sa taille d'origine.

Une différence notable entre les propriétés `viewScale` et `scale` est que la propriété `viewScale` effectue toujours une mise à l'échelle en partant du centre du rectangle de délimitation de l'image-objet, tandis que la propriété `scale` effectue la mise à l'échelle en partant d'un point déterminé par la propriété `originMode` de l'animation Flash.

Cette propriété peut être testée et définie.

Remarque : Cette propriété doit avoir la valeur par défaut si la propriété `scaleMode` a pour valeur `#autoSize`. Sinon, l'image-objet n'est pas correctement affichée.

Exemple

Le script d'image-objet suivant configure l'image-objet animation Flash et double l'échelle de son affichage :

```
-- Syntaxe Lingo
property spriteNum

on beginSprite me
  sprite(spriteNum).viewScale = 200
end

// Syntaxe JavaScript
function beginSprite() {
  sprite(this.spriteNum).viewScale = 200;
}
```

Voir aussi

scaleMode, viewV, viewPoint, viewH

viewV

Utilisation

```
-- Syntaxe Lingo
réfObjActeurOuImageObjet.viewV

// Syntaxe JavaScript
réfObjActeurOuImageObjet.viewV;
```

Description

Propriété d'acteur et d'image-objet ; contrôle la coordonnée verticale d'une animation Flash et le point de vue d'une forme vectorielle, exprimés en pixels. Les valeurs peuvent être des nombres à virgule flottante. La valeur par défaut est 0.

Le point de vue d'une animation Flash est défini par rapport à son point d'origine.

La définition d'une valeur positive pour `viewV` décale l'animation vers le haut à l'intérieur de l'image-objet ; si une valeur négative est choisie, l'animation est décalée vers le bas. Par conséquent, la modification de la propriété `viewV` peut entraîner le recadrage de l'animation, voire son retrait total de l'écran.

Cette propriété peut être testée et définie.

Remarque : Cette propriété doit avoir la valeur par défaut si la propriété `scaleMode` a pour valeur `#autoSize`. Sinon, l'image-objet n'est pas correctement affichée.

Exemple

Le gestionnaire suivant accepte une référence d'image-objet comme paramètre et déplace la vue d'une image-objet animation Flash du haut vers le bas, à l'intérieur du rectangle de délimitation de cette image-objet :

```
-- Syntaxe Lingo
on panoramiqueVersLeBas(quelleImageObjet)
  repeat with i = 120 down to -120
    sprite(quelleImageObjet).viewV = i
    _movie.updateStage()
  end repeat
end
```

```
// Syntaxe JavaScript
function panoramiqueVersLeBas(quelleImageObjet) {
  var i = 120;
  while (i > -121) {
    sprite(quelleImageObjet).viewV = i;
    _movie.updateStage();
    i--;
  }
}
```

Voir aussi

scaleMode, viewV, viewPoint, viewH

visible

Utilisation

```
-- Syntaxe Lingo
réfObjFenêtre.visible

// Syntaxe JavaScript
réfObjFenêtre.visible;
```

Description

Propriété de fenêtre ; détermine si une fenêtre est visible (TRUE) ou non (FALSE). Lecture/écriture.

Exemple

L'instruction suivante rend la fenêtre Tableau_de_commande visible :

```
-- Syntaxe Lingo
window("Tableau_de_commande").visible = TRUE

// Syntaxe JavaScript
window("Tableau_de_commande").visible = true;
```

Voir aussi

Fenêtre

visible (image-objet)

Utilisation

```
sprite(quelleImageObjet).visible
the visible of sprite quelleImageObjet
```

Description

Propriété d'image-objet ; détermine si l'image-objet spécifiée par *quelleImageObjet* est visible (TRUE) ou non (FALSE). Cette propriété affecte toutes les images-objets de la piste, quelle que soit leur position dans le scénario.

Remarque : Si la propriété `visible` d'une piste d'image-objet a la valeur `FALSE`, l'image-objet est invisible et seuls les événements liés à la souris ne sont pas envoyés à cette piste. L'envoi des événements `beginSprite`, `endSprite`, `prepareFrame`, `enterFrame` et `exitFrame` continue sans que le réglage de visibilité de l'image-objet soit pris en compte. Cependant, le fait de cliquer sur le bouton Désactiver la piste du scénario a pour effet d'affecter à la propriété `visible` la valeur `FALSE` et d'empêcher l'envoi de tous les événements à cette piste. La sélection de ce bouton désactive la piste, alors que le réglage de la propriété `visible` d'une image-objet sur la valeur `FALSE` n'affecte qu'une propriété graphique.

Cette propriété peut être testée et définie. Si cette propriété a la valeur `FALSE`, elle n'est pas automatiquement réinitialisée sur `TRUE` à la fin de l'image-objet. Vous devez donner à la propriété `visible` de l'image-objet la valeur `TRUE` afin de voir tous les autres acteurs utilisant cette piste.

Exemple

L'instruction suivante rend l'image-objet 8 visible :

```
sprite(8).visible = TRUE
```

visibility

Utilisation

```
member(quelActeur).model(quelModèle).visibility
référenceDobjetDeModèle.visibility
```

Description

Propriété 3D ; permet d'obtenir ou de définir la propriété `visibility` du modèle référencé. Cette propriété détermine la façon dont la géométrie du modèle est dessinée. Elle peut avoir une des valeurs suivantes :

- `#none` spécifie de ne pas tracer les polygones et que le modèle est invisible.
- `#front` spécifie que seuls les polygones faisant face à la caméra seront dessinés. Cette méthode optimise la vitesse de rendu. Il s'agit du paramètre par défaut de cette propriété.
- `#back` spécifie que seuls les polygones en direction opposée à la caméra seront dessinés. Utilisez ce paramètre lorsque vous souhaitez dessiner l'intérieur d'un modèle ou pour les modèles qui ne sont pas correctement dessinés, peut-être parce qu'ils ont été importés à partir d'un format de fichier utilisant une valeur différente pour le calcul des normales.
- `#both` spécifie que les deux côtés de tous les polygones sont dessinés. Utilisez ce paramètre lorsque vous souhaitez voir le plan quelle que soit la direction et pour les modèles qui ne sont pas correctement dessinés.

Exemple

L'instruction suivante indique que la propriété de visibilité du modèle Monstre02 a pour valeur `#none`. Le modèle est invisible.

```
put member("3D").model("Monstre02").visibility
-- #none
```

volume (DVD)

Utilisation

```
-- Syntaxe Lingo
réfObjDvd.volume

// Syntaxe JavaScript
réfObjDvd.volume;
```

Description

Propriété DVD. Détermine le volume courant. Lecture/écriture.

Le volume doit être un entier compris entre 0 (silencieux) et 100 (volume maximum).

Voir aussi

DVD

volume (acteur)

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.volume

// Syntaxe JavaScript
réfObjActeur.volume;
```

Description

Propriété d'acteur Shockwave Audio (SWA) ; détermine le volume de l'acteur SWA lu en flux continu spécifié. Les valeurs utilisables sont comprises entre 0 et 255.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante place le volume de l'acteur SWA lu en flux continu sur la moitié du volume maximum :

```
-- Syntaxe Lingo
member("fichierSWA").volume = 128

// Syntaxe JavaScript
member("fichierSWA").volume = 128;
```

volume (piste audio)

Utilisation

```
-- Syntaxe Lingo
réfObjPisteAudio.volume

// Syntaxe JavaScript
réfObjPisteAudio.volume;
```

Description

Propriété de piste audio ; détermine le volume d'une piste audio. Lecture/écriture.

Les pistes audio sont numérotées 1, 2, 3, etc. jusqu'à 8. Les pistes 1 et 2 sont les pistes qui apparaissent dans le scénario.

La valeur de la propriété `volume` est comprise entre 0 (son désactivé) et 255 (volume maximum). Une valeur de 255 indique le volume maximum du système, contrôlé par la propriété `soundLevel` de l'objet Son, et les valeurs plus petites sont établies en fonction du volume total. Cette propriété permet à plusieurs pistes d'avoir des paramètres indépendants dans la plage disponible.

Plus la valeur de `volume` est basse, plus vous entendrez de bruit ou de souffle. L'utilisation de `soundLevel` peut produire moins de bruit, mais offre moins de contrôle.

Vous pourrez voir un exemple de `volume` dans une animation en consultant l'animation Sound Control du dossier Learning/Lingo, lui-même dans le dossier de Director.

Exemple

L'instruction suivante règle le volume de la piste audio 2 sur 130, ce qui représente un niveau sonore moyen :

```
-- Syntaxe Lingo
sound(2).volume = 130

// Syntaxe JavaScript
sound(2).volume = 130;
```

Voir aussi

`Piste audio`, `soundLevel`

volume (image-objet)

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.volume

// Syntaxe JavaScript
réfObjImageObjet.volume;
```

Description

Propriété d'image-objet ; contrôle le volume d'un acteur animation vidéo numérique ou Windows Media spécifié par un nom ou un numéro. Les valeurs du volume varient entre 0 et 256. Les valeurs inférieures ou égales à 0 correspondent au son désactivé. Les valeurs supérieures à 256 correspondent à un niveau sonore très élevé et provoquent une distorsion considérable.

Exemple

L'instruction suivante règle le volume de l'animation QuickTime exécutée dans la piste d'image-objet 7 sur 256, ce qui représente le volume sonore maximum :

```
-- Syntaxe Lingo
sprite(7).volume = 256

// Syntaxe JavaScript
sprite(7).volume = 256;
```

Voir aussi

soundLevel

volume (Windows Media)

Utilisation

```
-- Syntaxe Lingo
réfObjWindowsMedia.volume

// Syntaxe JavaScript
réfObjWindowsMedia.volume;
```

Description

Propriété d'image-objet Windows Media ; détermine le volume d'une image-objet Windows Media.

La valeur de cette propriété est un nombre entier compris entre 0 (muet) et 7 (fort).

Vous pouvez aussi définir cette propriété avec le menu Contrôle > Volume de Director.

Exemple

L'instruction suivante règle le volume de l'image-objet 7 sur 2 :

```
-- Syntaxe Lingo
sprite(7).volume = 2

// Syntaxe JavaScript
sprite(7).volume = 2;
```

Voir aussi

Windows Media

warpMode

Utilisation

```
-- Syntaxe Lingo
réfObjImageObjet.warpMode

// Syntaxe JavaScript
réfObjImageObjet.warpMode;
```

Description

Propriété d'image-objet QuickTime VR ; indique le type d'effet de torsion appliqué à un panorama.

Les valeurs possibles sont `#full` (complet), `#partial` (partiel) et `#none` (aucun).

Cette propriété peut être testée et définie. Lorsque cette propriété est testée et que les valeurs des modes statiques et de mouvement diffèrent, la valeur de la propriété est la valeur définie pour le mode courant. Lorsque définie, cette propriété détermine l'effet de torsion à la fois pour les modes statiques et de mouvement.

Exemple

L'instruction suivante donne à la propriété `warpMode` de l'image-objet 1 la valeur `#full` :

```
-- Syntaxe Lingo
sprite(1).warpMode = #full

// Syntaxe JavaScript
sprite(1).warpMode = symbol("full");
```

width

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.width
réfObjImage.width
réfObjImageObjet.width

// Syntaxe JavaScript
réfObjActeur.width;
réfObjImage.width;
réfObjImageObjet.width;
```

Description

Propriété d'acteur, d'image et d'image-objet ; pour les acteurs forme vectorielle, Flash, GIF animé, RealMedia, Windows Media, bitmap et forme, détermine la largeur, en pixels, d'un acteur. Lecture seule pour les acteurs et les objets image, lecture/écriture pour les images-objets.

Cette propriété n'affecte pas les acteurs champ et bouton.

Exemple

L'instruction suivante affecte la largeur de l'acteur 50 à la variable `hauteur` :

```
-- Syntaxe Lingo
hauteur = member(50).width

// Syntaxe JavaScript
var hauteur = member(50).width;
```

L'instruction suivante règle la largeur de l'image-objet 10 sur 26 pixels :

```
-- Syntaxe Lingo
sprite(10).width = 26

// Syntaxe JavaScript
sprite(10).width = 26;
```

L'instruction suivante affecte la largeur de l'image-objet numéro i + 1 à la variable `largeur` :

```
-- Syntaxe Lingo
largeur = sprite(i + 1).width

// Syntaxe JavaScript
var largeur = sprite(i + 1).width;
```

Voir aussi

`height`, `image (image)`, `Acteur`, `Image-objet`

width (3D)

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).width
référenceDobjetDeRessourceDeModèle.width
```

Description

Propriété 3D ; permet d'obtenir ou de définir la largeur du plan d'une ressource de modèle de type #box ou #plane. Cette propriété doit être supérieure à 0.0 et a un paramètre par défaut de 1.0. Pour les objets de type #box, la valeur par défaut de `width` est 50.0. Pour les objets de type #plane, la valeur par défaut est 1.0. La propriété `width` est mesurée le long de l'axe des x.

Exemple

L'instruction suivante donne au plan de la ressource de modèle Gazon une largeur de 250.0.

```
member("Univers 3D").modelResource("Gazon").width = 250.0
```

widthVertices

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).
widthVertices
référenceDobjetDeRessourceDeModèle.widthVertices
```

Description

Propriété 3D ; permet d'obtenir ou de définir le nombre de sommets (sous forme d'un nombre entier) sur l'axe des x d'une ressource de modèle de type #box ou #plane. Cette propriété doit être supérieure ou égale à 2 et a une valeur par défaut de 2.

Exemple

L'instruction suivante donne à la propriété `widthVertices` de la ressource de modèle Tour la valeur 10. Dix-huit polygones (2 * (10-1) triangles) seront utilisés pour définir la géométrie de la ressource de modèle le long de son axe des *x*.

```
member("Univers 3D").modelResource("Tour").widthVertices = 10
```

wind

Utilisation

```
member(quelActeur).modelResource(quelleRessourceDeModèle).wind
référenceDobjetDeRessourceDeModèle.wind
```

Description

Propriété 3D ; permet d'obtenir ou de définir la propriété `wind` d'une ressource de modèle de type `#particle`, sous la forme d'un vecteur.

Cette propriété `wind` définit la direction et la force du vent appliquées à toutes les particules au cours de chaque étape de la simulation. La valeur par défaut de cette propriété est `vector(0, 0, 0)`, qui spécifie qu'aucun vent n'est appliqué.

Exemple

```
put member("3D").modelResource("nappe de brouillard").wind
-- vector(10.5,0,0)
```

window

Utilisation

```
-- Syntaxe Lingo
_player.window[nomOuNumDeFenêtre]

// Syntaxe JavaScript
_player.window[nomOuNumDeFenêtre];
```

Description

Propriété de lecteur ; fournit un accès nommé ou indexé aux objets Window créés par le lecteur Director. En lecture seule.

L'argument *nomOuNumDeFenêtre* est une chaîne spécifiant le nom de la fenêtre à laquelle accéder ou un entier spécifiant la position d'index de la fenêtre à laquelle accéder.

La fonctionnalité de cette propriété est identique à la méthode `window()` de niveau supérieur.

Exemple

Cette instruction affecte la variable `maFenêtre` au troisième objet fenêtre :

```
-- Syntaxe Lingo
maFenêtre = _player.window[3]

// Syntaxe JavaScript
var maFenêtre = _player.window[3];
```

Voir aussi

Lecteur, window()

windowBehind

Utilisation

```
-- Syntaxe Lingo
réfObjFenêtre.windowBehind

// Syntaxe JavaScript
réfObjFenêtre.windowBehind;
```

Description

Propriété de fenêtre ; renvoie une référence à la fenêtre se trouvant derrière toutes les autres fenêtres. En lecture seule.

Exemple

Ces instructions affectent la variable `fenêtreArrière` à la fenêtre qui se trouve derrière toutes les autres fenêtres, puis placent cette fenêtre au premier plan :

```
-- Syntaxe Lingo
fenêtreArrière = _player.windowList[5].windowBehind
fenêtreArrière.moveToFront()

// Syntaxe JavaScript
var fenêtreArrière = _player.windowList[5].windowBehind;
fenêtreArrière.moveToFront();
```

Voir aussi

moveToBack(), moveToFront(), Fenêtre, windowInFront, windowList

windowInFront

Utilisation

```
-- Syntaxe Lingo
réfObjFenêtre.windowInFront

// Syntaxe JavaScript
réfObjFenêtre.windowInFront;
```

Description

Propriété de fenêtre ; renvoie une référence à la fenêtre se trouvant devant toutes les autres fenêtres. En lecture seule.

Exemple

Ces instructions affectent la variable `fenêtreAvant` à la fenêtre qui se trouve devant toutes les autres fenêtres, puis placent cette fenêtre à l'arrière-plan :

```
-- Syntaxe Lingo
fenêtreAvant = _player.windowList[5].windowInFront
fenêtreAvant.moveToBack()

// Syntaxe JavaScript
var fenêtreAvant = _player.windowList[5].windowInFront
fenêtreAvant.moveToBack();
```

Voir aussi

moveToBack(), moveToFront(), Fenêtre, windowBehind, windowList

windowList

Utilisation

```
-- Syntaxe Lingo
_player.windowList

// Syntaxe JavaScript
_player.windowList;
```

Description

Propriété de lecteur ; affiche une liste des références à toutes les fenêtres d'animation recensées. En lecture seule.

La scène est également considérée comme une fenêtre.

Exemple

L'instruction suivante affiche une liste de toutes les fenêtres d'animation connues dans la fenêtre Messages :

```
-- Syntaxe Lingo
trace(_player.windowList)

// Syntaxe JavaScript
trace(_player.windowList);
```

Voir aussi

Lecteur

wordWrap

Utilisation

```
-- Syntaxe Lingo
réfObjActeur.wordWrap

// Syntaxe JavaScript
réfObjActeur.wordWrap;
```

Description

Propriété d'acteur champ ; détermine si le retour à la ligne automatique est autorisé (TRUE) ou non (FALSE).

Exemple

L'instruction suivante désactive la fonction de retour à la ligne automatique pour l'acteur champ Pierre :

```
-- Syntaxe Lingo
member("Pierre").wordWrap = FALSE

// Syntaxe JavaScript
member("Pierre").wordWrap = false;
```

worldPosition

Utilisation

```
member(quelActeur).model(quelModèle).worldPosition
member(quelActeur).light(quelleLumière).worldPosition
member(quelActeur).camera(quelleCaméra).worldPosition
member(quelActeur).group(quelGroupe).worldPosition
```

Description

Propriété 3D ; permet d'obtenir, mais pas de définir, la position d'un nœud avec les coordonnées de l'univers. Un nœud peut être un modèle, un groupe, une caméra ou une lumière. Cette propriété à un résultat équivalent à celui de la commande `getWorldTransform().position`. La position d'un nœud est représentée par un objet vecteur.

Exemple

L'instruction suivante indique que la position du modèle Mars, en coordonnées de l'univers, est `vector(-1333.2097, 0.0000, -211.0973)`.

```
put member("Séquence").model("Mars").worldPosition
-- vector( -211,0973, 1.0000, 0.0000 )
```

Voir aussi

`getWorldTransform()`, `position (transformation)`

worldTransform

Utilisation

```
member(quelActeur).model(quelModèle).bonesPlayer.bone[index].\
  worldTransform
```

Description

Propriété 3D du modificateur `bonesPlayer` ; permet d'obtenir la transformation relative à l'univers d'un segment spécifique, au contraire de la propriété `transform`, qui renvoie la transformation relative au parent du segment. La propriété `worldTransform` ne peut être utilisée qu'avec des modèles associés au modificateur `bonesPlayer`.

Exemple

L'instruction suivante enregistre la transformation relative à l'univers d'un segment dans la variable `finalTransform` :

```
transformFinale =
  member("3D").model("bipède").bonesPlayer.bone[3].worldTransform
```

Voir aussi

`bone`, `getWorldTransform()`, `transform (propriété)`

wrapTransform

Utilisation

```
member(quelActeur).shader(nomDeMatériau).wrapTransform
member(quelActeur).shader[indexDeMatériau].wrapTransform
member(quelActeur).model[nomDeModèle].shader.wrapTransform
member( quelActeur ).model.shaderlist[ indexDeListeDeMatériaux ].\
  wrapTransform
```

Description

Propriété 3D de matériau standard ; cette propriété donne accès à une transformation qui fait correspondre les coordonnées de texture en fonction de la `texture` du matériau. Faites pivoter cette transformation pour changer la façon dont la texture est projetée sur la surface d'un modèle. La texture proprement dite n'est pas affectée ; la transformation ne modifie que l'orientation de la texture appliquée par le matériau.

Remarque : Remarque : Cette commande n'a d'effet que lorsque la `textureModeList` a pour valeur `#planar`, `#spherical` ou `#cylindrical`.

Exemple

L'instruction suivante donne au `transformMode` du matériau Mat2 la valeur `#wrapCylindrical`, puis fait pivoter cette projection cylindrique de 90° autour de l'axe des *x* de façon à ce que le placage cylindrique s'enroule autour de l'axe des *y* au lieu de l'axe des *z*.

```
s = member("Séquence").shader("Mat2")
s.textureMode= #wrapCylindrical
s.wrapTransform.rotate(90.0, 0.0, 0.0)
```

wrapTransformList

Utilisation

```
member( quelActeur ).shader( NomDeMatériau ).wrapTransformList\
  [ indexDeCoucheDeTexture ]
member( quelActeur ).shader[ indexDeListeDeMatériaux ].\
  wrapTransformList[ indexDeCoucheDeTexture ]
member( quelActeur ).model( nomDeModèle ).\
  shader.wrapTransformList[ indexDeCoucheDeTexture ]
member( quelActeur ).model( nomDeModèle ).shaderList\
  [ indexDeListeDeMatériaux ]. wrapTransformList[ indexDeCoucheDeTexture ]
```

Description

Propriété 3D de matériau standard ; donne accès à une transformation qui modifie les coordonnées de texture d'une couche de texture spécifiée. Faites pivoter cette transformation pour changer la façon dont la texture est projetée sur la surface des modèles. La texture même n'est pas affectée, la transformation ne modifiant que la façon dont le matériau applique la texture.

Remarque : wrapTransformList[indexDeCoucheDeTexture] n'a d'effet que lorsque `textureModeList[indexDeCoucheDeTexture]` a pour valeur `#planar`, `#spherical` ou `#cylindrical`.

Exemple

La ligne 2 de cet exemple donne à la propriété `transformMode` de la troisième couche de texture du matériau Mat2 la valeur `#wrapCylindrical`. La ligne 3 fait pivoter cette projection cylindrique de 90° autour de l'axe des *x* de façon à ce que le placage cylindrique s'enroule autour de l'axe des *y* au lieu de l'axe des *z*.

```
s = member("Séquence").shader("Mat2")
s.textureModeList[3] = #wrapCylindrical
s.wrapTransformList[3].rotate(90.0, 0.0, 0.0)
```

Voir aussi

`newShader`, `textureModeList`

x (vecteur)

Utilisation

```
member(quelActeur).vector.x
member(quelActeur).vector[1]
```

Description

Propriété 3D ; permet d'obtenir ou de définir le composant *x* d'un vecteur.

Exemple

L'instruction suivante indique le composant *x* d'un vecteur.

```
vec = vector(20, 30, 40)
put vec.x
-- 20.0000
```

xAxis

Utilisation

```
member(quelActeur).transform.xAxis
```

Description

Propriété 3D de transformation ; permet d'obtenir, mais pas de définir, le vecteur représentant l'axe des *x* canonique dans l'espace de transformation.

Exemple

La première ligne de cet exemple définit la transformation du modèle cylindreMod en transformation d'identité. Les deux lignes suivantes indiquent que l'axe des *x* de cylindreMod est `vector(1.0000, 0.0000, 0.0000)`. Cela signifie que l'axe des *x* de cylindreMod est aligné sur l'axe des *x* de l'univers. La ligne suivante fait pivoter cylindreMod de 90° autour de son axe des *y*. Les axes de cylindreMod sont également pivotés. Les deux dernières lignes indiquent que l'axe des *x* de cylindreMod est maintenant vector(0.0000, 0.0000, 1,0000). Cela signifie que l'axe des *x* de cylindreMod est maintenant aligné sur l'axe des *z* négatif de l'univers.

```
member("Moteur").model("cylindreMod").transform.identity()
put member("Moteur").model("cylindreMod").transform.xAxis
-- vector( 0.0000, 1.0000, 0.0000 )
member("Moteur").model("cylindreMod").rotate(0, 90, 0)
put member("Moteur").model("cylindreMod").transform.xAxis
-- vector( 0.0000, 0.0000, -1.0000 )
```

xtra

Utilisation

```
-- Syntaxe Lingo
_player.xtra[nomOuNumXtra]

// Syntaxe JavaScript
_player.xtra[nomOuNumXtra];
```

Description

Propriété de lecteur ; fournit un accès nommé ou indexé aux Xtras disponibles sur le lecteur Director. En lecture seule.

L'argument *nomOuNumXtra* est une chaîne spécifiant le nom de l'Xtra auquel accéder ou un entier spécifiant la position d'index de l'Xtra auquel accéder.

La fonctionnalité de cette propriété est identique à la méthode `xtra()` de niveau supérieur.

Exemple

L'instruction suivante affecte la variable monXtra à l'Xtra Texte en voix :

```
-- Syntaxe Lingo
monXtra = _player.xtra["XtraTexteEnVoix"]

// Syntaxe JavaScript
var monXtra = _player.xtra["XtraTexteEnVoix"];
```

Voir aussi

`Lecteur`, `xtra()`

xtraList (animation)

Utilisation

```
-- Syntaxe Lingo
_movie.xtraList

// Syntaxe JavaScript
_movie.xtraList;
```

Description

Propriété d'animation ; affiche une liste linéaire de tous les Xtras ajoutés à l'animation dans la boîte de dialogue Xtras de l'animation. En lecture seule.

Deux propriétés différentes peuvent figurer dans la propriété `xtraList` :

- `#filename` – Spécifie le nom de fichier de l'Xtra sur la plate-forme courante. Il est possible que la liste ne contienne pas d'entrée `#filename`, comme lorsque l'Xtra n'existe que sur une seule plate-forme.
- `#packageurl` – Indique l'emplacement, sous la forme d'une adresse URL, du dossier de téléchargement spécifié par `#packagefiles`.
- `#packagefiles` – Ne s'utilise que lorsque l'Xtra sélectionné doit être téléchargé. La valeur de cette propriété est une autre liste contenant la liste de propriétés de chaque fichier du dossier de téléchargement pour la plate-forme utilisée. Les propriétés de cette liste de sous-propriétés sont `#name` et `#version` et contiennent les mêmes informations que la liste `xtraList (Lecteur)`.

Exemple

L'instruction suivante affiche la liste `xtraList` dans la fenêtre Messages :

```
-- Syntaxe Lingo
put(_movie.xtraList)

// Syntaxe JavaScript
put(_movie.xtraList);
```

Voir aussi

`Animation`, `xtraList (lecteur)`

xtraList (lecteur)

Utilisation

```
-- Syntaxe Lingo
_player.xtraList

// Syntaxe JavaScript
_player.xtraList;
```

Description

Propriété de lecteur ; affiche une liste linéaire des propriétés de tous les Xtras disponibles et des versions de leurs fichiers. En lecture seule.

Cette propriété est utile lorsque la fonctionnalité d'une animation dépend d'une certaine version d'un Xtra.

Deux types de propriétés peuvent apparaître dans `xtraList` :

- `#filename` – Spécifie le nom de fichier de l'Xtra sur la plate-forme courante. Il est possible que la liste ne contienne pas d'entrée `#filename`, comme lorsque l'Xtra n'existe que sur une seule plate-forme.
- `#version` – Spécifie le même numéro de version apparaissant dans la boîte de dialogue Propriétés (Windows) ou Lire les informations (Macintosh) lorsque le fichier est sélectionné sur le bureau. Un Xtra ne possède pas obligatoirement de numéro de version.

Exemple

Cette instruction affiche, dans la fenêtre Messages, tous les Xtras de médias disponibles sur le lecteur Director.

```
-- Syntaxe Lingo
trace(_player.xtraList)

// Syntaxe JavaScript
trace(_player.xtraList);
```

Voir aussi

`mediaXtraList`, `Lecteur`, `scriptingXtraList`, `toolXtraList`, `transitionXtraList`

y (vecteur)

Utilisation

```
member(quelActeur).vector.y
member(quelActeur).vector[2]
```

Description

Propriété 3D ; permet d'obtenir ou de définir le composant *y* d'un vecteur.

Exemple

L'instruction suivante indique le composant *y* d'un vecteur.

```
vec = vector(20, 30, 40)
put vec.y
-- 30.0000
```

yAxis

Utilisation

```
member(quelActeur).transform.yAxis
```

Description

Propriété 3D de transformation ; permet d'obtenir, mais pas de définir, le vecteur représentant l'axe des *y* canonique dans l'espace de transformation.

Exemple

La première ligne de cet exemple définit la transformation du modèle cylindreMod en transformation d'identité. Les deux lignes suivantes indiquent que l'axe des y de cylindreMod est `vector(0.0000, 1.0000, 0.0000)`. Cela signifie que l'axe des *y* de cylindreMod est aligné sur l'axe des *y* de l'univers. La ligne suivante fait pivoter cylindreMod de 90° autour de son axe des *x*. Les axes de cylindreMod sont également pivotés. Les deux dernières lignes indiquent que l'axe des *x* de cylindreMod est maintenant vector(0.0000, 0.0000, 1,0000). Cela signifie que l'axe des *y* de cylindreMod est maintenant aligné sur l'axe des *z* positif de l'univers.

```
member("Moteur").model("cylindreMod").transform.identity()
put member("Moteur").model("cylindreMod").transform.yAxis
-- vector( 1.0000, 0.0000, 0.0000 )
member("Moteur").model("cylindreMod").rotate(90, 0, 0)
put member("Moteur").model("cylindreMod").transform.yAxis
-- vector( 0.0000, 0.0000, 1.0000 )
```

yon

Utilisation

```
member(quelActeur).camera(quelleCaméra).yon
```

Description

Propriété 3D ; permet d'obtenir ou de définir la distance, depuis la caméra, définissant l'emplacement de recadrage du frustrum de la vue le long de l'axe des z. Les objets à une distance supérieure à `yon` ne sont pas dessinés.

La valeur par défaut de cette propriété est 3.40282346638529e38.

Exemple

L'instruction suivante donne à la propriété `yon` de la caméra 1 la valeur 50000.

```
member("Univers 3D").camera[1].yon = 50000
```

Voir aussi

`hither`

z (vecteur)

Utilisation

```
member(quelActeur).vector.z
member(quelActeur).vector[3]
```

Description

Propriété 3D ; permet d'obtenir ou de définir le composant *z* d'un vecteur.

Exemple

L'instruction suivante indique le composant *z* d'un vecteur.

```
vec = vector(20, 30, 40)
put vec.z
-- 40.0000
```

zAxis

Utilisation

```
member(quelActeur).transform.zAxis
```

Description

Propriété 3D de transformation ; permet d'obtenir, mais pas de définir, le vecteur représentant l'axe des *z* canonique dans l'espace de transformation.

Exemple

La première ligne de cet exemple définit la transformation du modèle cylindreMod en transformation d'identité. Les deux lignes suivantes indiquent que l'axe des *z* de cylindreMod est `vector(0.0000, 0.0000, 1.0000)`. Cela signifie que l'axe des *z* de cylindreMod est aligné sur l'axe des *z* de l'univers. La ligne suivante fait pivoter cylindreMod de 90° autour de son axe des *y*. Les axes de cylindreMod sont également pivotés. Les deux dernières lignes indiquent que l'axe des *z* de cylindreMod est maintenant vector(1.0000, 0.0000, 0.0000). Cela signifie que l'axe des *z* de cylindreMod est maintenant aligné sur l'axe des *x* de l'univers.

```
member("Moteur").model("cylindreMod").transform.identity()
put member("Moteur").model("cylindreMod").transform.zAxis
-- vector( 0.0000, 1.0000, 0.0000 )
member("Moteur").model("cylindreMod").rotate(0, 90, 0)
put member("Moteur").model("cylindreMod").transform.zAxis
-- vector( 0.0000, 0.0000, -1.0000 )
```

INDEX

Symboles

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

z