Dictionnaire Lingo

Macromedia Director MX

Remerciements

Rédaction : Jay Armstrong, George Brown, Stephanie Gowin et Tim Statler

Révision : Rosana Francescato, Mary Ferguson, Mary Kraemer et Noreen Maher

Gestion du projet : Stuart Manning

Production : Chris Basmajian, Caroline Branch, John Francis et Patrice O'Neill

Multimédia : Aaron Begley et Benjamin Salles

Chef de projet Localisation : Yuko Yagi

Localisation : Gwenhaël Jacq, Masayo Noda, Florian de Joannès

Première édition : mars 2003

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

CHAPITRE 1
Lingo par fonction

Ce chapitre contient la liste des différentes fonctions de Macromedia Director MX et des éléments Lingo qui peuvent être utilisés pour les implémenter.

Accessibilité

Les termes suivants servent à rendre les animations accessibles :

Texte en voix

voiceCount()	voiceSet()
voiceGet()	voiceSetPitch()
voiceGetPitch()	voiceSetRate()
voiceGetRate()	voiceSetVolume()
voiceGetVolume()	voiceSpeak()
voiceInitialize()	voiceState()
voicePause()	voiceStop()
voiceResume()	voiceWordPos()

Navigation avec le clavier

autoTab	selection (propriété de distribution)
hilite (commande)	selectedText
keyboardFocusSprite	selEnd
selection (propriété d'acteur texte)	selStart
selection() (fonction)	

Acteurs

Les termes de cette section sont utilisés avec les acteurs.

Création d'acteurs

Utilisez `importFileInto` et `new()` pour créer des acteurs.

Programmation

Utilisez `duplicate member`, `erase member` et `pasteClipBoardInto` pour manipuler des acteurs pendant la programmation.

Acteurs graphiques

Les termes suivants sont utilisés pour vérifier et définir les images associées aux acteurs graphiques :

`center`	`palette`
`crop (propriété d'acteur)`	`picture (propriété d'acteur)`
`depth`	`pictureP()`
`media`	`regPoint`

Propriétés d'acteurs générales

Les termes suivants sont utilisés pour vérifier et définir des propriétés d'acteurs :

`fileName (propriété d'acteur)`	`number (propriété d'acteur)`
`media`	`preLoadMode`
`modified`	`type (propriété d'acteur)`
`name (propriété d'acteur)`	`URL`

Dimensions des acteurs graphiques

Utilisez `height`, `rect (acteur)` et `width` pour vérifier et définir les dimensions des acteurs graphiques.

Animation

Les termes suivants sont utilisés pour créer des séquences animées avec Lingo :

`blend`	`locV`
`ink`	`member (propriété d'image-objet)`
`loc`	`regPoint`
`locH`	`tweened`

Animations

Les termes de cette section sont utilisés pour gérer les animations.

Arrêt des animations

Les termes suivants sont utilisés pour arrêter ou fermer une animation ou une projection :

`exitLock`	`quit`
`halt`	`restart`
`pauseState`	`shutDown`

Informations sur les animations

Les termes suivants sont utilisés pour obtenir des informations sur une animation et son environnement :

`environment`	`moviePath`
`lastFrame`	`number (propriété système)`
`movie`	`runMode`
`movieFileFreeSize`	`safePlayer`
`movieFileSize`	`version`
`movieName`	`movieFileVersion`

Contrôle de la source

Les termes suivants sont utilisés pour gérer les projets Director sur lesquels plusieurs personnes travaillent :

`comments`	`creationDate`
`modifiedBy`	`modifiedDate`
`linkAs()`	`seconds`

Enregistrement des animations

Utilisez `saveMovie` et `updateMovieEnabled` pour enregistrer les modifications d'une animation.

Vérification des erreurs

Utilisez l'événement `alertHook` pour afficher des messages d'alerte décrivant des erreurs dans une projection.

Evénements dans les animations

Utilisez les gestionnaires d'événements `on prepareMovie`, `on startMovie` et `on stopMovie` contenant le code Lingo que vous souhaitez exécuter en réponse aux événements se produisant dans les animations.

Animations dans une fenêtre

Les termes de cette section sont utilisés avec les animations dans une fenêtre.

Evénements des animations dans une fenêtre

Utilisez les gestionnaires d'événements suivants contenant le code Lingo que vous souhaitez exécuter en réponse aux événements des animations dans une fenêtre :

`activeCastLib`	`on openWindow`
`on closeWindow`	`on resizeWindow`
`on moveWindow`	`on zoomWindow`

Ouverture et fermeture d'animations dans une fenêtre

Les termes suivants sont utilisés pour ouvrir et fermer des fenêtres :

`close window`	`open window`
`forget`	`windowList`

Apparence des fenêtres

Les termes suivants sont utilisés pour vérifier et définir l'apparence de la fenêtre d'une animation :

`drawRect`	`sourceRect`
`fileName (propriété de fenêtre)`	`tell`
`frontWindow`	`title`
`modal`	`titleVisible`
`moveToBack`	`visible (propriété de fenêtre)`
`moveToFront`	`windowPresent()`
`name (propriété de fenêtre)`	`windowType`
`rect (fenêtre)`	`appMinimize`

Communication entre différentes animations

Utilisez la commande `tell` pour envoyer des messages entre différentes animations.

Bitmaps

Les termes de cette section sont utilisés avec les bitmaps.

Propriétés des bitmaps

Les termes suivants sont utilisés pour vérifier et définir les propriétés des bitmaps :

alphaThreshold	foreColor
backColor	palette
blend	picture (propriété d'acteur)
depth	pictureP()
dither	rect (acteur)
trimWhiteSpace (propriété)	imageCompression
imageQuality	movieImageCompression
movieImageQuality	

Couche alpha

Les termes suivants sont utilisés pour contrôler les effets de couche alpha :

alphaThreshold	dither
depth	useAlpha
createMask()	createMatte()
extractAlpha()	setAlpha()

Objets image

Les termes suivants sont utilisés pour vérifier et contrôler les objets image :

copyPixels()	fill()
crop() (commande d'objet image)	image
draw()	image()
duplicate() (fonction d'image)	rect (image)
getPixel()	setPixel()

Boutons

Consultez Boutons et cases à cocher dans la section Interface.

Cadence

Utilisez la commande `puppetTempo` pour contrôler la cadence d'une animation.

Comportements

Les termes de cette section sont utilisés pour créer des comportements et les utiliser pendant la lecture d'une animation.

Création de comportements

Les termes suivants sont utilisés pour créer des comportements et en définir les paramètres dans la boîte de dialogue Paramètres :

ancestor	on getBehaviorDescription
on runPropertyDialog	on getPropertyDescriptionList
on getBehaviorTooltip	property
on isOKToAttach	

Envoi de messages aux comportements

Les commandes suivantes sont utilisées pour envoyer des messages aux comportements associés aux images-objets :

call	sendSprite
callAncestor	sendAllSprites

Identification des comportements

Les termes suivants sont utilisés pour identifier les comportements associés aux images-objets :

currentSpriteNum	scriptInstanceList
me	spriteNum

Distributions

Les termes de cette section sont utilisés avec les distributions.

Chargement des distributions

Utilisez `preLoadMode` pour vérifier et définir le moment auquel Director précharge une distribution.

Propriétés des distributions

Les termes suivants sont utilisés pour spécifier les propriétés des distributions :

castLib	number (propriété de distribution)
fileName (propriété de distribution)	number (propriété système)
name (propriété de distribution)	

Gestion des distributions

Les termes suivants sont utilisés pour gérer les distributions :

activeCastLib	number of members
duplicate member	pasteClipBoardInto
erase member	save castLib
findEmpty()	selection (propriété de distribution)
move member	

Interface

Les termes Lingo de cette section sont utilisés avec les éléments d'interface.

Menus

Les termes suivants sont utilisés pour créer des menus :

enabled	name (propriété d'élément de menu)
installMenu	number (éléments de menu)
menu	number (menus)
name (propriété de menu)	script

Boutons et cases à cocher

Les termes suivants sont utilisés pour spécifier des boutons et des cases à cocher :

alert	checkBoxType
buttonStyle	checkMark
buttonType	hilite (propriété d'acteur)
checkBoxAccess	

Esclaves

Les termes suivants sont utilisés pour contrôler la propriété `puppet` des pistes d'effets et des images-objets :

puppet	puppetTempo
puppetPalette	puppetTransition
puppetSound	updateStage
puppetSprite	

Evénements

Utilisez les gestionnaires d'événements suivants avec le code Lingo devant être exécuté lorsqu'un événement spécifique se produit :

activeCastLib	on moveWindow
close window	on mouseWithin
on cuePassed	open window
on deactivateWindow	on prepareFrame
on enterFrame	on prepareMovie
on EvalScript	on resizeWindow
on exitFrame	on mouseUpOutside
on idle	on rightMouseDown (gestionnaire d'événement)
on keyDown	on rightMouseUp (gestionnaire d'événement)
on keyUp	on startMovie
on mouseDown (gestionnaire d'événement)	on stepFrame
on mouseEnter	on streamStatus
on mouseLeave	on timeOut
on mouseUp (gestionnaire d'événement)	on zoomWindow
on stopMovie	on beginSprite
on endSprite	on hyperlinkClicked

Utilisez les commandes `pass` et `stopEvent` pour annuler l'ordre dans lequel Director envoie les messages.

Fichiers externes

Les termes de cette section sont utilisés avec les fichiers externes.

Chemins et noms de fichiers

Les termes suivants sont utilisés pour vérifier et définir les chemins et noms de fichiers :

@ (chemin d'accès)	getNthFileNameInFolder()
applicationPath	moviePath
fileName (propriété de distribution)	searchCurrentFolder
fileName (propriété d'acteur)	URL

Obtention de médias externes

Les termes suivants sont utilisés pour obtenir des médias externes :

downloadNetThing	preloadNetThing()
importFileInto	

Gestion des fichiers externes

Les termes suivants sont utilisés pour gérer les fichiers externes :

closeXlib	showXlib
open	sound playFile
openXlib	

Flash

Les termes suivants sont utilisés avec les acteurs Flash :

actionsEnabled	originV
broadcastProps	pathName (propriété d'animation)
bufferSize	pausedAtStart (Flash, vidéo numérique)
buttonsEnabled	percentStreamed
bytesStreamed	play
callFrame()	playBackMode
centerRegPoint	playing
clearError	posterFrame
clickMode	print()
defaultRect	printAsBitmap()
defaultRectMode	quality
directToStage	rewind sprite
endTellTarget(). Consultez tellTarget()	rotation
eventPassMode	scale
findLabel()	scaleMode
fixedRate	on sendXML
flashRect	setCallback()
flashToStage()	settingsPanel()
frame() (fonction)	setFlashProperty()
frame (propriété d'image-objet)	setVariable()
frameCount	showProps()
frameRate	sound
frameReady()	the soundMixMedia
getError()	sourceFileName
getFlashProperty()	stageToFlash()
getFrameLabel()	state (Flash, SWA)
getVariable()	static
goToFrame	stop (Flash)

hitTest()	stream
hold	streamMode
imageEnabled	streamSize
linked	tellTarget()
loop (mot-clé)	URL
mouseOverButton	viewH
newObject()	viewPoint
obeyScoreRotation	viewScale
originH	viewV
originMode	volume (propriété d'acteur)
originPoint	

Les termes suivants sont utilisés avec les objets Flash globaux, qui n'ont pas besoin d'un acteur Flash :

clearAsObjects()	setCallback()
newObject()	

Formes

Utilisez les termes Lingo suivants avec les formes :

filled	pattern
lineDirection	shapeType
lineSize	

Formes vectorielles

Les termes suivants sont utilisés avec les formes vectorielles :

addVertex	gradientType
antiAlias	imageEnabled
backgroundColor	moveVertex()
broadcastProps	moveVertexHandle()
centerRegPoint	originH
closed	originMode
defaultRect	originPoint
defaultRectMode	originV
deleteVertex()	rotation
directToStage	scale
endColor	scaleMode
fillColor	showProps()

fillCycles	skew
fillDirection	static
fillMode	strokeColor
fillOffset	strokeWidth
fillScale	vertexList
flashRect	viewPoint
flipH	viewScale
flipV	viewV
curve	newCurve()
regPointVertex	

GIF animés

Les termes suivants sont utilisés avec les GIF animés :

directToStage	pause (lecture d'animation)
frameRate	playBackMode
linked	resume sprite
moviePath	rewind sprite

Images

Les termes Lingo de cette section vous permettent de travailler avec des images.

Evénements images

Utilisez les gestionnaires d'événements `on enterFrame`, `on exitFrame` et `on prepareFrame` avec le code Lingo qui doit être exécuté lorsque des événements spécifiques se produisent dans une image.

Propriétés d'images

Utilisez les termes Lingo suivants pour vérifier et définir les propriétés d'images :

frameLabel	frameTempo
framePalette	frameTransition
frameScript	label()
frameSound1	labelList
frameSound2	marker()
the markerList	

Images-objets

Les termes Lingo de cette section sont utilisés pour travailler avec les images-objets.

Evénements d'images-objets

Utilisez les gestionnaires d'événements `on beginSprite` et `on endSprite` pour contenir des instructions Lingo que vous souhaitez exécuter lorsqu'une image-objet commence ou se termine.

Affectation d'acteurs aux images-objets

Utilisez `castLibNum`, `member (propriété d'image-objet)` ou `memberNum` pour spécifier l'acteur associé à une image-objet.

Rotation des images-objets

Utilisez la propriété d'image-objet `rotation` pour faire pivoter les images-objets.

Glissement d'images-objets

Les termes suivants sont utilisés pour définir comment l'utilisateur pourra faire glisser les images-objets :

`constrainH()`	`moveableSprite`
`constrainV()`	`sprite...intersects`
`constraint`	`sprite...within`

Images-objets et Lingo

Les termes suivants sont utilisés pour gérer la façon dont Lingo contrôle les images-objets :

`puppetSprite`	`spriteNum`
`puppet`	`sendSprite`
`scriptNum`	`sendAllSprites`
`scriptInstanceList`	

Dessin d'images-objets sur la scène

Les termes suivants sont utilisés pour contrôler la façon dont Director dessine les images-objets sur la scène :

`blend`	`skew`
`flipH`	`trails`
`flipV`	`tweened`
`ink`	`updateStage`
`quad`	`visible (propriété d'image-objet)`
`rotation`	

Dimensions des images-objets

Les termes suivants sont utilisés pour vérifier et définir la taille du rectangle de délimitation des images-objets :

bottom	right
height	top
left	width
quad	zoomBox

Vous pouvez également manipuler le rectangle de délimitation d'une image-objet avec les termes Lingo utilisés avec les rectangles. Pour plus d'informations, consultez Points et rectangles.

Emplacement des images-objets

Utilisez les propriétés d'image-objet `loc`, `locH` et `locV` pour vérifier et définir l'emplacement des images-objets.

Couleur des images-objets

Les termes suivants sont utilisés pour vérifier et définir la couleur d'une image-objet :

backColor	color (propriété d'image-objet et d'acteur)
bgColor	foreColor

Interaction avec la souris

Les termes de cette section sont utilisés avec le code Lingo se référant à l'utilisation de la souris.

Clics de la souris

Les termes suivants sont utilisés pour détecter ce que l'utilisateur fait avec la souris :

clickOn	mouseLine
doubleClick	mouseLoc
emulateMultiButtonMouse	mouseMember
lastClick()	mouseOverButton
lastEvent()	on mouseUp (gestionnaire d'événement)
lastRoll	mouseV
mouseChar	mouseWord
on mouseDown (gestionnaire d'événement)	on rightMouseDown (gestionnaire d'événement)
mouseH	on rightMouseUp (gestionnaire d'événement)
mouseItem	rollOver()
mouseLevel	stillDown

Evénements souris

Les termes suivants sont utilisés pour définir les gestionnaires qui répondent à la souris :

mouseDownScript	on mouseUp (gestionnaire d'événement)
mouseUpScript	on mouseUpOutside
on mouseDown (gestionnaire d'événement)	on mouseWithin
on mouseEnter	on rightMouseDown (gestionnaire d'événement)
on mouseLeave	on rightMouseUp (gestionnaire d'événement)

Contrôle du curseur

Utilisez `cursor (commande)`, `cursor (propriété d'image-objet)` et `cursorSize` pour contrôler le curseur.

Lingo

Les termes Lingo de cette section sont des éléments importants du langage de programmation Lingo que vous utilisez lors de la rédaction de scripts.

Valeurs booléennes

Les termes suivants sont utilisés pour vérifier si une condition existe :

- `FALSE` (`0` est l'équivalent numérique de `FALSE`).
- `TRUE` (`1` est l'équivalent numérique de `TRUE`).
- `not`
- `or`

Contrôle des scripts

Les termes suivants sont utilisés pour contrôler l'exécution des scripts :

abort	pass
do	result
exit	scriptsEnabled
halt	scriptText
nothing	stopEvent

Structures du code

Utilisez `if` pour créer des instructions `if..then`.

Utilisez `case`, `end case` et `otherwise` dans les instructions de cas.

Les termes suivants sont utilisés avec les boucles de répétition :

end repeat	repeat with
exit repeat	repeat with...down to
next repeat	repeat with...in list
repeat while	

Eléments de syntaxe

Les termes suivants sont utilisés dans la syntaxe de Lingo :

# (symbole)	member (mot-clé)
" (chaîne)	of
¬ (symbole de continuation)	or
-- (séparateur de commentaires)	property
() (parenthèses)	sprite
castLib	the
end	window
global	

Lingo réseau

Les termes de cette section sont utilisés avec les opérations réseau.

Téléchargement et lecture en flux continu des médias

Les termes suivants sont utilisés pour récupérer des médias ou les télécharger en flux continu depuis le réseau :

downloadNetThing (pour les projections et la programmation uniquement)	gotoNetPage
getNetText()	postNetText
gotoNetMovie	preloadNetThing()

Vérification de la disponibilité des médias

Utilisez `frameReady()` et `mediaReady` pour vérifier si le téléchargement d'un média spécifique est terminé.

Utilisation des opérations réseau

Les termes suivants sont utilisés pour vérifier la progression d'une opération réseau ou obtenir des informations sur les médias du réseau :

getStreamStatus()	netMIME()
getLatestNetID	netTextResult()
netAbort	on streamStatus
netDone()	proxyServer
netError()	tellStreamStatus()
netPresent	URLEncode
netLastModDate()	

Travail avec l'ordinateur local

Les termes suivants sont utilisés pour travailler avec l'ordinateur de l'utilisateur :

`browserName()`	`clearCache (pour les projections et la programmation uniquement)`
`cacheDocVerify() (pour les projections et la programmation uniquement)`	`getPref()`
`cacheSize() (pour les projections et la programmation uniquement)`	`setPref`

Navigateurs web

Utilisez `on EvalScript`, `externalEvent` et `netStatus` pour l'interactivité avec les navigateurs web. Pour plus d'informations sur la rédaction de scripts pour navigateurs web au moyen de langages de programmation tels que JavaScript, consultez Publication d'animations Shockwave sur le centre de support de Director, à www.macromedia.com/support/director/internet.html.

Accès aux paramètres des balises EMBED et OBJECT

Utilisez `externalParamCount()`, `externalParamName()` et `externalParamValue()` pour accéder aux paramètres des balises `EMBED` et `OBJECT`.

Listes

Les termes de cette section sont utilisés avec les listes.

Création de listes

Utilisez `[ ] (liste)`, `duplicate() (fonction de liste)` ou `list()` pour créer une liste.

Ajout d'éléments aux listes

Les termes suivants sont utilisés pour ajouter des éléments à une liste :

`[ ] (crochets d'accès)`	`addVertex`
`add`	`append`
`addVertex`	

Suppression d'éléments des listes

Les termes suivants sont utilisés pour supprimer des éléments d'une liste :

`deleteAll`	`deleteOne`
`deleteAt`	`deleteProp`

Récupération de valeurs d'une liste

Les termes suivants sont utilisés pour récupérer des valeurs d'une liste :

[ ] (crochets d'accès)	getOne()
deleteProp	getPos()
deleteProp	getProp()
getLast()	getPropAt()

Obtention d'informations sur les listes

Les termes suivants sont utilisés pour obtenir des informations sur des listes :

count()	max()
findPos	min
findPosNear	param()
ilk()	paramCount()
listP()	

Définition des valeurs d'une liste

Les termes suivants sont utilisés pour définir les valeurs d'une liste :

[ ] (crochets d'accès)	setAt
setaProp	setProp

Mémoire

Les termes de cette section sont utilisés pour déterminer la mémoire exigée et contrôler le moment auquel les acteurs sont chargés, ou purgés, de la mémoire.

Evénements d'attente

Utilisez le gestionnaire d'événement `on idle` avec le code Lingo qui doit être exécuté lorsque l'animation est en période d'inactivité.

Chargement en période d'inactivité

Les termes suivants sont utilisés pour contrôler le chargement des acteurs en période d'inactivité :

cancelIdleLoad	idleLoadPeriod
finishIdleLoad	idleLoadTag
idleHandlerPeriod	idleReadChunkSize
idleLoadDone()	netThrottleTicks

Préchargement et interrogation des médias

Les termes suivants sont utilisés pour charger des médias en mémoire et vérifier s'ils sont disponibles :

frameReady()	preloadNetThing()
loaded	preLoadMember
mediaReady	preLoadMovie
preLoad (commande)	preLoadRAM
preLoad (propriété d'acteur)	purgePriority
preLoadBuffer member	unLoad
preLoadEventAbort	unLoadMember
preLoadMode	unloadMovie

Mémoire disponible

Les termes suivants sont utilisés pour vérifier la quantité de mémoire disponible :

freeBlock()	movieFileFreeSize
freeBytes()	movieFileSize
memorySize	

Exigences en mémoire

Utilisez `ramNeeded()` et `size` pour déterminer la quantité de mémoire qu'exige un acteur ou une série d'images.

Menus

Consultez Menus dans la section Eléments d'interface.

Messages, fenêtre

Les termes suivants sont utilisés dans la fenêtre Messages :

put	traceLoad
showXlib	traceLogFile
trace	appMinimize

Moniteur

Utilisez `colorDepth`, `deskTopRectList` et `switchColorDepth` pour vérifier et contrôler le moniteur.

Navigation

Les termes suivants sont utilisés pour passer à différents endroits de l'animation :

delay	goToFrame
go	gotoNetMovie
go loop	gotoNetPage
go next	play
go previous	play done

Nombres aléatoires

Utilisez `random()` et `randomSeed` pour créer des nombres aléatoires.

Opérateurs

Les termes de cette section représentent les opérateurs offerts par Lingo.

Opérateurs mathématiques

Les termes suivants sont utilisés dans les instructions mathématiques :

* (multiplication)	<> (différent de)
/ (division)	> (supérieur à)
+ (addition)	> = (supérieur ou égal à)
- (signe moins)	< (inférieur à)
= (signal égal à)	<= (inférieur ou égal à)

Opérateurs de comparaison

Utilisez `and`, `not` et `or` pour comparer des expressions.

Ordinateur et système d'exploitation

Les termes suivants sont utilisés pour vérifier et contrôler les fonctions de l'ordinateur :

beep	freeBlock()
beepOn	freeBytes()
cpuHogTicks	maxInteger
emulateMultiButtonMouse	multiSound
floatPrecision	romanLingo

Contrôle du système d'exploitation

Utilisez `restart` et `shutDown` pour contrôler le système d'exploitation.

Palettes et couleurs

Les termes suivants sont utilisés pour vérifier et définir les palettes des animations et des acteurs :

color()	paletteMapping
depth	puppetPalette
palette	rgb()

Points et rectangles

Les termes suivants sont utilisés pour vérifier et définir les points et les rectangles.

inflate	quad
inside()	rect (caméra)
intersect()	rect (image-objet)
map()	sourceRect
offset() (fonction de rectangle)	union()
point()	

Pour plus d'informations sur les termes Lingo contrôlant le rectangle de délimitation des images-objets, consultez Dimensions des images-objets.

Projections

Les termes suivants sont utilisés avec les projections :

alertHook	platform
environment	runMode
editShortCutsEnabled	

Scénario

Les termes suivants sont utilisés avec le scénario.

Propriétés du scénario

Utilisez `lastFrame`, `score` et `scoreSelection` pour travailler avec le scénario.

Création du scénario

Les termes suivants sont utilisés pour créer du contenu dans le scénario à partir de Lingo :

beginRecording	scoreSelection
clearFrame	scriptNum
deleteFrame	scriptType
duplicateFrame	tweened
endRecording	updateFrame

insertFrame	updateLock
scoreColor	

Scène

Les termes suivants sont utilisés pour contrôler la scène et déterminer sa taille et son emplacement :

centerStage	stageColor
fixStageSize	stageLeft
picture (propriété de fenêtre)	stageRight
stage	stageTop
stageBottom	updateStage

Scripts parents

Les termes suivants sont utilisés avec les scripts parents et les objets enfants :

actorList	property
ancestor	on stepFrame
new()	handler()
handlers()	rawNew()

Serveur multiutilisateur

Les utilisateurs de Director MX devraient utiliser Macromedia Flash Communication Server MX pour les opérations de communication entre les animations Director et les serveurs d'applications. Pour plus d'informations sur l'utilisation de Flash Communication Server MX, consultez Utilisation de Flash Communication Server MX dans le mode d'emploi de Director.

Shockwave Audio

Les termes suivants sont utilisés pour vérifier, charger en flux continu et lire les sons Shockwave Audio :

bitRate	play member
bitsPerSample	preLoadBuffer member
copyrightInfo	preLoadTime
duration	sampleRate
getError()	soundChannel (SWA)
getErrorString()	state (Flash, SWA)
numChannels	stop member
pause (lecture d'animation)	streamName
percentPlayed	URL
percentStreamed	volume (propriété d'acteur)

Son

Les termes de cette section sont utilisés pour lire les sons.

Informations sur les sons

Les termes suivants sont utilisés pour obtenir des informations sur un son :

channelCount	soundEnabled
sound	volume (propriété d'image-objet)
soundBusy()	isBusy()
sampleCount	status

Lecture des sons

Les termes suivants sont utilisés pour contrôler la lecture du son :

puppetSound	sound fadeOut
sound close	sound playFile
sound fadeIn	sound stop
breakLoop()	elapsedTime
endTime	fadeIn()
fadeOut()	fadeTo()
getPlaylist()	setPlaylist()
loopCount	loopEndTime
loopStartTime	loopsRemaining
member (propriété audio)	pan (propriété audio)
pause() (lecture audio)	playNext()
queue()	rewind()
stop() (audio)	play() (audio)

Son RealMedia

Pour plus d'informations, consultez Vidéo numérique.

Synchronisation des médias

Les termes suivants sont utilisés pour synchroniser l'animation et le son :

cuePointNames	on cuePassed
cuePointTimes	isPastCuePoint()
mostRecentCuePoint	

Temps

Les termes de cette section sont utilisés avec le temps.

Date et heure actuelles

Les termes suivants sont utilisés pour déterminer la date et l'heure actuelles :

abbr, abbrev, abbreviated	short
date() (horloge du système)	systemDate
long	

Mesure du temps

Les termes suivants sont utilisés pour mesurer le temps dans une animation :

framesToHMS()	ticks
HMStoFrames()	time()
milliseconds	timer
startTimer	

Temporisations

Les termes suivants sont utilisés avec les temporisations :

timeoutKeyDown	timeoutMouse
timeoutLapsed	timeoutPlay
timeoutLength	timeoutScript
name (propriété de temporisation)	period
persistent	target
time()	timeout()
timeoutHandler	timeoutList

Texte

Les termes de cette section sont utilisés avec le texte, les chaînes et les champs.

Manipulation des chaînes

Les termes suivants sont utilisés pour manipuler les chaînes :

& (opérateur de concaténation)	put...before
&& (opérateur de concaténation)	put...into
delete	string()
hilite (propriété d'acteur)	stringP()
put...after	text

Expressions de sous-chaînes

Les termes suivants sont utilisés pour identifier les sous-chaînes de texte :

char...of	number (mots)
chars()	offset() (fonction de chaîne)
contains	paragraph
EMPTY	ref
item...of	selection (propriété d'acteur texte)
itemDelimiter	selectedText
last()	selEnd (champs uniquement)
length()	selStart (champs uniquement)
line...of	string()
number (caractères)	stringP()
number (éléments)	value()
number (lignes)	word...of

Texte modifiable

Utilisez la propriété `editable` pour spécifier si le texte est modifiable ou non.

Polices Shockwave

Les termes suivants sont utilisés pour inclure les polices Shockwave avec le texte téléchargé :

recordFont	bitmapSizes
originalFont	characterSet

Formatage des caractères

Les termes suivants sont utilisés pour formater le texte :

backColor	fontf
bgColor	fontSize
charSpacing	fontStyle
color()	foreColor
dropShadow	

Formatage des paragraphes

Les termes suivants sont utilisés pour formater les paragraphes :

alignment	rightIndent
bottomSpacing	tabCount
firstIndent	tabs
fixedLineSpace	top (3D)

leftIndent	wordWrap
margin	

Propriétés des acteurs texte

Les termes suivants sont utilisés avec le contenu entier d'un acteur texte :

antiAlias	kerning
antiAliasThreshold	kerningThreshold
autoTab	picture (propriété d'acteur)
HTML	RTF

Les termes Lingo utilisés avec les expressions de sous-chaînes peuvent également être utilisés avec le texte des acteurs texte.

Position du pointeur de la souris dans le texte

Les termes suivants sont utilisés pour détecter l'endroit du texte auquel se trouve le pointeur de la souris :

pointInHyperlink()	pointToParagraph()
pointToChar()	pointToWord()
pointToItem()	

Zones de texte des acteurs champ

Les termes suivants sont utilisés pour créer la zone de texte d'un acteur champ :

border	lineHeight() (fonction)
boxType	lineHeight (propriété d'acteur)
lineCount	pageHeight

Texte défilant

Les termes suivants sont utilisés avec le texte défilant :

linePosToLocV()	scrollByLine
locToCharPos()	scrollByPage
locVToLinePos()	scrollTop

Constantes

Les termes suivants sont utilisés avec les constantes :

BACKSPACE	RETURN (constante)
EMPTY	VOID
ENTER	

Touches

Les termes Lingo de cette section sont liés à l'utilisation du clavier.

Identification des touches

Les termes suivants sont utilisés pour identifier les touches :

charToNum()	keyPressed()
commandDown	mouseChar
controlDown	numToChar()
key()	optionDown
keyCode()	shiftDown

Interaction avec le clavier

Utilisez keyPressed(), lastEvent() et lastKey pour détecter ce que l'utilisateur tape avec le clavier.

Evénements clavier

Les termes suivants sont utilisés pour définir les gestionnaires qui répondent à l'enfoncement des touches du clavier :

on keyDown	keyDownScript
on keyUp	keyUpScript
flushInputEvents	

Transitions

Les termes suivants sont utilisés avec les transitions :

changeArea	puppetTransition
chunkSize	transitionType
duration	

Types de données

Les termes suivants sont utilisés pour spécifier des types de données :

# (symbole)	string()
float()	stringP()
floatP()	symbol()
integer()	symbolP()
integerP()	VOID
objectP()	voidP()

Variables

Les termes de cette section sont utilisés pour créer et modifier des variables :

Création de variables

Les termes suivants sont utilisés pour créer des variables :

= (signal égal à)	property
global	

Test et modification des variables

Les termes suivants sont utilisés pour vérifier et modifier les valeurs affectées aux variables :

= (signal égal à)	put
clearGlobals	set...to, set...=
globals	showGlobals
ilk()	showLocals

Vidéo numérique

Les termes suivants sont utilisés avec la vidéo numérique AVI et QuickTime :

controller	trackNextSampleTime
digitalVideoTimeScale	trackPreviousKeyTime
digitalVideoType	trackPreviousSampleTime
directToStage	trackStartTime (propriété d'image-objet)
duration	trackStartTime (propriété d'acteur)
frameRate	trackStopTime (propriété d'image-objet)
loop (propriété d'acteur)	trackStopTime (propriété d'acteur)
movieRate	trackText
movieTime	trackType (propriété d'acteur)
pausedAtStart (Flash, vidéo numérique)	trackType (propriété d'image-objet)
quickTimeVersion()	trackCount (propriété d'acteur)
timeScale	trackCount (propriété d'image-objet)
trackEnabled	video (QuickTime, AVI)
trackNextKeyTime	videoForWindowsPresent

QuickTime

Les termes suivants sont utilisés avec QuickTime :

enableHotSpot	nodeType
fieldOfView	nudge
getHotSpotRect()	pan (propriété QTVR)

hotSpotExitCallback	ptToHotSpotID()
hotSpotEnterCallback	quickTimeVersion()
invertMask	rotation
isVRMovie	scale
loopBounds	swing()
mask	staticQuality
motionQuality	tilt
mouseLevel	translation
node	triggerCallback
nodeEnterCallback	warpMode
nodeExitCallback	

Vidéo RealMedia

Les termes suivants sont utilisés avec la vidéo RealMedia :

audio (RealMedia)	play
currentTime (RealMedia)	realPlayerNativeAudio()
displayRealLogo	realPlayerPromptToInstall()
duration (RealMedia)	realPlayerVersion()
image (RealMedia)	seek
lastError	soundChannel (RealMedia)
mediaStatus	state (RealMedia)
password	stop (RealMedia)
pause (RealMedia)	userName (RealMedia)
pausedAtStart (RealMedia)	video (RealMedia)
percentBuffered	

XML

Les termes Lingo suivants sont utilisés pour l'analyse XML dans Director.

attributeName	ignoreWhiteSpace()
attributeValue	makeList()
child (XML)	makeSubList()
count()	name (propriété XML)
doneParsing()	parseString()
getError() (XML)	parseURL()

Xtras

Les termes suivants sont utilisés avec les Xtras :

movieXtraList	xtra
name (propriété système)	xtraList
number of xtras	xtras

CHAPITRE 2
Lingo 3D par fonction

Ce chapitre contient la liste des différentes fonctions 3D de Macromedia Director MX et des éléments Lingo qui peuvent être utilisés pour les implémenter.

Animation

Les termes suivants sont utilisés avec l'animation 3D. Consultez également les listes de termes utilisés avec les modificateurs de lecteur de segments et d'images-clés.

`animationEnabled`	`pause() (3D)`
`autoblend`	`play() (3D)`
`blendTime`	`playing (3D)`
`cloneMotionFromCastmember`	`playlist`
`count`	`playNext() (3D)`
`currentLoopState`	`playRate`
`currentTime (3D)`	`positionReset`
`deleteMotion`	`queue() (3D)`
`lockTranslation`	`removeLast()`
`motion`	`rotationReset`
`name`	`type (mouvement)`
`newMotion()`	

Anti-aliasing

Les termes suivants sont utilisés avec l'anti-aliasing :

`antiAliasingEnabled`	`antiAliasingSupported`

Brouillard

Les termes suivants sont utilisés avec le brouillard :

`color (brouillard)`	`far (brouillard)`
`decayMode`	`fog`
`enabled (brouillard)`	`near (brouillard)`

Calculs vectoriels

Les termes suivants sont utilisés pour effectuer des opérations mathématiques :

angleBetween	getNormalized
axisAngle	magnitude
cross	normalize
crossProduct()	randomVector
distanceTo()	vector()
dot()	x (propriété de vecteur)
dotProduct()	y (propriété de vecteur)
duplicate	z (propriété de vecteur)

Caméras

Les termes suivants sont utilisés pour contrôler les caméras et leurs propriétés :

addCamera	orthoHeight
addToWorld	pointAt
autoCameraPosition	pointAtOrientation
boundingSphere	position (transformation)
camera	projection
cameraCount()	projectionAngle
cameraPosition	rect (caméra)
cameraRotation	removeFromWorld
clone	rootNode
cloneDeep	rotate
count	scale (transformation)
deleteCamera	transform (propriété)
fieldOfView (3D)	translate
hither	userData
isInWorld()	worldPosition
name	yon
newCamera	

Création et suppression d'objets

Les termes suivants sont utilisés pour créer et supprimer des objets :

add (texture 3D)	deleteShader
addBackdrop	deleteTexture
addModifier	duplicate
addOverlay	insertBackdrop
addToWorld	insertOverlay
camera	newLight
child	newMesh
clone	newModel
cloneDeep	newModelResource
cloneModelFromCastmember	newMotion()
cloneMotionFromCastmember	newShader
deleteCamera	newTexture
deleteGroup	removeModifier
deleteLight	removeBackdrop
deleteModel	removeFromWorld
deleteModelResource	removeOverlay
deleteMotion	

Détection des collisions

Les termes suivants sont utilisés pour détecter et répondre aux collisions entre modèles :

collision (modificateur)	pointOfContact
collisionData	registerForEvent()
collisionNormal	registerScript()
enabled (collision)	resolve
immovable	resolveA
mode (collision)	resolveB
modelA	setCollisionCallback()
modelB	

Divers

clearAtRender	resetWorld
clearValue	revertToWorldDefaults
directToStage	sendEvent
loadFile()	setCollisionCallback()
registerForEvent()	unregisterAllEvents
registerScript()	revertToWorldDefaults

Fonds et recouvrements

Les termes suivants sont utilisés pour manipuler les fonds et les recouvrements des acteurs 3D :

addBackdrop	regPoint (3D)
addOverlay	removeBackdrop
blend (3D)	removeOverlay
count	rotation (fond et recouvrement)
insertBackdrop	scale (fond et recouvrement)
insertOverlay	source
loc (fond et recouvrement)	

Groupes

Les termes suivants sont utilisés avec les groupes :

addChild	newGroup
addToWorld	pointAt
boundingSphere	pointAtOrientation
child	position (transformation)
clone	removeFromWorld
cloneDeep	rotate
count	scale (transformation)
deleteGroup	transform (propriété)
group	translate
isInWorld()	userData
name	worldPosition

Images-objets (3D)

Les termes suivants sont utilisés pour contrôler les propriétés des images-objets 3D :

rect (caméra)	registerForEvent()

Lecture en flux continu

Les termes suivants sont utilisés pour contrôler la lecture en flux continu des acteurs 3D :

bytesStreamed (3D)	state (3D)
preLoad (3D)	streamSize (3D)

Lumières

Les termes suivants sont utilisés pour contrôler les lumières et leurs propriétés :

addToWorld	pointAt
ambientColor	pointAtOrientation
attenuation	position (transformation)
boundingSphere	removeFromWorld
color (lumière)	rotate
count	scale (transformation)
clone	specular (lumière)
cloneDeep	spotAngle
deleteLight	spotDecay
directionalColor	transform (propriété)
directionalPreset	translate
isInWorld()	type (lumière)
light	userData
name	worldPosition
newLight	

Matériaux

Les termes suivants sont utilisés avec les matériaux :

ambient	renderStyle
blend (3D)	shadowPercentage
blendConstant	shaderList
blendConstantList	shadowPercentage
blendFunction	shadowStrength
blendFunctionList	silhouettes
blendSource	specular (matériau)
blendSourceList	specularColor
count	specularLightMap
deleteShader	style
diffuse	textureMode

diffuseColor	textureModeList
diffuseLightMap	textureRepeat
emissive	textureRepeatList
flat	textureTransform
glossMap	textureTransformList
name	transparent
newShader	type (matériau)
renderStyle	useDiffuseWithTexture
region	wrapTransformList
reflectivity	

Matériau #engraver

Les termes suivants sont utilisés avec le matériau #engraver :

density	rotation (matériau de gravure)
brightness	

Matériau #newsprint

Les termes suivants sont utilisés avec le matériau #newsprint :

density	brightness

Matériau #painter

Les termes suivants sont utilisés avec le matériau #painter :

colorSteps	shadowPercentage
highlightPercentage	shadowStrength
highlightStrength	style

Modèles

Les termes suivants sont utilisés avec les modèles 3D :

addToWorld	position (transformation)
boundingSphere	removeFromWorld
clone	renderStyle
cloneDeep	resource
cloneModelFromCastmember	rotate
count	scale (transformation)
deleteModel	shadowPercentage

isInWorld()	shaderList
model	transform (propriété)
modifier	translate
name	userData
newModel	visibility
pointAt	worldPosition
pointAtOrientation	

Modificateurs

Les termes suivants sont pratiques pour appliquer des modificateurs aux modèles et ressources de modèles. Consultez le nom des différents modificateurs pour afficher une liste des termes correspondants.

addModifier	modifiers
count	removeModifier
modifier	

Modificateur #inker

Les termes suivants sont utilisés pour contrôler le modificateur #inker :

boundary	lineColor
creaseAngle	lineOffset
creases	silhouettes
inker (modificateur)	useLineOffset

Modificateur #toon

Les termes suivants sont utilisés pour contrôler le modificateur #toon :

boundary	lineOffset
colorSteps	shadowPercentage
creaseAngle	shadowStrength
creases	silhouettes
highlightPercentage	style
highlightStrength	toon (modificateur)
lineColor	useLineOffset

Modificateur de déformation de maille

Les termes suivants sont utilisés pour contrôler le modificateur de déformation de maille :

add (texture 3D)	normalList
face	textureCoordinateList

mesh (propriété)	textureLayer
meshDeform (modificateur)	vertexList (déformation de maille)
neighbor	

Modificateur de fractionnement de surface

Les termes suivants sont utilisés pour contrôler la fonction de fractionnement de surface :

depth (3D)	sds (modificateur)
enabled (sds)	subdivision
error	tension

Modificateur de lecteur d'images-clés

Les termes suivants sont utilisés pour contrôler le modificateur de lecteur d'images-clés :

autoblend	playing (3D)
blendFactor	playlist
blendTime	playNext() (3D)
count	playRate
currentLoopState	positionReset
currentTime (3D)	queue() (3D)
keyframePlayer (modificateur)	removeLast()
lockTranslation	rootLock
pause() (3D)	rotationReset
play() (3D)	update

Modificateur de lecteur de segments

Les termes suivants sont utilisés pour contrôler le modificateur de lecteur de segments :

autoblend	play() (3D)
blendTime	playing (3D)
bonesPlayer (modificateur)	playlist
count	playNext() (3D)
currentTime (3D)	playRate
getBoneID	queue() (3D)
currentLoopState	removeLast()
getWorldTransform()	rootLock
lockTranslation	rotationReset
positionReset	transform (propriété)
pause() (3D)	

Modificateur de niveau de détail

Les termes suivants sont utilisés pour contrôler la fonction de niveau de détail :

auto	level
bias	lod (modificateur)

Nœuds

Les termes suivants sont utilisés pour gérer les nœuds. Un nœud est un objet de l'univers, tel que les lumières, caméras, modèles et groupes.

addToWorld	isInWorld()
clone	name
cloneDeep	removeFromWorld
count	userData

Nœuds parents et enfants

Les termes suivants sont utilisés pour contrôler les relations parents-enfants entre modèles :

addChild	count
child	parent

Prélèvement

Pour plus d'informations, consultez Sélection de modèles.

Primitives

Les sections suivantes contiennent les termes utilisés pour chaque type de primitive.

Utilisez la propriété `primitives` pour déterminer les types de primitives supportés par le moteur de rendu 3D courant.

Boîtes

Les termes suivants sont utilisés pour contrôler les propriétés des boîtes 3D :

back	length (3D)
bottom (3D)	lengthVertices
front	right (3D)
height (3D)	top (3D)
heightVertices	width (3D)
left (3D)	widthVertices

Cylindres

Les termes suivants sont utilisés pour contrôler les propriétés des cylindres 3D :

bottomCap	resolution
bottomRadius	state (3D)
endAngle	topCap
height (3D)	topRadius
numSegments	

Mailles

Les termes suivants sont utilisés pour contrôler les propriétés des mailles 3D :

build()	normalList
colorList	shadowPercentage
count	textureCoordinateList
face	textureCoordinates
generateNormals()	vertexList (déformation de maille)
newMesh	

Systèmes de particules

Les termes suivants sont utilisés pour contrôler les propriétés des systèmes de particules 3D :

angle	minSpeed
blendRange	mode (émetteur)
colorRange	numParticles
direction	path
distribution	pathStrength
drag	region
gravity	sizeRange (avec début et fin)
lifetime	texture
loop (émetteur)	tweenMode
maxSpeed	wind

Plans

Les termes suivants sont utilisés pour contrôler les propriétés des plans 3D :

length (3D)	width (3D)
lengthVertices	widthVertices

Sphères

Les termes suivants sont utilisés pour contrôler les propriétés des sphères 3D :

endAngle	resolution
radius	state (3D)

Propriétés système et d'animation

Les termes suivants sont utilisés pour déterminer les capacités de l'ordinateur de lecture :

active3dRenderer	getRendererServices()
colorBufferDepth	preferred3DRenderer
depthBufferDepth	renderer
getHardwareInfo()	rendererDeviceList

Ressources de modèle

Les termes suivants sont utilisés avec les ressources de modèle 3D :

count	newModelResource
deleteModelResource	resolution
modelResource	resource
name	type (ressource de modèle)

Sélection de modèles

Les termes suivants sont utilisés pour permettre la sélection des différents modèles d'un acteur 3D et la réponse aux clics de la souris. On utilise également le terme ***prélèvement***.

modelsUnderLoc	spriteSpaceToWorldSpace
modelsUnderRay	worldSpaceToSpriteSpace
modelUnderLoc	

Systèmes de particules

Pour plus d'informations, consultez Primitives.

Texte 3D

Les termes suivants sont utilisés pour contrôler l'apparence du texte 3D :

autoCameraPosition	displayMode
bevelDepth	extrude3D
bevelType	smoothness
displayFace	tunnelDepth

Textures

Les termes suivants sont utilisés avec les textures :

compressed	newTexture
count	quality (3D)
deleteTexture	renderFormat
height (3D)	texture
member	textureRenderFormat
name	textureType
nearFiltering	type (texture)

Transformations

Les termes suivants sont utilisés avec les transformations :

duplicate	preRotate
getWorldTransform()	preScale()
identity()	preTranslate()
interpolate()	rotate
interpolateTo()	rotation (transformation)
inverse()	scale (transformation)
invert()	transform (propriété)
multiply()	translate
pointAt	worldPosition
pointAtOrientation	xAxis
position (transformation)	yAxis
preMultiply	zAxis

CHAPITRE 3
Dictionnaire Lingo

Ce dictionnaire présente la syntaxe et l'usage des divers éléments Lingo de Macromedia Director MX. Les opérateurs non alphabétiques apparaissent en premier, suivis de tous les autres opérateurs classés par ordre alphabétique.

Les entrées de ce dictionnaire correspondent à celles de l'aide en ligne de Director. Vous pouvez utiliser les exemples des scripts en copiant le texte de l'aide en ligne et en le collant dans la fenêtre Script.

(symbole)

Syntaxe

#nomDeSymbole

Description

Opérateur de symbole ; définit un symbole, 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 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

L'instruction suivante affecte le symbole `#Lecture` à la variable nommée *`état`* :

```
état = #Lecture
```

Voir aussi

`ilk()`, `string()`, `symbol()`, `symbolP()`

. (opérateur point)

Syntaxe

```
référenceDobjet.propriétéDobjet
expressionTest.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.

Exemples

L'instruction suivante indique l'acteur courant contenu dans l'image-objet de la piste 10 :

```
put sprite(10).member
```

Pour utiliser une autre syntaxe et appeler une fonction, utilisez la forme :

```
monObjetCouleur = color(#rgb, 124, 22, 233)
put monObjetCouleur.ilk()
-- #color
```

- (signe moins)

Syntaxe

```
(Négation) : -expression
```

Description

Opérateur mathématique ; inverse le signe de la valeur de l'*`expression`*.

Cet opérateur arithmétique a un niveau de priorité de 5.

Syntaxe

```
(Soustraction) : expression1 - expression2
```

Description

Opérateur mathématique ; effectue la soustraction arithmétique de deux expressions numériques, soustrayant *`expression2`* de *`expression1`*. Si les deux expressions sont des nombres entiers, la différence est 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.

Cet opérateur arithmétique a un niveau de priorité de 3.

Exemples

(Négation) : L'instruction suivante inverse le signe de l'expression 2 + 3 :

```
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 :

```
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 :

```
put 3.25 - 1.5
```

Le résultat est 1,75, qui est un nombre à virgule flottante.

-- (séparateur de commentaires)

Syntaxe

```
-- 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.

Le lecteur Director pour Java accepte le Lingo utilisant ce séparateur, mais les commentaires ne figurent pas dans le code Java final.

Exemple

Le gestionnaire suivant utilise un double trait d'union pour transformer les deuxième, quatrième et sixième lignes en commentaire :

```
on razCouleurs
  -- Ce gestionnaire réinitialise les couleurs de l'image-objet.
  sprite(1).forecolor = 35
  -- rouge vif
  sprite(1).backcolor = 36
  -- bleu clair
end
```

& (opérateur de concaténation)

Syntaxe

```
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. Le résultat est une chaîne.

Cet opérateur de chaîne a un niveau de priorité de 2.

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 :

```
open window (the applicationPath & "Animation")
```

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.

Exemples

L'instruction suivante concatène les chaînes « abra » et « cadabra » et affiche la chaîne résultante dans la fenêtre Messages :

```
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 :

```
member("Prix").text = prix & "euros"
```

&& (opérateur de concaténation)

Syntaxe

`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. Le résultat est une chaîne.

Cet opérateur de chaîne a un niveau de priorité de 2.

Exemples

L'instruction suivante concatène les chaînes « abra » et « cadabra » et insère une espace entre les deux chaînes :

```
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 :

```
put "Date :" && the long date
```

Si la date du jour est le mardi 30 juillet 2000, le résultat est la chaîne « Date : Mardi 30 juillet 2000 ».

() (parenthèses)

Syntaxe

```
(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 :

```
open window (the applicationPath & "Animation")
```

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) :

```
put (2 + 3) * (4 + 5)
-- 45
put 2 + (3 * (4 + 5))
-- 29
put 2 + 3 * 4 + 5
-- 19
```

* (multiplication)

Syntaxe

```
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.

Exemples

L'instruction suivante multiplie le nombre entier 2 par 3 et affiche le résultat dans la fenêtre Messages :

```
put 2 * 3
```

Le résultat est 6, qui est un nombre entier.

L'instruction suivante multiplie le nombre à virgule flottante 2,0 par 3,1416 et affiche le résultat dans la fenêtre Messages :

```
put 2.0 * 3,1416
```

Le résultat est 6.2832, qui est un nombre à virgule flottante.

+ (addition)

Syntaxe

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.

Exemples

L'instruction suivante additionne les nombres entiers 2 et 3, puis affiche le résultat (le nombre entier 5) dans la fenêtre Messages :

```
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 :

```
put 2,5 + 3,25
```

+ (addition) (3D)

Syntaxe

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* au composant correspondant de *vecteur2* et renvoie un nouveau vecteur.

vecteur + *scalaire* ajoute la valeur scalaire à chaque composant du vecteur et renvoie un nouveau vecteur.

- (soustraction)

Syntaxe

```
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)

Syntaxe

```
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 supporté.

Voir aussi

dotProduct()

/ (division)

Syntaxe

```
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.

Exemples

L'instruction suivante divise le nombre entier 22 par 7, puis affiche le résultat dans la fenêtre Messages :

```
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 :

```
put 22.0/7.0
```

Le résultat est 3.1429, qui est un nombre à virgule flottante.

/ (division) (3D)

Syntaxe

vecteur / scalaire

Description

Opérateur 3D de vecteur ; divise chaque composant du vecteur par la valeur scalaire et renvoie un nouveau vecteur.

< (inférieur à)

Syntaxe

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)

Syntaxe

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)

Syntaxe

```
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)

Syntaxe

```
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)

Syntaxe

```
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)

Syntaxe

```
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 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.

[ ] (crochets d'accès)

Syntaxe

```
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 :

```
put member("Prénoms").text.line[3].word[1]
```

[ ] (liste)

Syntaxe

```
[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 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, au contraire des chaînes, ne différencient pas les majuscules des minuscules.

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
-- [#nom: "Pierre"]
```

Exemples

L'instruction suivante définit une liste en rendant la variable *`machine`* égale à la liste :

```
set machine = [#engrenages:6, #billes:3, #rampes:8]
```

Le gestionnaire suivant trie la liste uneListe, puis affiche le résultat dans la fenêtre Messages :

```
on trierListe uneListe
  uneListe.sort()
  put uneListe
end trierListe
```

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]`.

L'instruction suivante crée une liste linéaire vide :

```
set x = [ ]
```

L'instruction suivante crée une liste de propriétés vide :

```
set x = [:]
```

Voir aussi

`add`, `addVertex`, `append`, `count()`, `deleteAt`, `duplicate() (fonction de liste)`, `findPos`, `findPosNear`, `getProp()`, `getAt`, `getLast()`, `getPos()`, `ilk()`, `list()`, `max()`, `min`, `setAt`, `setaProp`, `sort`

" (chaîne)

Syntaxe

```
"
```

Description

Constante de chaîne ; lorsque utilisés avant et après une chaîne, les guillemets droits signifient que la chaîne qu'ils contiennent est une chaîne littérale, et non une variable, une valeur numérique ou un élément Lingo. 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, en l'occurrence le nom d'un acteur :

```
put member("San Francisco").loaded
```

Voir aussi

`QUOTE`

¬ (symbole de continuation)

Description

Le symbole ¬ est obsolète. Utilisez maintenant le caractère `\`. Pour plus d'informations, consultez `\ (continuation)`. Il est recommandé de remplacer ce symbole par le symbole `\` dans vos scripts antérieurs.

\ (continuation)

Syntaxe

```
première partie d'une instruction sur cette ligne \
deuxième partie de l'instruction \
troisième partie de l'instruction
```

Description

Symbole de continuation ; lorsque 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 :

```
set the memberNum of sprite monImageObjet \
to member "Ceci est un long nom d'acteur."
```

@ (chemin d'accès)

Syntaxe

```
@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 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.

Exemples

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
```

Exemples

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 un niveau en amont du dossier contenant l'animation courante.

```
@::autreDossier:fichierLié
@\\autreDossier\fichierLié
@//autreDossier/fichierLié
```

Voir aussi

`searchPath`, `fileName (propriété de distribution)`, `fileName (propriété d'acteur)`, `fileName (propriété de fenêtre)`

abbr, abbrev, abbreviated

Ces éléments sont utilisés par les fonctions `date` et `time`.

Voir aussi

`date() (horloge du système)`

abort

Syntaxe

```
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 de ce(s) gestionnaire(s). 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.

Exemple

L'instruction suivante indique à Lingo de sortir du gestionnaire courant et de tout gestionnaire qui l'a appelé si la quantité de mémoire disponible est inférieure à 50 Ko :

```
if the freeBytes < 50*1024 then abort
```

Voir aussi

`exit`, `halt`, `quit`

abs()

Syntaxe

```
abs (expressionNumérique)
```

Description

Fonction mathématique ; calcule la valeur absolue d'une expression numérique. Si *`expressionNumérique`* est un nombre entier, sa valeur absolue est également un nombre entier. Si *`expressionNumérique`* est un nombre à virgule flottante, sa valeur absolue est également un nombre à virgule flottante.

La fonction `abs` sert à plusieurs choses. 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`.

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 *`startV`* est supérieure à 30 (puisque vous n'allez pas utiliser un nombre négatif pour exprimer une distance). Le cas échéant, la couleur de premier plan de l'image-objet 6 est modifiée.

```
if (the mouseV - startV).abs > 30 then sprite(6).forecolor = 95
```

actionsEnabled

Syntaxe

```
the actionsEnabled of sprite quelleImageObjetFlash
the actionsEnabled of member quelActeurFlash
sprite quelleImageObjetFlash.actionenabled
member quelActeurFlash.actionenabled
```

Description

Propriété d'acteur et propriété d'image-objet ; contrôle si les actions d'une animation 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 à point :

```
on ToggleActions quelleImageObjet
  sprite (quelleImageObjet).actionsEnabled = not sprite
  (quelleImageObjet).actionsEnabled
end
```

Syntaxe verbose :

```
on ToggleActions quelleImageObjet
  set the actionsEnabled of sprite quelleImageObjet = not the actionsEnabled
  of sprite quelleImageObjet
end
```

activateApplication

Syntaxe

```
on activateApplication
```

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 :

```
on activateApplication
  sound(1).queue(member("sonDouverture"))
  sound(1).play()
end
```

Voir aussi

`deactivateApplication`, `activeCastLib`, `on deactivateWindow`

on activateWindow

Syntaxe

```
on activateWindow
     instruction(s)
end
```

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

Le gestionnaire suivant lit le son Hourra lorsque la fenêtre dans laquelle l'animation est exécutée devient active :

```
on activateWindow
  puppetSound 2, "Hourra"
end
```

Voir aussi

`activeWindow`, `close window`, `on deactivateWindow`, `frontWindow`, `on moveWindow`, `open`

active3dRenderer

Syntaxe

```
the active3dRenderer
```

Description

Propriété d'animation Lingo 3D ; 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`.

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 lorsque aucune des trois premières options n'est disponible.

La propriété `active3dRenderer` peut être testée mais pas définie. Vous devrez utiliser `getRendererServices().renderer` pour définir cette propriété.

Exemples

Les exemples ci-dessous indiquent les deux façons de déterminer le moteur de rendu en cours d'utilisation.

```
put the active3dRenderer
-- #openGL
put getRendererServices().renderer
-- #openGL
```

Voir aussi

`renderer`, `rendererDeviceList`, `getRendererServices()`

activeCastLib

Syntaxe

```
the activeCastLib
```

Description

Propriété système ; indique la distribution la plus récemment activée. 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 castLib`. Utilisez-la pour déterminer la distribution à laquelle la sélection fait référence.

Cette propriété peut être testée, mais pas définie.

Exemple

Les instructions suivantes affectent les acteurs sélectionnés de la distribution la plus récemment sélectionnée à la variable *acteursSélectionnés* :

```
castLibDintérêt = the activeCastLib
acteursSélectionnés = castLib(castLibDintérêt).selection
```

Une façon équivalente de rédiger cette instruction serait :

```
acteursSélectionnés = castLib(the activeCastLib).selection
```

activeWindow

Syntaxe

```
the activeWindow
```

Description

Propriété d'animation ; indique la fenêtre d'animation active. Dans le cas de l'animation principale, `activeWindow` correspond à la scène. Dans le cas d'une animation dans une fenêtre, `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 :

```
on activateWindow
   set fenêtreCliquée = (the windowlist).getPos(the activeWindow)
   set nombreDeFenêtres = (the windowlist).count
   repeat with x = 1 to nombreDeFenêtres
    if x = fenêtreCliquée then
       (the activeWindow).title = "Active"
    else
       (the windowlist[x]).title = "Inactive"
    end if
  end repeat
end
```

Voir aussi

`activeCastLib`, `windowList`

actorList

Syntaxe

```
the actorList
```

Description

Propriété d'animation ; liste d'objets enfants explicitement ajoutés à cette liste. 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 `add actorList, Lobjet`. Le gestionnaire `on stepFrame` de l'objet 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 rend cette liste vide.

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 empêcher le transfert des objets enfants de l'animation courante à la nouvelle animation, insérez l'instruction `set the actorList = [ ]` dans le gestionnaire `on prepareMovie` de la nouvelle animation.

Contrairement aux versions précédentes de Director, la propriété `actorList` est maintenant supportée dans le lecteur Director pour Java.

Exemples

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.

```
add the actorList, new(script "Balle roulante", 1, 200,200)
```

L'instruction suivante affiche le contenu d'`actorList` dans la fenêtre Messages :

```
put the actorList
```

L'instruction suivante efface les objets d'`actorList`.

```
the actorList = [ ]
```

Voir aussi

`new()`

add

Syntaxe

`listeLinéaire``.add(`*`valeur`*`)`
`add` *`listeLinéaire`*`,` *`valeur`*

Description

Commande de liste ; ajoute, pour les listes linéaires uniquement, la valeur spécifiée par *`valeur`* à la liste linéaire spécifiée par *`listeLinéaire`*. 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, utilisée dans une liste de propriétés provoque une erreur.

Remarque Veillez à ne pas confondre la commande `add` et l'opérateur + utilisé pour les additions ou l'opérateur & utilisé pour concaténer les chaînes.

Exemples

Les instructions suivantes ajoutent la valeur 2 à la liste nommée `devis`. La liste résultante est [3, 4, 1, 2].

```
devis = [3, 4, 1]
devis.add(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.

```
devis.add(2)
```

Voir aussi

`sort`

add (texture 3D)

Syntaxe

```
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
```

Exemple

L'instruction suivante crée une nouvelle couche de texture pour la première maille du modèle Oreille.

```
member("Scène").model("Oreille").meshdeform.mesh[1].\
  textureLayer.add()
```

Voir aussi

`meshDeform (modificateur)`, `textureLayer`, `textureCoordinateList`

addAt

Syntaxe

```
liste.AddAt(position, valeur)
addAt liste, position, valeur
```

Description

Commande de liste ; ajoute, pour les listes linéaires uniquement, la valeur spécifiée par *`valeur`* à la liste à la position spécifiée par *`position`*.

Cette commande, utilisée avec une liste de propriétés provoque une erreur.

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

Syntaxe

```
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. Ce fond est affiché dans l'image-objet 3D à l'*emplacementDansLimageObjet* avec la rotation indiquée. Le paramètre *emplacementDansLimageObjet* est l'emplacement 2D mesuré à partir du coin supérieur gauche de l'image-objet.

Exemples

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

Syntaxe

```
sprite(quelleImageObjet).addCamera(quelleCaméra, index)
```

Description

Commande 3D ; ajoute la caméra *quelleCaméra*, à la position *index* donnée, à la liste des caméras de l'image-objet. Si la valeur *index* est supérieure à la valeur de `cameraCount()`, la caméra est ajoutée à la fin de la liste. 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.

Exemple

L'instruction suivante insère la caméra caméraDeVol en cinquième position de la liste des caméras de l'image-objet 12 :

```
sprite(12).addCamera(member("scène").camera("caméraDeVol"), 5)
```

Voir aussi

cameraCount(), deleteCamera

addChild

Syntaxe

```
member(quelActeur).nœud(quelNœudParent).addChild(member\
  (quelActeur).nœud(quelNœudEnfant) {,#conserverLunivers?})
```

Description

Commande 3D ; ajoute le nœud *quelNœudEnfant* à la liste des enfants du nœud *quelNœudParent* et le supprime de la liste des enfants de son parent précédent. Un argument *nœud* peut être un modèle, un groupe, une caméra ou une lumière. Une opération équivalente serait de donner à la propriété `parent` de *quelNœudEnfant* la valeur *quelNœudParent*.

Le paramètre facultatif *#conserverLunivers?* a deux valeurs possibles : `#preserveWorld` ou `#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.

Exemples

L'instruction suivante ajoute le modèle Pneu à la liste des enfants du modèle Voiture.

```
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.

```
member("3D").camera("maCaméra").addChild(member("3D").model\
("Oiseau"), #preserveWorld)
```

Voir aussi

`parent`, `addToWorld`, `removeFromWorld`

addModifier

Syntaxe

```
member(quelActeur).model(quelModèle).addModifier\
  (#typeDeModificateur)
```

Description

Commande 3D de modèle ; ajoute le modificateur spécifié au modèle. Les modificateurs possibles sont les suivants :

- `#bonesPlayer`
- `#collision`
- `#inker`
- `#keyframePlayer`
- `#lod` (niveau de détail)
- `#meshDeform`
- `#sds`
- `#toon`

Il n'existe aucune valeur par défaut pour cette commande.

Pour plus d'informations, consultez les entrées des différents modificateurs.

Exemple

L'instruction suivante ajoute le modificateur `toon` au modèle Boîte.

```
member("formes").model("Boîte").addModifier(#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

Syntaxe

```
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 de la caméra. Ce recouvrement est affiché dans l'image-objet 3D à l'*emplacementDansLimageObjet* avec la rotation indiquée. Le paramètre *emplacementDansLimageObjet* est l'emplacement 2D mesuré à partir du coin supérieur gauche de l'image-objet.

Exemples

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.

```
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)
```

Voir aussi

`removeOverlay`

addProp

Syntaxe

```
liste.addProp(propriété, valeur)
addProp liste, propriété, valeur
```

Description

Commande de liste de propriétés ; ajoute, pour les listes de propriétés uniquement, la propriété spécifiée par *propriété* et la valeur spécifiée par *valeur* à la liste de propriétés spécifiée par *liste*. 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, Lingo en crée une copie. Pour 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 lorsque utilisée avec une liste linéaire.

Exemples

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 en 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`, Lingo en crée une copie :

```
devis.addProp(#dupont, 7)
```

La liste qui en résulte est [#avatar: 4, #dupont: 3, #dupont: 7, #soldes: 1].

addToWorld

Syntaxe

```
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 Univers.

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.

Exemple

L'instruction suivante ajoute le modèle gbCyl à l'univers 3D de l'acteur Scène.

```
member("Scène").model("gbCyl").addToWorld()
```

Voir aussi

`isInWorld()`, `removeFromWorld`

addVertex

Syntaxe

```
member(réfActeur).AddVertex(indexOùAjouter, pointOùAjouterSommet \
{,[ emplacementH, emplacementV ], [ emplacementH, emplacementV ]})
addVertex(member réfActeur, indexOùAjouter, pointOùAjouterSommet \
{,[emplacementH, emplacementV], [emplacementH,emplacementV]})
```

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 relativement à celle du sommet ; par conséquent, si aucune position n'est spécifiée, la poignée sera placée sans décalage horizontal ou vertical (valeur 0,0).

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 :

```
member("Archie").addVertex(2, point(25, 15))
```

Voir aussi

`vertexList`, `moveVertex()`, `deleteVertex()`, `originMode`

after

Consultez

`put...after`

alert

Syntaxe

`alert` *`message`*

Description

Commande ; déclenche un bip sonore du système et affiche une boîte de dialogue d'alerte contenant la chaîne spécifiée par *`message`* ainsi qu'un bouton OK. Cette commande est pratique pour fournir des messages d'alerte pouvant contenir un maximum de 255 caractères.

Le message doit être une chaîne. Pour inclure une variable sous forme de nombre dans le message d'alerte, utilisez la fonction `string` afin de convertir la variable en chaîne.

Exemples

L'instruction suivante crée un message d'alerte indiquant qu'aucun lecteur de CD-ROM n'est connecté :

```
alert "Aucun lecteur de CD-ROM n'est connecté."
```

L'instruction suivante crée un message d'alerte indiquant qu'un fichier est introuvable :

```
alert "Le fichier" && QUOTE & nomDuFichier & QUOTE && "est introuvable."
```

Voir aussi

`string()`, `alertHook`

alertHook

Syntaxe

```
the alertHook
```

Description

Propriété système ; spécifie un script parent contenant le gestionnaire `on alertHook`. Utilisez `alertHook` pour contrôler l'affichage de messages d'alerte concernant des erreurs de fichiers ou des erreurs de script Lingo. Si une erreur survient alors qu'un script parent est affecté à `alertHook`, Director exécute le gestionnaire `on alertHook` dans le script parent.

Bien qu'il soit possible de placer des gestionnaires `on alertHook` dans des scripts d'animation, il est vivement recommandé de placer un gestionnaire `on alertHook` dans un script parent ou de comportement afin de ne pas risquer d'appeler accidentellement le gestionnaire depuis divers emplacements, ce qui risquerait de créer une certaine confusion quant à l'emplacement de l'erreur.

Le gestionnaire `on alertHook` étant exécuté en cas d'erreur, évitez de l'utiliser pour le Lingo ne servant pas à traiter les erreurs. Par exemple, le gestionnaire `on alertHook` n'est pas l'emplacement approprié pour inclure une instruction `go to movie`.

Le gestionnaire `on 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 une de ces quatre valeurs :

- `#alert` – provoque le déclenchement du gestionnaire, par la commande `alert`.
- `#movie` – provoque le déclenchement du gestionnaire, par une erreur de fichier introuvable lors de l'exécution de la commande `go to movie`.
- `#script` – provoque le déclenchement du gestionnaire, par une erreur de script.
- `#safeplayer` – provoque le déclenchement du gestionnaire, par la vérification de la propriété `safePlayer`.

Suivant le Lingo qu'il contient, le gestionnaire `on alertHook` peut ignorer l'erreur ou la signaler d'une autre façon.

Exemple

L'instruction suivante spécifie 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, Lingo affecte les chaînes d'erreur et de message à l'acteur champ Sortie et renvoie la valeur 1.

```
on prepareMovie
  the alertHook = script "Alerte"
end

-- script parent "Alerte"
on alertHook me, err, msg
  member("Sortie").text = err && msg
  return 1
end
```

Voir aussi

`safePlayer`

alignment

Syntaxe

```
member(quelActeur).alignment
the alignment of member quelActeur
```

Description

Propriété d'acteur ; détermine l'alignement utilisé pour afficher des caractères dans l'acteur champ 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 *`quelActeur`* 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.

Exemples

L'instruction suivante donne à la variable intitulée *`alignementDeCaractère`* la valeur courante de `alignment` pour l'acteur champ Pierre :

Syntaxe à point :

```
alignementDeCaractère = member("Pierre").alignment
```

Syntaxe verbose :

```
set alignementDeCaractère = the alignment of member "Pierre"
```

Cette boucle de répétition définit successivement l'alignement de l'acteur champ Jacques sur la gauche, au centre puis à droite :

Syntaxe à point :

```
repeat with i = 1 to 3
  member("Jacques").alignment = ("left center right").word[i]
end repeat
```

Syntaxe verbose :

```
repeat with i = 1 to 3
  set the alignment of member "Jacques" to word i of "left center right"
end repeat
```

Voir aussi

`text`, `font`, `lineHeight (propriété d'acteur)`, `fontSize`, `fontStyle`, `& (opérateur de concaténation)`, `&& (opérateur de concaténation)`

allowCustomCaching

Syntaxe

```
the allowCustomCaching
```

Description

Propriété d'animation ; contiendra les informations concernant un cache privé dans les futures versions de Director.

Cette propriété est, par défaut, `TRUE` et peut être testée et définie.

Voir aussi

`allowGraphicMenu`, `allowSaveLocal`, `allowTransportControl`, `allowVolumeControl`, `allowZooming`

allowGraphicMenu

Syntaxe

```
the 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 Shockwave.

Affectez à cette propriété la valeur `FALSE` si vous préférez afficher un menu textuel plutôt que le menu contextuel graphique.

Cette propriété est, par défaut, `TRUE` et peut être testée et définie.

Exemple

```
the allowGraphicMenu = 0
```

Voir aussi

`allowSaveLocal`, `allowTransportControl`, `allowVolumeControl`, `allowZooming`

allowSaveLocal

Syntaxe

```
the 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.

Cette propriété permettra des améliorations dans de futures versions de Shockwave.

Cette propriété est, par défaut, `TRUE` et peut être testée et définie.

Voir aussi

`allowGraphicMenu`, `allowTransportControl`, `allowVolumeControl`, `allowZooming`

allowTransportControl

Syntaxe

```
the allowTransportControl
```

Description

Propriété d'animation ; cette propriété est fournie pour permettre des améliorations dans de futures versions de Shockwave.

Cette propriété est, par défaut, `TRUE` et peut être testée et définie.

Voir aussi

`allowGraphicMenu`, `allowSaveLocal`, `allowVolumeControl`, `allowZooming`

allowVolumeControl

Syntaxe

```
the 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.

L'un des contrôles de volume est activé ou désactivé selon que vous affectez respectivement `TRUE` ou `FALSE` à cette propriété.

Cette propriété est, par défaut, `TRUE` et peut être testée et définie.

Voir aussi

`allowGraphicMenu`, `allowSaveLocal`, `allowTransportControl`, `allowZooming`

allowZooming

Syntaxe

```
the allowZooming
```

Description

Propriété d'animation ; détermine si l'utilisateur peut agrandir ou zoomer sur l'animation lors de la lecture sur Shockwave et ShockMachine. La valeur par défaut est `TRUE`. Cette propriété peut être testée et définie. Affectez la valeur `FALSE` à cette propriété pour empêcher le redimensionnement de l'animation dans un navigateur ou dans ShockMachine.

Voir aussi

`allowGraphicMenu`, `allowSaveLocal`, `allowTransportControl`, `allowVolumeControl`

alphaThreshold

Syntaxe

```
member(quelActeur).alphaThreshold
the alphaThreshold of member quelActeur
```

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

Syntaxe

```
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 tout 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 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

Syntaxe

```
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

Syntaxe

```
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 le comportement et les 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`

and

Syntaxe

expressionLogique1 `et` *expressionLogique2*

Description

Opérateur logique ; détermine si *expressionLogique1* et *expressionLogique2* ont toutes les deux la valeur `TRUE` ou si l'une ou les deux expressions ont la valeur `FALSE`.

Cet opérateur logique a un niveau de priorité de 4.

Exemples

L'instruction suivante détermine si les deux expressions logiques ont la valeur `TRUE` et affiche le résultat dans la fenêtre Messages :

```
put 1 < 2 and 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.

```
put 1 < 2 and 2 < 1
-- 0
```

Voir aussi

not, or

angle

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).\
  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

angleBetween

Syntaxe

```
vecteur1.angleBetween(vecteur2)
```

Description

Méthode 3D de vecteur ; renvoie l'angle entre deux vecteurs, en degrés.

Exemple

Dans l'exemple suivant, pos1 est un vecteur sur l'axe des x et pos2 est un vecteur sur l'axe des y. L'angle entre ces deux vecteurs est de 90°. La valeur renvoyée par pos1.angleBetween(pos2) est 90.0000.

```
pos1 = vector(100, 0, 0)
pos2 = vector(0, 100, 0)
put pos1.angleBetween(pos2)
-- 90.0000
```

Voir aussi

dot(), dotProduct()

animationEnabled

Syntaxe

```
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("Scène").animationEnabled = FALSE
```

antiAlias

Syntaxe

```
member(quelActeur).antiAlias
sprite(quelleImageObjetVectorielle).antiAlias
```

Description

Propriété d'acteur ; contrôle si le rendu d'un acteur texte, Flash ou forme vectorielle 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.

```
property spriteNum

on beginsprite me
  if the colorDepth <= 8 then
    sprite(spriteNum).antiAlias = FALSE
  end if
end
```

Voir aussi

antiAliasThreshold, quality

antiAliasingEnabled

Syntaxe

```
sprite(quelleImageObjet).antiAliasingEnabled
```

Type

Propriété 3D d'image-objet.

Description

Cette propriété 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 supporte l'anti-aliasing à l'aide de la propriété `antiAliasingSupported`. Si l'anti-aliasing est supporté, 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

Syntaxe

```
sprite(quelleImageObjet).antiAliasingSupported
```

Type

Propriété 3D d'image-objet.

Description

Cette propriété indique si l'anti-aliasing est supporté 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 supporté, 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

Syntaxe

```
member(quelActeurTexte).antiAliasThreshold
```

Description

Propriété d'acteur texte ; contrôle la taille de point à 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`

append

Syntaxe

```
liste.append(valeur)
append liste, valeur
```

Description

Commande de liste ; ajoute, pour les listes linéaires uniquement, 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 approprié de cette liste.

Cette commande, utilisée avec une liste de propriétés, provoque une erreur de script.

Exemple

L'instruction suivante ajoute la valeur 2 à la fin de la liste triée 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`

applicationPath

Syntaxe

```
the applicationPath
```

Description

Propriété système ; 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. La valeur de la propriété est une chaîne.

Si vous utilisez `the applicationPath` suivi de `&` et d'un chemin de dossier en aval, encadrez l'expression entière de parenthèses de façon à ce que Lingo l'analyse comme un tout.

Ni le lecteur Director pour Java ni Shockwave ne supportent cette propriété.

Cette propriété peut être testée, mais pas définie.

Exemples

L'instruction suivante affiche le chemin du dossier contenant l'application Director.

```
put the applicationPath
--"Z:\Program Files\Macromedia\Director"
```

L'instruction suivante ouvre l'animation Champs Elysées dans une fenêtre (sur un ordinateur Windows) :

```
open window (the applicationPath & "\Avenues\Champs Elysées")
```

Voir aussi

`@ (chemin d'accès)`, `moviePath`

appMinimize

Syntaxe

```
appMinimize
```

Description

Commande ; sous Windows, `appMinimize` entraîne la réduction d'une projection en une icône dans la barre d'état. Sur Macintosh, `appMinimize` entraîne la disparition de la projection. Une fois masquée, la projection peut être rouverte depuis le menu des applications du Macintosh.

Cette commande est pratique pour les projections et animations MIAW lues sans barre de titre.

Voir aussi

`windowType`

atan()

Syntaxe

```
(nombre).atan
atan (nombre)
```

Description

Fonction mathématique ; 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.

Exemples

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.

Veuillez noter que les fonctions trigonométriques utilisent normalement les radians ; vous devrez peut-être convertir les valeurs de degrés en radians.

Le gestionnaire suivant vous permet d'effectuer les conversions de degrés en radians :

```
on degrésEnRadians valeurDeDegré
  return valeurDeDegré * PI/180
end
```

Le gestionnaire suivant 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()`

attenuation

Syntaxe

`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

Syntaxe

`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>élément 2</e2>
    <e3>élément 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

Syntaxe

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>élément 2</e2>
    <e3>élément 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 (RealMedia)

Syntaxe

`sprite(`*`quelleImageObjet`*`).audio`

`member(`*`quelActeur`*`).audio`

Description

Propriété d'acteur ou image-objet RealMedia ; permet de lire (`TRUE`) ou de mettre en sourdine (`FALSE`) l'audio du train 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 fonction `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.

Exemples

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 train RealMedia sera lue.

```
put sprite(2).audio
-- 1

put member("Real").audio
-- 1
```

Le code Lingo 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 train RealMedia ne sera pas lue en même temps que l'animation.

```
sprite(2).audio = FALSE
member("Real").audio = FALSE
```

Voir aussi

`soundChannel (RealMedia)`, `video (RealMedia)`, `sound`

auto

Syntaxe

```
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

Syntaxe

```
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 non `(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("nouveauMartiens").model("Martien3").keyframePlayer.\
  autoblend = FALSE
```

Voir aussi

`blendFactor`, `blendTime`

autoCameraPosition

Syntaxe

```
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 non (`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

Syntaxe

```
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 à point :

```
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

Syntaxe

```
member(quelActeur).autoTab
the autoTab of member quelActeur
```

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.

Cette propriété a toujours la valeur `TRUE` dans une applet créée avec la fonction d'enregistrement au format Java de Director.

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 à point :

```
member ("Commentaires").autotab = TRUE
```

Syntaxe verbose :

```
set the autoTab of member "Commentaires" to TRUE
```

axisAngle

Syntaxe

```
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 autour duquel la transformation est pivotée. 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]`.

Exemples

L'instruction suivante indique la rotation du modèle boîteAuxLettres. Le modèle pivote de 145°, 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

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).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

Syntaxe

```
member(quelActeur).backColor = numéroDeCouleur
set the backColor of member quelActeur to numéroDeCouleur
sprite(quelleImageObjet).backColor
the backColor of sprite quelleImageObjet
```

Description

Propriété d'acteur et d'image-objet ; définit la couleur d'arrière-plan de l'acteur ou de l'image-objet spécifié en fonction de la valeur de couleur affectée.

- Pour les acteurs : elle affecte l'affichage de l'acteur champ ou bouton.
- Pour les images-objets : la définition de `the 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 Lingo dure au-delà de l'image-objet courante, l'image-objet doit être asservie. La couleur d'arrière-plan ne s'applique qu'aux acteurs bitmap d'un bit et aux acteurs forme.

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.

Vous ne devriez pas appliquer cette propriété à des acteurs bitmap supérieurs à 1 bit, les résultats étant difficiles à prévoir.

Pour une animation lue sous la forme d'une applet créée avec la fonction d'enregistrement au format Java de Director, spécifiez les couleurs souhaitées pour la propriété `backColor` au moyen de l'équivalent décimal des valeurs hexadécimales sur 24 bits utilisées dans les documents HTML.

Par exemple, la valeur hexadécimale du rouge pur, FF0000, est l'équivalent de 16711680 en nombre décimaux. L'instruction suivante affecte le rouge absolu comme couleur d'arrière-plan d'un acteur :

```
set the backColor of member = 16711680
```

Cette propriété peut être testée et définie.

Remarque Il est recommandé d'utiliser la nouvelle propriété `bgColor` à la place de la propriété `backColor`.

Exemples

L'instruction suivante modifie la couleur des caractères de l'acteur 1 et lui donne la couleur 250 de la palette.

Syntaxe à point :

```
member(1).backColor = 250
```

Syntaxe verbose :

```
set the backColor of member 1 to 250
```

L'instruction suivante donne à la variable *`ancienneCouleur`* la couleur d'arrière-plan de l'image-objet 5 :

```
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 images-objets 11 et 13 et lui affecte la couleur 36 :

```
sprite(10 + random(3)).backColor = 36
```

Voir aussi

`bgColor`, `color (propriété d'image-objet et d'acteur)`

backdrop

Syntaxe

```
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 plus d'informations, 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 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

Syntaxe

```
member(quelActeurFormeVectorielle).backgroundColor
the backgroundColor of member quelActeurFormeVectorielle
```

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

```
member("Archie").backgroundColor= rgb(255,255,255)
```

Voir aussi

`bgColor`

BACKSPACE

Syntaxe

```
BACKSPACE
```

Description

Constante ; représente la touche Suppression/Retour Arrière. Cette touche est appelée Retour Arrière sur un clavier Windows et Delete 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` :

```
on keyDown
  if the key = BACKSPACE then effacerEntrée
  stopEvent
end keyDown
```

beep

Syntaxe

```
beep {nombreDeFois}
```

Description

Commande ; déclenche un bip sonore du système qui se répète autant de fois que spécifié par *nombreDeFois*. Si vous n'avez pas spécifié *nombreDeFois*, 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.
- Pour le 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.

Exemple

L'instruction suivante provoque deux bips sonores si le champ Réponse est vide :

```
if field "Réponse" = EMPTY then beep 2
```

beepOn

Syntaxe

```
the 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).

Les scripts définissant la propriété `beepOn` devraient être placés dans les scripts d'images ou d'animations.

Cette propriété peut être testée et définie.

Exemples

L'instruction suivante donne à la propriété `beepOn` la valeur `TRUE` :

```
the beepOn = TRUE
```

L'instruction suivante inverse la valeur de la propriété de `beepOn` :

```
the beepOn = not the beepOn
```

before

Consultez

`put...before`

beginRecording

Syntaxe

```
beginRecording
```

Description

Mot-clé ; lance une session de création du scénario. Vous ne pouvez procéder qu'à une seule session de mise à jour du scénario à la fois dans une animation.

Chaque mot-clé `beginRecording` doit avoir un mot-clé `endRecording` lui correspondant afin de terminer la session de création du scénario.

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`.

```
on animBalle nombreDimages
  beginRecording
    horizontal = 0
    vertical = 100
    repeat with i = 1 to nombreDimages
      go to frame i
      sprite(20).member = member "Balle"
      sprite(20).locH = horizontal
      sprite(20).locV = vertical
      sprite(20).type = 1
      sprite(20).foreColor = 255
      horizontal = horizontal + 3
      vertical = vertical + 2
      updateFrame
    end repeat
  endRecording
end
```

Voir aussi

`endRecording`, `updateFrame`, `scriptNum`, `tweened`

on beginSprite

Syntaxe

```
on beginSprite
  instruction(s)
end
```

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 dans 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 son Georges Brassens lorsque l'image-objet commence :

```
on beginSprite me
  puppetSound "Georges Brassens"
end
```

Voir aussi

`on endSprite`, `on prepareFrame`, `scriptInstanceList`

bevelDepth

Syntaxe

```
member(quelActeurTexte).bevelDepth
member(quelActeur3D).modelResource(quelleRessDeMod).\
  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("Scène").model("Slogan").resource.bevelDepth = 5
```

Voir aussi

`bevelType`, `extrude3D`, `displayMode`

bevelType

Syntaxe

```
member(quelActeurTexte).bevelType
member(quelActeur3D).modelResource(quelleRessDeMod).\
  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("Scène").model("Slogan").resource.bevelType = #miter
```

Voir aussi

bevelDepth, extrude3D, displayMode

bgColor

Syntaxe

```
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 *quelleImageObjet*, 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 point :

```
(the stage).bgColor = rgb(255, 153, 0)
```

Syntaxe verbose :

```
set the bgColor of the stage = rgb(255, 153, 0)
```

Voir aussi

color(), backColor, backgroundColor, stageColor

bias

Syntaxe

```
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`

bitAnd()

Syntaxe

```
bitAnd(entier1, entier2)
```

Description

Fonction ; convertit les deux entiers spécifiés en nombres binaires 32 bits et renvoie un nombre binaire dont les chiffres sont des 1 dans les positions dans lesquelles les deux chiffres avaient des 1 et des 0 pour toutes les autres positions. Le résultat est un nouveau nombre binaire, que Lingo affiche sous la forme d'un entier de base 10.

Entier	Nombre binaire (abrégé)
6	00110
7	00111
Résultat	
6	00110

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()`

bitmapSizes

Syntaxe

```
member(quelActeurPolice).bitmapSizes
the bitmapSizes of member quelActeurPolice
```

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 :

```
put member(11).bitmapSizes
-- [12, 14, 18]
```

Voir aussi

`recordFont`, `characterSet`, `originalFont`

bitNot()

Syntaxe

```
(entier).bitNot
bitNot(entier)
```

Description

Fonction ; convertit l'entier spécifié en nombre binaire 32 bits et inverse la valeur de chaque chiffre binaire, 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 entier de base 10.

Entier	Nombre binaire
1	00000000000000000000000000000001
Résultat	
-2	11111111111111111111111111111110

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()

Syntaxe

```
bitOr(entier1, entier2)
```

Description

Fonction ; convertit les deux entiers spécifiés en nombres binaires 32 bits et renvoie un nombre binaire dont les chiffres sont des 1 dans les positions dans lesquelles un des deux chiffres avaient des 1 et des 0 pour toutes les autres positions. Le résultat est un nouveau nombre binaire, que Lingo affiche sous la forme d'un entier de base 10.

Entier	Nombre binaire (abrégé)
5	0101
6	0110
Résultat	
7	0111

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()`

bitRate

Syntaxe

```
member(quelActeur).bitRate
the bitRate of member quelActeur
```

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 à point :

```
property spriteNum

on beginSprite me
  acteurCourant = sprite(spriteNum).member.name
  put "Le débit de l'acteur"&&memName&&"est de"&&member(acteurCourant).bitRate
end
```

Syntaxe verbose :

```
property spriteNum

on beginSprite me
  acteurCourant = sprite(spriteNum).member.name
  put "Le débit de l'acteur"&&memName&&"est de"&&member(acteurCourant).bitRate
end
```

bitsPerSample

Syntaxe

```
member(quelActeur).bitsPerSample
the bitsPerSample of member quelActeur
```

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 à point :

```
put member "Paul Robin".bitsPerSample into member "Codage"
```

Syntaxe verbose :

```
put the bitsPerSample of member "Paul Robin" into member "Codage"
```

bitXor()

Syntaxe

```
bitXor(entier1, entier2)
```

Description

Fonction ; convertit les deux entiers spécifiés en nombres binaires 32 bits et renvoie un nombre binaire dont les chiffres sont des 1 dans les positions dans lesquelles les chiffres de nombres donnés ne correspondaient pas, et des 0 pour les positions dans lesquelles les chiffres étaient les mêmes. Le résultat est un nouveau nombre binaire, que Lingo affiche sous la forme d'un entier de base 10.

Entier	Nombre binaire (abrégé)
5	0101
6	0110
Résultat	
3	0011

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()`

blend

Syntaxe

```
sprite(quelleImageObjet).blend
the blend of sprite quelleImageObjet
```

Description

Propriété d'image-objet ; définit ou détermine 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.

Les couleurs possibles dépendent des couleurs disponibles dans la palette, quel que soit le codage de couleurs du moniteur.

Le lecteur Director pour Java supporte la propriété d'image-objet `blend` pour les images-objets bitmap uniquement.

Pour assurer des résultats optimaux, utilisez l'encre Opacité avec des images possédant un codage de couleurs supérieur à 8 bits.

Exemples

L'instruction suivante affecte une valeur d'opacité de 40 pour cent à l'image-objet 3.

Syntaxe à point :

```
sprite(3).blend = 40
```

Syntaxe verbose :

```
set the blend of sprite 3 to 40
```

L'instruction suivante affiche la valeur d'opacité de l'image-objet 3 dans la fenêtre Messages :

```
put the blend of sprite 3
```

Voir aussi

`blendLevel`

blend (3D)

Syntaxe

```
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`

blendConstant

Syntaxe

```
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("Niveau2").model("Mystère").shaderList[2].\
  blendConstant = 20
```

Voir aussi

`blendConstantList`, `blendFunction`, `blendFunctionList`, `blendSource`, `blendSourceList`, `useDiffuseWithTexture`, `diffuse`, `diffuseColor`

blendConstantList

Syntaxe

```
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

Syntaxe

```
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("nouveauMartiens").model("Martien3").keyframePlayer.blendFactor = 50
```

Voir aussi

`autoblend`, `keyframePlayer (modificateur)`

blendFunction

Syntaxe

```
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 ci-dessus).

#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

Syntaxe

```
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 huit positions d'index chacune. 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 fait que la texture recouvre la couche de texture inférieure. Aucune fusion n'a lieu.

#blend fait 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

Syntaxe

```
sprite(quelNuméroDimage).blendLevel
the blendLevel of sprite quelNuméroDimage
```

Description

Propriété d'image-objet ; permet de définir ou d'accéder à la valeur d'opacité courante d'une image-objet. 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`

blendRange

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod)\
  .blendRange.start
référenceDobjetDeRessourceDeModèle.blendRange.end
member(quelActeur).modelResource(quelleRessDeMod)\
  .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

Syntaxe

```
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` fait 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` fait 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

Syntaxe

```
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

Syntaxe

```
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 à 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("nouveauMartiens").model("Martien5").keyframePlayer.\
  blendTime = 1200
```

Voir aussi

`autoblend`, `blendFactor`

bone

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).\
  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`.

Pour plus d'informations, consultez `count`, `bonesPlayer (modificateur)`, `transform (propriété)` et `worldTransform`.

Voir aussi

`count`, `bonesPlayer (modificateur)`, `transform (propriété)`, `worldTransform`

bonesPlayer (modificateur)

Syntaxe

```
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` 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 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

Syntaxe

```
member(quelActeurChamp).border
the border of member quelActeurChamp
```

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 à point :

```
member("Titre").border = 10
```

Syntaxe verbose :

```
set the border of member "Titre" to 10
```

bottom

Syntaxe

```
sprite(quelleImageObjet).bottom
the bottom of sprite quelleImageObjet
```

Description

Propriété d'image-objet ; spécifie la coordonnée verticale du bord inférieur du rectangle de délimitation de l'image-objet spécifiée par *quelleImageObjet*.

Lorsqu'une animation est lue sous forme d'applet Java, sa valeur est mesurée depuis le coin supérieur gauche de l'applet.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante affecte la coordonnée verticale de la partie inférieure de l'image-objet numérotée (i + 1) à la variable *plusbas* :

Syntaxe à point :

```
set plusbas = sprite (i + 1).bottom
```

Syntaxe verbose :

```
set plusbas = the bottom of sprite (i + 1)
```

Voir aussi

`height`, `left`, `locH`, `locV`, `right`, `top`, `width`

bottom (3D)

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).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

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).\
  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

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).\
  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

Syntaxe

`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 point.

Exemple

Dans l'exemple suivant, une espace est ajoutée après le premier paragraphe de l'acteur texte Nouvelles du jour.

```
member("Nouvelles du jour").paragraph[1].bottomSpacing=20
```

Voir aussi

`top (3D)`

boundary

Syntaxe

```
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

Syntaxe

```
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

Syntaxe

```
member(quelActeur).boxDropShadow
the boxDropShadow of member quelActeur
```

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 à point :

```
member("Titre").boxDropShadow = 10
```

Syntaxe verbose :

```
set the boxDropShadow of member "Titre" to 10
```

boxType

Syntaxe

```
member(quelActeur).boxType
the boxType of member quelActeur
```

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 à point :

```
member("Editorial").boxType = #scroll
```

Syntaxe verbose :

```
set the boxType of member "Editorial" to #scroll
```

breakLoop()

Syntaxe

```
sound(numéroDePiste).breakLoop()
```

Description

Cette fonction interrompt la lecture du son mis en boucle dans la piste `numéroDePiste` et entraîne sa lecture jusqu'à la limite `posFinale`. Si aucun son n'est actuellement en boucle, cette fonction n'a pas d'effet.

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.

```
on continuerLaMusiqueDeFond
  sound(2).breakLoop()
end
```

Voir aussi

`end`, `loopCount`, `loopEndTime`, `loopsRemaining`, `loopStartTime`

brightness

Syntaxe

```
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("scène").model("gbCyl2").shader.brightness = 50
```

Voir aussi

`newShader`

broadcastProps

Syntaxe

```
member(quelActeurVecteurOuFlash).broadcastProps
the broadcastProps of member quelActeurVecteurOuFlash
```

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 à point :

```
on enterFrame
  member("Animation de navigation").broadcastProps = TRUE
  member("Animation de navigation").viewScale = 200
  member("Animation de navigation").broadcastProps = FALSE
end
```

Syntaxe verbose :

```
on enterFrame
  set the broadcastProps of member "Animation de navigation" = TRUE
  set the viewScale of member "Animation de navigation" = 200
  set the broadcastProps of member "Animation de navigation" = FALSE
end
```

browserName()

Syntaxe

```
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 spécifier un navigateur de son choix. 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 web.

Cette propriété peut être testée et définie.

Exemples

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()
```

bufferSize

Syntaxe

```
member(quelActeurFlash).bufferSize
the bufferSize of member quelActeurFlash
```

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 à point :

```
on startMovie
  member.("Démo Flash").preload = FALSE
  member.("Démo Flash").bufferSize = 65536
end
```

Syntaxe verbose :

```
on startMovie
  set the preload of member "Démo Flash" = FALSE
  set the bufferSize of member "Démo Flash" = 65536
end
```

Voir aussi

`bytesStreamed`, `preLoadRAM`, `stream`, `streamMode`

build()

Syntaxe

```
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().

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 créé 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.

```
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

buttonsEnabled

Syntaxe

```
sprite(quelleImageObjetFlash).buttonsEnabled
the buttonsEnabled of sprite quelleImageObjetFlash
member(quelActeurFlash).buttonsEnabled
the buttonsEnabled of member quelActeurFlash
```

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 à point :

```
on ToggleButtons quelleImageObjet
  sprite(quelleImObj).buttonsEnabled = not sprite(quelleImObj).buttonsEnabled
end
```

Syntaxe verbose :

```
on ToggleButtons quelleImObj
  set the buttonsEnabled of sprite quelleImObj = not the buttonsEnabled of  \
  sprite quelleImObj
end
```

Voir aussi

`actionsEnabled`

buttonStyle

Syntaxe

```
the buttonStyle
```

Description

Propriété d'animation ; détermine la réponse visuelle des boutons lorsque l'utilisateur en fait sortir le pointeur de la souris. 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.

Cette propriété peut être testée et définie dans n'importe quel type de script.

Exemples

L'instruction suivante donne à la propriété `buttonStyle` la valeur 1 :

```
the buttonStyle = 1
```

L'instruction suivante permet de mémoriser le paramètre courant de la propriété `buttonStyle` en le stockant dans la variable *`valeurDeStyleDeBouton`* :

```
valeurDeStyleDeBouton = the buttonStyle
```

Voir aussi

`checkBoxAccess`, `checkBoxType`

buttonType

Syntaxe

```
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 point :

```
member("Editorial").buttonType = #checkBox
```

Syntaxe verbose :

```
set the buttonType of member "Editorial" to #checkBox
```

bytesStreamed

Syntaxe

```
member(quelActeurFlashOuSWA).bytesStreamed
the bytesStreamed of member quelActeurFlashOuSWA
```

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 à point :

```
on téléchargerLanimation quelleAnimationFlash
  repeat while member(quelleAnimationFlash).percentStreamed < 100
    stream(member quelleAnimationFlash)
    put "Nombre d'octets transmis :" &&
  member(quelleAnimationFlash).bytesStreamed
  end repeat
end
```

Syntaxe verbose :

```
on téléchargerLanimation quelleAnimationFlash
  repeat while the percentStreamed of member quelleAnimationFlash < 100
    stream(member quelleAnimationFlash)
    put "Nombre d'octets transmis :" && the bytesStreamed of member  \
    quelleAnimationFlash
  end repeat
end
```

Voir aussi

`bufferSize`, `percentStreamed`, `stream`

bytesStreamed (3D)

Syntaxe

`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)`

cacheDocVerify()

Syntaxe

```
cacheDocVerify #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. 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 qu'elle appelle une URL.

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 comme projections. Elle n'est pas valide pour les animations Shockwave, celles-ci utilisant les paramètres réseau du navigateur web dans lequel elles sont exécutées.

```
on razCache
  valeurCourante = cacheDocVerify()
  if valeurCourante = #once then
    alert "Vérification du cache activée"
    cacheDocVerify #always
  end if
end
```

Voir aussi

`cacheSize()`, `clearCache`

cacheSize()

Syntaxe

```
cacheSize taille
cacheSize()
```

Description

Fonction et commande ; définit la taille de la mémoire cache de Director. La valeur est exprimée en kilo-octets.

La fonction `cacheSize` n'est valide que pour les animations exécutées sous Director ou comme projections. Elle n'est pas valide pour les animations Shockwave, celles-ci utilisant les paramètres réseau du navigateur web sous lequel elles sont exécutées.

Exemple

Le gestionnaire suivant vérifie si les paramètres de cache de Director sont établis à moins de 1 Mo. Le cas échéant, il affiche un message d'alerte et définit la taille de la mémoire cache à 1 Mo :

```
on vérifierCache if
  cacheSize()<1000 then
    alert "augmentation de la mémoire cache à 1 Mo"
    cacheSize 1000
  end if
end
```

Voir aussi

`cacheDocVerify()`, `clearCache`

call

Syntaxe

```
call #nomDuGestionnaire, script, {args...}
call (#nomDuGestionnaire, instanceDeScript, {args...})
```

Description

Commande ; envoie un message appelant un gestionnaire dans les scripts spécifiés, où *`nomDuGestionnaire`* représente le nom du gestionnaire devant être activé, *`script`* une référence au script ou à une liste de scripts et *`arguments`* tout paramètre facultatif devant être transmis au gestionnaire.

Si *`script`* est une instance de script, un message d'alerte est envoyé si le gestionnaire n'est pas défini dans le script ancêtre du script.

Si *script* est une liste d'instances de scripts, le message est envoyé à chaque élément de la liste tour à tour ; aucun message d'alerte n'est envoyé si le gestionnaire n'est pas défini dans le script ancêtre.

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.

Exemples

Le gestionnaire suivant envoie le message augmenterLeCompteur au premier script de comportement lié à l'image-objet 1 :

```
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
```

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 :

```
-- 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&" jambes"
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 jambes"
```

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 jambes "
-- "Homme courant sur 2 jambes "
```

callAncestor

Syntaxe

```
callAncestor nomDuGestionnaire, script, {args...}
```

Description

Commande ; envoie un message au script ancêtre d'un objet enfant, où *nomDuGestionnaire* représente le nom du gestionnaire devant être activé, *script* une référence au script ou à une liste de scripts et `arguments` tout paramètre facultatif devant être transmis au gestionnaire.

Si *script* est une instance de script, un message d'alerte est envoyé si le gestionnaire n'est pas défini dans l'ancêtre du script.

Si *script* est une liste de scripts, le message est envoyé à chaque élément de la liste tour à tour. Dans ce cas, si le gestionnaire n'est pas défini dans le script ancêtre, aucun message d'alerte n'est envoyé.

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.

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&" jambes"
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()

Syntaxe

sprite(*quelleImageObjet*).callFrame("*étiquetteFlash*")

sprite(*quelleImageObjet*).callFrame(*numéroDimageFlash*)

Définition

Commande ; utilisée pour appeler une série d'actions résidant dans une image d'une image-objet d'animation Flash. Vous pouvez spécifier l'image à appeler à l'aide d'un numéro ou d'un libellé. Cette commande transmet un message au moteur ActionScript de Flash et déclenche l'exécution des actions dans l'animation Flash.

Exemple

Cette instruction Lingo lance l'exécution des actions associées à l'image 10 de l'animation Flash dans l'image-objet 1 :

```
sprite(1).callFrame(10)
```

camera

Syntaxe

```
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 Chapitre 2, Lingo 3D par fonction, page 33.

Exemples

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()

Syntaxe

```
sprite(quelleImageObjet).cameraCount()
```

Description

Commande 3D ; renvoie le nombre d'éléments de la liste des caméras de l'image-objet.

Exemple

L'instruction suivante indique que l'image-objet 5 contient trois caméras.

```
put sprite(5).cameraCount()
-- 3
```

Voir aussi

`addCamera`, `deleteCamera`

cameraPosition

Syntaxe

```
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

Syntaxe

```
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`

cancelIdleLoad

Syntaxe

```
cancelIdleLoad baliseDeChargement
```

Description

Commande ; annule le chargement de tous les acteurs portant la balise de chargement spécifiée.

Exemple

L'instruction suivante annule le chargement des acteurs portant la balise de chargement numéro 20 :

```
cancelIdleLoad 20
```

Voir aussi

idleLoadTag

case

Syntaxe

```
case expression of
  expression1 : instruction
  expression2 :
    instructionsMultiples
    .
    .
    .
  expression3, expression4 :
    instruction(s)
  {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 en retrait immédiatement 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 d'une telle 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.

Si les valeurs de test de l'instruction `case` ne sont pas toutes des constantes entières, l'Xtra d'exportation pour Java convertit l'instruction `case` en une instruction `if...then`.

Exemples

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 (the key) of
    "A": go to frame "Pomme"
    "B", "C":
      puppetTransition 99
       go to frame "Oranges"
    otherwise 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 the rollOver of
  1: puppetSound "Clarinette"
  2: puppetSound "Tambour"
  3: puppetSound "Bongos"
end case
```

castLib

Syntaxe

`castLib` *`quelleDistribution`*

Description

Mot-clé ; indique que la distribution spécifiée par *`quelleDistribution`* est une distribution.

La distribution par défaut est la distribution numéro 1. Pour spécifier un acteur dans une distribution autre que la distribution numéro 1, définissez `castLib` pour spécifier l'autre distribution.

Exemples

L'instruction suivante affiche le numéro de la distribution Boutons dans la fenêtre Messages :

Syntaxe à point :

```
put castLib("Boutons").number
```

Syntaxe verbose :

```
put the number of castLib "Boutons"
```

L'instruction suivante affecte l'acteur 5 de la distribution 4 à l'image-objet 10 :

```
sprite(10).member = member(5, 4)
```

castLibNum

Syntaxe

```
member(quelActeur).castLibNum
the castLibNum of member quelActeur
sprite(quelleImageObjet).castLibNum
the castLibNum of sprite quelleImageObjet
```

Description

Propriété d'acteur et d'image-objet ; détermine le numéro de la distribution contenant l'acteur spécifié ou celui de la castLib associée à l'image-objet spécifiée.

Si vous modifiez la propriété d'image-objet `the castLibNum` sans modifier `the memberNum`, Director utilise l'acteur qui a le même numéro dans la nouvelle distribution. Cela est particulièrement utile pour les animations que vous utilisez comme modèles et mettez à jour en fournissant de nouvelles distributions. Si vous organisez le contenu de la distribution de manière à ce que tous les acteurs aient un numéro qui corresponde à leur rôle dans l'animation, Director insère automatiquement les nouveaux acteurs à l'emplacement correct. Pour changer l'acteur affecté à une image-objet, quelle que soit sa distribution, définissez la propriété d'acteur `member`.

Dans le cas d'un acteur, cette propriété peut être testée mais non définie. Elle peut être testée et définie dans le cas d'une image-objet.

Exemples

L'instruction suivante détermine le numéro de la distribution à laquelle l'acteur Jazz est affecté :

Syntaxe à point :

```
put member("Jazz").castLibNum
```

Syntaxe verbose :

```
put the castLibNum of member "Jazz"
```

L'instruction suivante modifie l'acteur affecté à l'image-objet 5 en remplaçant sa distribution par la distribution Mercredi :

Syntaxe à point :

```
sprite(5).castLibNum = castLib("Mercredi").number
```

Syntaxe verbose :

```
set the castLibNum of sprite 5 to the number of castLib "Mercredi"
```

castMemberList

Syntaxe

```
member(quelActeurCurseur).castmemberList
the castMemberList of member quelActeurCurseur
```

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 à point :

```
member("monCurseur").castmemberList =  \
[member 1, member 2, member 1 of castlib 2, member 2 of castlib 2]
```

Syntaxe verbose :

```
set the castmemberList of member "monCurseur" =  \
[member 1, member 2, member 1 of castlib 2, member 2 of castlib 2]
```

center

Syntaxe

```
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 `the 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 point :

```
member("Interview").center = FALSE
```

Syntaxe verbose :

```
set the center of member "Interview" to FALSE
```

Voir aussi

`crop (propriété d'acteur)`, `centerRegPoint`, `regPoint`, `scale`

centerRegPoint

Syntaxe

```
member(quelActeur).centerRegPoint
the centerRegPoint of member quelActeur
```

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 à point :

```
on beginSprite me
  if sprite(the spriteNum of me).member.centerRegPoint = TRUE then
    sprite(the spriteNum of me).member.regPoint = point(0,0)
  end if
end
```

Syntaxe verbose :

```
on beginSprite me
  if the centerRegPoint of member the memberNum of me = TRUE then
    set the regPoint of member the memberNum of me = point(0,0)
  end if
end
```

Voir aussi

`regPoint`

centerStage

Syntaxe

```
the 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`). 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.

Cette propriété peut être testée et définie.

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.

Exemples

L'instruction suivante déplace l'animation vers une image spécifique si la scène n'est pas centrée :

```
if the centerStage = FALSE then go to frame "Décalé"
```

L'instruction suivante inverse la valeur courante de la propriété `centerStage` :

```
set the centerStage to (not the centerStage)
```

Voir aussi

`fixStageSize`

changeArea

Syntaxe

```
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 point :

```
member("Vague").changeArea = TRUE
```

Syntaxe verbose :

```
set the changeArea of member "Vague" to TRUE
```

channelCount

Syntaxe

```
member(quelActeur).channelCount
the channelCount of member quelActeur
sound(numéroDePiste).channelCount
```

Description

Propriété d'acteur et de piste audio ; pour les pistes audio, détermine le nombre de pistes du son en cours ou en pause dans la piste audio spécifiée. Pour les acteurs son, détermine le nombre de pistes contenues dans l'acteur spécifié.

Elle est utile pour savoir si un son est mono ou stéréo. Cette propriété peut être testée, mais pas définie.

Exemples

L'instruction suivante détermine le nombre de pistes contenues dans l'acteur son Jazz.

Syntaxe à point :

```
put member("Jazz").channelCount
```

Syntaxe verbose :

```
put the channelCount of member "Jazz"
```

L'instruction suivante détermine le nombre de pistes contenues dans l'acteur son Jazz lu dans la piste 2.

```
put sound(2).channelCount
```

char...of

Syntaxe

```
expressionActeurTexte.char[quelCaractère]
char quelCaractère of 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 est 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.

Exemples

L'instruction suivante affiche le premier caractère de la chaîne 9,00 euros :

```
put ("9,00 euros").char[1..1]
-- "9"
```

L'instruction suivante affiche la chaîne 9,00 euros entière :

```
put ("9,00 euros").char[1..10]
-- "9,00 euros"
```

L'instruction suivante modifie les cinq premiers caractères du deuxième mot de la troisième ligne d'un acteur champ :

```
member("question").line[3].word[2].char[1..5] = "?????"
```

Voir aussi

`mouseMember`, `mouseItem`, `mouseLine`, `mouseWord`

characterSet

Syntaxe

```
member(quelActeurPolice).characterSet
the characterSet of member quelActeurPolice
```

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 étaient 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.

```
put member(11).characterSet
-- "1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
```

Voir aussi

recordFont, bitmapSizes, originalFont

charPosToLoc()

Syntaxe

```
member(quelActeur).charPosToLoc(énièmeCaractère)
charPosToLoc(member quelActeur, énièmeCaractère)
```

Description

Fonction de champ ; renvoie le point de l'acteur champ entier (et non uniquement la partie affichée sur la scène) qui est le plus proche du caractère spécifié par *énièmeCaractère*. 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.

Exemple

L'instruction suivante détermine le point auquel apparaît le cinquantième caractère de l'acteur Titre et affecte le résultat à la variable *emplacement* :

```
emplacement = charPosToLoc(member "Titre", 50)
```

chars()

Syntaxe

```
chars(expressionChaîne, premierCaractère, dernierCaractère)
```

Description

Fonction ; identifie une sous-chaîne de caractères dans *expressionChaîne*. La sous-chaîne commence au *premierCaractère* et se termine au *dernierCaractère*. 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.

Vous pourrez voir un exemple de `chars()` dans une animation en consultant l'animation Text du dossier Learning/Lingo_Examples, lui-même dans le dossier de Director.

Exemples

L'instruction suivante identifie le sixième caractère du mot Macromedia :

```
put chars ("Macromedia", 6, 6)
-- "m"
```

L'instruction suivante identifie les caractères compris entre le sixième et le dixième caractère 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)`

charSpacing

Syntaxe

`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 :

```
on monEspaceDeCaractères
  maValeurDespace = member("monTitre").word[3..5].charSpacing
  member("monTitre").word[3..5].charSpacing = (maValeurDespace + 2)
end
```

charToNum()

Syntaxe

(*`expressionChaîne`*)`.charToNum`
`charToNum(`*`expressionChaîne`*`)`

Description

Fonction ; renvoie le numéro de code ASCII correspondant au premier caractère de *`expressionChaîne`*.

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 alphanumérique.

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.

Exemples

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 :

```
on vérifierTouchePressée laTouche
  if (laTouche).charToNum = 65 then
    go "Réponse correcte"
  else
    go "Réponse incorrecte"
  end if
end
```

Voir aussi

`numToChar()`

checkBoxAccess

Syntaxe

```
the checkBoxAccess
```

Description

Propriété d'animation ; spécifie un des trois résultats possibles lorsque l'utilisateur clique sur une case à cocher ou un bouton radio créé avec les outils de bouton de la fenêtre Outils :

- `0` (valeur par défaut) – Permet à l'utilisateur d'activer et de désactiver les cases à cocher et les boutons radio.
- `1` – Permet à l'utilisateur d'activer les cases à cocher et les boutons radio, mais pas de les désactiver.
- `2` – Empêche l'utilisateur d'activer et de désactiver les cases à cocher et les boutons radio, ceux-ci ne pouvant être activés/désactivés que par des scripts.

Cette propriété peut être testée et définie.

Exemples

L'instruction suivante donne à la propriété `checkBoxAccess` la valeur 1, ce qui permet à l'utilisateur de cliquer sur les cases à cocher et les boutons radio pour les activer, mais ne permet pas leur désactivation :

```
the checkBoxAccess to 1
```

L'instruction suivante enregistre le paramètre courant de la propriété `checkBoxAccess` en incorporant sa valeur dans la variable *`ancienAccès`* :

```
ancienAccès to the checkBoxAccess
```

Voir aussi

`hilite (propriété d'acteur)`, `checkBoxType`

checkBoxType

Syntaxe

```
the checkBoxType
```

Description

Propriété d'animation ; spécifie une des trois façons d'indiquer si une case à cocher est sélectionnée ou non :

- `0` (valeur par défaut) – Crée une case à cocher standard contenant un X lorsqu'elle est activée.
- `1` – Crée une case à cocher contenant un rectangle noir lorsqu'elle est activée.
- `2` – Crée une case à cocher contenant un rectangle noir rempli lorsqu'elle est activée.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante donne à la propriété `checkBoxType` la valeur 1, qui crée un rectangle noir au sein des cases à cocher lorsque l'utilisateur clique dessus :

```
the checkBoxType to 1
```

Voir aussi

`hilite (propriété d'acteur)`, `checkBoxAccess`

checkMark

Syntaxe

```
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.

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

Syntaxe

```
member(quelActeur).model(quelNœudParent).\
  child(quelNœudEnfant)
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)

Syntaxe

`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>élément 2</e2>
    <e3>élément 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

Syntaxe

```
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 point :

```
member("Brouillard").chunkSize = 4
```

Syntaxe verbose :

```
set the chunkSize of member "Brouillard" to 4
```

clearAsObjects()

Syntaxe

```
clearAsObjects()
```

Définition

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 :

```
monTableauGlobal = newObject(#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)
```

Exemple

Cette instruction supprime de la mémoire tous les objets ActionScript créés globalement :

```
clearAsObjects()
```

Voir aussi

`newObject()`, `setCallback()`

clearAtRender

Syntaxe

```
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`

clearCache

Syntaxe

```
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).

Exemple

Le gestionnaire suivant vide la mémoire cache au lancement de l'animation :

```
on startMovie
  clearCache
end
```

Voir aussi

`cacheDocVerify()`, `cacheSize()`

clearError

Syntaxe

```
member(quelActeurFlash).clearError()
clearError (member quelActeurFlash)
```

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. Vous pouvez utiliser la fonction `getError` pour déterminer le type d'erreur survenue, puis utiliser 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 `chemin`, `linked` et `preload` d'un acteur élimine automatiquement la condition d'erreur.

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.

```
on vérifierEtatFlash
  if member("Dali").getError() = #memory then
    member("Dali").clearError()
    unloadCast
    go to frame "Artistes"
  else
    go to frame "Désolé"
  end if
end
```

Voir aussi

`state (Flash, SWA)`, `getError()`

clearFrame

Syntaxe

```
clearFrame
```

Description

Commande ; efface tout ce qui se trouve dans les pistes d'images-objets de l'image courante (pendant l'enregistrement du scénario uniquement).

Exemple

Le gestionnaire suivant supprime le contenu de chaque image avant de les modifier pendant la création du scénario :

```
on nouveauScénario
  beginRecording
  repeat with compteur = 1 to 50
    clearFrame
    the frameScript to 25
    updateFrame
  end repeat
  endRecording
end
```

Voir aussi

`beginRecording`, `endRecording`, `updateFrame`

clearGlobals

Syntaxe

```
clearGlobals
```

Description

Commande ; donne à toutes les variables globales la valeur `VOID`.

Cette commande est particulièrement utile lors de l'initialisation de variables globales ou de l'ouverture d'une nouvelle animation qui exige un nouveau jeu de variables globales.

Exemple

L'instruction suivante donne à toutes les variables globales la valeur `VOID` :

```
clearGlobals
```

clearValue

Syntaxe

```
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

Syntaxe

```
the clickLoc
```

Description

Fonction ; identifie le dernier endroit de l'écran où un clic de la souris a eu lieu.

Exemple

Le gestionnaire `on mouseDown` suivant affiche l'emplacement du dernier clic de la souris :

```
on mouseDown
  put the clickLoc
end mouseDown
```

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)
```

clickMode

Syntaxe

```
sprite(quelleImageObjetFlash).clickMode
the clickMode of sprite quelleImageObjetFlash
member(quelActeurFlash).clickMode
the clickMode of member quelActeurFlash
```

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 à point :

```
on beginSprite me
  if sprite(the spriteNum of me).directToStage = FALSE then
    sprite(the spriteNum of me).clickMode = #opaque
  else
    sprite(the spriteNum of me).clickMode = #boundingBox
  end if
end
```

Syntaxe verbose :

```
on beginSprite me
  if the directToStage of sprite the spriteNum of me = FALSE then
    set the clickMode of sprite the spriteNum of me = #opaque
  else
    set the clickMode of sprite the spriteNum of me = #boundingBox
  end if
end
```

clickOn

Syntaxe

```
the clickOn
```

Description

Fonction ; renvoie la dernière image-objet active sur laquelle l'utilisateur a cliqué. 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 factice (par exemple `"--,"`) afin de permettre sa détection par la fonction `clickOn`.

La fonction `clickOn` peut être testée dans une boucle de répétition. Cependant, cette fonction, comme la propriété `clickLoc`, ne change pas de valeur lors de l'exécution du gestionnaire. La valeur que vous obtenez est la valeur précédant le démarrage du gestionnaire.

Exemples

L'instruction suivante vérifie si l'image-objet 7 est la dernière image-objet active sur laquelle l'utilisateur a cliqué :

```
if the clickOn = 7 then 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 :

```
sprite(the clickOn).foreColor = random(255)-1
```

Voir aussi

`doubleClick`, `the mouseDown (propriété système)`, `mouseMember`, `the mouseUp (propriété système)`

clone

Syntaxe

```
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 est nommé *nomDuClone* et 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.

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

Syntaxe

```
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

Veuillez noter que `cloneDeep` utilise plus de mémoire et prend plus de temps que la commande `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

Syntaxe

```
member(quelActeur).cloneModelFromCastmember\
  (nomDeNouveauModèle, nomDeModèleSource, acteurSource)
```

Description

Commande 3D ; copie le modèle *nomDeModèleSource* de l'acteur *acteurSource*, le renomme *nomDeNouveauModèle*, et l'insère dans l'acteur *quelActeur* comme 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.

Exemple

L'instruction suivante crée une copie du modèle Pluton de l'acteur Séquence et l'insère dans l'acteur Séquence2 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("Séquence2").cloneModelFromCastmember("Planète", "Pluton", \
  member("Séquence"))
```

Voir aussi

`cloneMotionFromCastmember`, `clone`, `cloneDeep`, `loadFile()`

cloneMotionFromCastmember

Syntaxe

```
member(quelActeur).cloneMotionFromCastmember(nomDeNouveauMouvement, \
  nomDeMouvementSource, acteurSource)
```

Description

Commande 3D ; copie le mouvement *nomDeMouvementSource* de l'acteur *acteurSource*, le renomme *nomDeNouveauMouvement*, et l'insère dans l'acteur *quelActeur* comme enfant de son univers 3D.

Le chargement de l'acteur source doit être terminé pour la bonne exécution de cette commande.

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()`

closed

Syntaxe

```
member(quelActeur).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.

close window

Syntaxe

```
window(identifiantDeFenêtre).close()
close window identifiantDeFenêtre
```

Description

Commande de fenêtre ; ferme la fenêtre spécifiée par *identifiantDeFenêtre*.

- Pour spécifier une fenêtre par nom, utilisez la syntaxe `close window` *nom*, où *nom* correspond au nom d'une fenêtre. Utilisez le chemin complet.
- Pour spécifier une fenêtre par numéro dans `windowList`, utilisez la syntaxe `close window` *numéro*, où *numéro* correspond au numéro de la fenêtre dans la liste `windowList`.

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 commande sert simplement à fermer la fenêtre dans laquelle l'animation est exécutée. Pour la rouvrir rapidement, utilisez la commande `open window`. 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 commande `forget window`. Si vous utilisez la commande `forget window`, assurez-vous qu'aucun élément ne fait référence à l'animation de cette fenêtre, faute de quoi le système génère des erreurs lorsque les scripts essaient de communiquer ou d'utiliser la fenêtre supprimée.

Exemples

L'instruction suivante ferme la fenêtre Panneau, qui se trouve dans le sous-dossier Sources MIAW dans le dossier de l'animation courante :

```
window("@/Sources MIAW/Panneau").close()
```

L'instruction suivante ferme la fenêtre 5 dans la liste `windowList` :

```
window(5).close()
```

Voir aussi

`forget`, `open window`, `windowList`

on closeWindow

Syntaxe

```
on closeWindow
  instruction(s)
end
```

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 le Lingo 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 :

```
on closeWindow
  -- opérations de maintenance standard
  forget the activeWindow
end
```

closeXlib

Syntaxe

```
closeXlib quelFichier
```

Description

Commande ; ferme le fichier Xlibrary spécifié par la chaîne *`quelFichier`*. Si le fichier Xlibrary se trouve dans un dossier autre que le dossier de l'animation courante, *`quelFichier`* doit spécifier un nom de chemin. Si vous ne spécifiez pas de fichier, tous les fichiers Xlibrary ouverts sont fermés.

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 supportée dans Shockwave.

Exemples

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, showXlib

collision (modificateur)

Syntaxe

```
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

Syntaxe

```
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, 0.000, 0.000)
-- vector(-1.000, 0.000, 0.000)
```

Voir aussi

Propriétés `collisionData` : `modelA`, `modelB`, `pointOfContact`, `collisionNormal`

Méthodes `collisionData` : `resolveA`, `resolveB`, `collision (modificateur)`

collisionNormal

Syntaxe

```
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 = collisionData.collisionNormal
  nm = member("maSéquence").newModel("modèleDétincelles", nmr)
  nm.transform.position = collisionData.pointOfContact
end
```

Voir aussi

pointOfContact, modelA, modelB, resolveA, resolveB, collision (modificateur)

color()

Syntaxe

```
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 les propriétés d'acteur color et bgColor, et dans les propriétés d'image-objet color et 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.

Exemples

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`

color (brouillard)

Syntaxe

```
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)

Syntaxe

```
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`

color (propriété d'image-objet et d'acteur)

Syntaxe

```
sprite(quelNuméroDimage).color
the color of sprite quelNuméroDimage
member(quelActeur).color
```

Description

Propriété d'image-objet et d'acteur ; pour les images-objets, détermine la couleur de premier plan de l'image-objet spécifiée par *quelleImageObjet*. La définition de la propriété d'image-objet `foreColor` revient à choisir la couleur de premier plan dans la fenêtre Outils lorsque l'image-objet est sélectionnée sur la scène.

Cette propriété a le même effet que la propriété d'image-objet `foreColor`, mais la valeur de couleur renvoyée est un objet couleur, quel que soit le type défini pour cette image-objet.

Pour les acteurs texte, cette propriété détermine la couleur du texte.

Cette propriété peut être testée et définie. La propriété `color` peut être n'importe quelle valeur rgb ou hexadécimale.

Exemples

L'instruction suivante donne au texte de l'acteur 3 une couleur rouge moyenne :

```
member(3).color = rgb(255, 0, 100)
```

L'instruction suivante donne au texte de l'acteur 3 une couleur bleue moyenne :

```
member(3).color = rgb("0033FF")
```

Voir aussi

`color()`, `bgColor`, `foreColor`

colorBufferDepth

Syntaxe

```
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

Syntaxe

```
the colorDepth
```

Description

Propriété système ; paramètre le codage des couleurs du moniteur.

- 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.

Cette propriété peut être testée et définie.

Exemples

L'instruction suivante ne permet la lecture du segment Couleurs complètes que si le codage des couleurs du moniteur est de 256 couleurs :

```
if the colorDepth = 8 then play movie "Couleurs complètes"
```

Le gestionnaire suivant tente de changer le codage des couleurs et, s'il n'y arrive pas, un message d'alerte apparaît :

```
on essaiDeRéglageDesCouleurs codageSouhaité
  the colorDepth = codageSouhaité
  if the colorDepth = codageSouhaité then
    return true
  else
    alert "Veuillez changer le codage des couleurs de votre ordinateur sur" &&
  codageSouhaité &&"bits et redémarrer."
    return false
  end if
end
```

Si vous changez le codage des couleurs du moniteur de l'utilisateur, il est toujours courtois de restaurer le codage d'origine une fois l'animation terminée. Sous Windows, la commande `set the colorDepth = 0` restaure les paramètres de l'utilisateur définis dans le Panneau de configuration.

Voir aussi

`switchColorDepth`

colorList

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).\
  colorList
member(quelActeur).modelResource(quelleRessDeMod).\
  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

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).\
  colorRange.start
member(quelActeur).modelResource(quelleRessDeMod).\
  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

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).\
  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 = [ 1,2,3 ]
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

face, vertices, colorList, flat

colorSteps

Syntaxe

```
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

Syntaxe

```
the commandDown
```

Description

Fonction ; détermine si l'utilisateur appuie sur la touche Cmd du Macintosh ou Ctrl sous Windows (TRUE) ou non (FALSE).

Vous pouvez utiliser la fonction commandDown avec l'élément the 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 pendant la lecture d'une projection.

Pour la lecture d'une animation avec le lecteur Director pour Java, cette fonction renvoie `TRUE` uniquement si l'utilisateur appuie simultanément sur la touche Cmd ou Ctrl et sur une autre touche. S'il appuie uniquement sur la touche Cmd ou Ctrl, `commandDown` renvoie `FALSE`. En effet, le navigateur reçoit les touches avant l'animation et répond et intercepte les combinaisons de touches qui servent également de raccourcis clavier pour le navigateur. Par exemple, si l'utilisateur entre Ctrl+R ou Cmd+R, le navigateur recharge la page courante et l'animation ne reçoit jamais la combinaison de touches.

Exemple

Les instructions suivantes mettent une projection sur pause à chaque fois que l'utilisateur appuie sur Ctrl+A (Windows) ou Cmd+A (Macintosh). En paramétrant la propriété `keyDownScript` sur `doCommandKey`, le gestionnaire `on prepareMovie` fait que le gestionnaire `doCommandKey` est le premier gestionnaire d'événement exécuté lorsque l'utilisateur appuie sur une touche. Le gestionnaire `toucheCmd` vérifie si le raccourci clavier Ctrl+A ou Cmd+A a été utilisé et, le cas échéant, met l'animation en pause.

```
on prepareMovie
  the keyDownScript = "doCommandKey"
end

on toucheCmd
  if (the commandDown) and (the key = "a") then go to the frame
end
```

Voir aussi

`controlDown`, `key()`, `keyCode()`, `optionDown`, `shiftDown`

comments

Syntaxe

```
quelActeur.comments
the comments of quelActeur
```

Description

La propriété d'acteur suivante 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. Cette propriété peut être testée et définie. Elle 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 ».

```
member("Fond").comments = "Obtenez la permission d'utiliser ce graphique"
```

Voir aussi

`creationDate`, `modifiedBy`, `modifiedDate`

compressed

Syntaxe

```
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`

constrainH()

Syntaxe

```
constrainH (quelleImageObjet, expressionEntière)
```

Description

Fonction ; évalue *`expressionEntière`*, puis renvoie une valeur dépendant des coordonnées horizontales des bords gauche et droit de *`quelleImageObjet`*, comme suit :

- Lorsque cette valeur est située entre les coordonnées gauche et droite, elle ne change pas.
- Lorsqu'elle est inférieure à la coordonnée horizontale gauche, elle est remplacée par la coordonnée gauche.
- Lorsqu'elle est supérieure à la coordonnée horizontale droite, elle est remplacée par la coordonnée droite.

Les fonctions `constrainH` et `constrainV` n'agissent que sur un seul axe, la propriété `constraint` limitant les deux. Cette fonction ne modifie pas les propriétés de l'image-objet.

Exemples

Les instructions suivantes vérifient la fonction `constrainH` sur l'image-objet 1 lorsque ses coordonnées gauche et droite sont 40 et 60 :

```
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 :

```
set the locH of sprite 1 to constrainH(2, the mouseH)
```

Voir aussi

`constrainV()`, `constraint`, `left`, `right`

constraint

Syntaxe

```
sprite(quelleImageObjet).constraint
the constraint of sprite quelleImageObjet
```

Description

Propriété d'image-objet ; détermine si le point d'alignement de l'image-objet spécifiée par *`quelleImageObjet`* est limité au rectangle de délimitation d'une autre image-objet (`TRUE`) ou non (`FALSE`, la valeur par défaut).

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 d'image-objet `constraint` s'applique aux images-objets mobiles et aux propriétés d'image-objet `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, le point de contrôle est le point d'alignement ; 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 d'image-objet `locH` et `locV`.

Cette propriété peut être testée et définie.

Exemples

L'instruction suivante supprime une propriété d'image-objet `constraint` :

Syntaxe à point :

```
sprite(quelleImageObjet).constraint = 0
```

Syntaxe verbose :

```
set the constraint of sprite quelleImageObjet to 0
```

L'instruction suivante contraint l'image-objet (i + 1) à la limite de l'image-objet 14 :

```
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` (l'opérateur <> exécute l'opération différent de) :

```
if sprite(3).constraint <> 0 then showConstraint
```

Voir aussi

`constrainH()`, `constrainV()`, `locH`, `locV`

constrainV()

Syntaxe

```
constrainV (quelleImageObjet, expressionEntière)
```

Description

Fonction ; évalue *expressionEntière*, puis renvoie une valeur dépendant des coordonnées verticales des bords supérieur et inférieur de l'image-objet spécifiée par *quelleImageObjet* :

- Lorsque la valeur est située entre les coordonnées supérieure et inférieure, elle ne change pas.
- Lorsqu'elle est inférieure à la coordonnée supérieure de l'image-objet, la valeur est remplacée par cette dernière.
- Lorsqu'elle est supérieure à la coordonnée inférieure de l'image-objet, la valeur est remplacée par cette dernière.

Cette fonction ne modifie pas les propriétés de l'image-objet.

Exemples

Les instructions suivantes vérifient la fonction `constrainV` de l'image-objet 1 lorsque ses coordonnées supérieure et inférieure sont 40 et 60 :

```
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 :

```
set the locV of sprite 1 to constrainV(2, the mouseV)
```

Voir aussi

`bottom`, `constraint`, `top`, `constrainH()`

contains

Syntaxe

```
expressionChaîne1 contains 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 de caractères spécifique. 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 un caractère transmis est un chiffre :

```
on estNombre uneLettre
  chiffres = "1234567890"
  if chiffres contains uneLettre then
    return TRUE
  else
    return FALSE
  end if
end
```

Remarque La comparaison de chaînes ne tient pas compte des différences entre majuscules et minuscules ; les lettres « a » et « A » étant traitées de la même manière.

Voir aussi

`offset() (fonction de chaîne)`, `starts`

continue

Ce terme Lingo est obsolète. Utilisez `go to the frame +1`.

controlDown

Syntaxe

```
the controlDown
```

Description

Fonction ; détermine si l'utilisateur appuie sur la touche Ctrl (`TRUE`) ou non (`FALSE`).

Vous pouvez utiliser la fonction `controlDown` avec l'élément `the key` pour vérifier l'utilisation de combinaisons de la touche Ctrl et d'une autre touche.

Pour la lecture d'une animation avec le lecteur Director pour Java, cette fonction renvoie `TRUE` uniquement si l'utilisateur appuie simultanément sur la touche Ctrl et sur une autre touche. S'il appuie uniquement sur la touche Cmd ou Ctrl, `controlDown` renvoie `FALSE`. Le lecteur Director pour Java supporte les combinaisons de touches avec la touche Ctrl. Néanmoins, le navigateur web reçoit les touches avant l'animation et, par conséquent, intercepte et répond à des combinaisons de touches qui correspondent également à des raccourcis de clavier du navigateur.

Pour une démonstration des combinaisons de touches dans Lingo, consultez l'animation Clavier et Lingo dans le système d'aide de Director.

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 `(the key)` identifie la touche enfoncée en plus de la touche Ctrl.

```
on keyDown
  if (the controlDown)then doControlKey (the key)
end
```

Voir aussi

`charToNum()`, `commandDown`, `key()`, `keyCode()`, `optionDown`, `shiftDown`

controller

Syntaxe

```
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 point :

```
member("Démo").controller = 1
```

Syntaxe verbose :

```
set the controller of member "Démo" to 1
```

Voir aussi

`directToStage`

copyPixels()

Syntaxe

```
objetImage.copyPixels(objetImageSource, rectDeDestination, rectSource
  {, listeDeParamètres})
objetImage.copyPixels(objetImageSource, quadDeDestination, rectSource
  {, listeDeParamètres })
```

Description

Cette fonction copie le contenu du *`rectSource`* depuis l'*`objetImageSource`* vers l'*`objetImage`* donné. Les pixels sont copiés du *`rectSource`* dans l'*`objetImageSource`* et placés dans le *`rectDeDestination`* ou le *`quadDeDestination`* de l'*`objetImage`* donné. Pour plus d'informations sur l'utilisation des quadrilatères, consultez `quad`.

Vous pouvez inclure une liste de propriétés de paramètres facultative en vue du traitement des pixels copiés avant leur placement dans le *rectDeDestination*. La liste de propriétés peut contenir tout ou 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 d'arrière-plan par défaut est 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. Pour plus d'informations sur les valeurs possibles, consultez ink.
#blendLevel	Degré d'opacité (transparence) à appliquer aux pixels copiés. La plage de valeurs s'étend de 0 à 225. La valeur par défaut est 255 (opaque). L'utilisation d'une valeur inférieure à 255 définit le paramètre #ink sur #blend ou #blendTransparent s'il était initialement défini sur #backgroundTransparent. Vous pouvez également substituer #blend comme nom de propriété et utiliser une valeur de 0 à 100. Pour plus d'informations, consultez blend.
#dither	Valeur TRUE ou FALSE déterminant si les pixels copiés seront tramés lorsque 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 l'*objetImage*.
#useFastQuads	Valeur TRUE ou FALSE déterminant 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. Laissez ce paramètre sur FALSE (valeur par défaut) pour des quadrilatères aléatoires, tels que ceux utilisés pour les transformations de perspective. Pour plus d'informations, consultez useFastQuads.
#maskImage	Utilisé pour spécifier un objet Masque ou Dessin seul créé avec les fonctions `createMask()` ou `createMatte()` lors d'une utilisation comme masque pour les pixels à copier. Ceci vous permet de reproduire les effets des encres d'images-objets Masque et Dessin seul. Notez que si l'image source possède une couche alpha et que sa propriété `useAlpha` est TRUE, la couche alpha sera utilisée et l'encre Masque ou Dessin seul spécifiée sera ignorée. 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'*imageSource*. Le décalage par défaut est (0, 0).

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. La copie directe d'une zone à l'autre dans la même image est déconseillée. Voir `duplicate()`.

Pour simuler l'encre Dessin seul avec `copyPixels()`, créez un objet Dessin seul avec `createMatte()`, puis faites passer cet objet comme paramètre `#maskImage` avec `copyPixels()`.

La copie de pixels d'un objet image vers lui-même est déconseillée. Utilisez plutôt des images séparées.

Vous pourrez voir un exemple de `quad` dans une animation en consultant l'animation Quad du dossier Learning/Lingo_Examples, lui-même dans le dossier de Director.

Exemples

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.

```
member("fleur").image.copyPixels(member("Joyeux").image,  \
member("fleur").rect, member("Joyeux").rect)
```

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.

```
member("fleur").image.copyPixels(member("Joyeux").image,\
rect(20, 20, 100, 40), rect(0, 0, 200, 90))
```

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 est de taille identique à celle du rectangle de l'acteur Joyeux, de sorte que l'image copiée ne doive pas être 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.

```
member("fleur").image.copyPixels(member("Joyeux").image,\
member("Joyeux").rect, member("Joyeux").rect, [#blendLevel: 50])
```

Voir aussi

`ink`, `color()`

copyrightInfo

Syntaxe

```
member(quelActeur).copyrightInfo
copyrightInfo of member quelActeur
```

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 à point :

```
set quelEtat = the state of member "Fichier SWA"
if quelEtat > 1 AND quelEtat < 9 then
  put member("Fichier SWA").copyrightInfo into member("Infos")
end if
```

Syntaxe verbose :

```
set quelEtat = the state of member "Fichier SWA"
if quelEtat > 1 AND quelEtat < 9 then
  put the copyrightInfo of member "Fichier SWA" into member "Infos"
end if
```

copyToClipBoard

Syntaxe

```
member(quelActeur).copyToClipBoard()
copyToClipBoard member quelActeur
```

Description

Commande ; copie l'acteur spécifié dans le Presse-papiers sans que la fenêtre Distribution soit nécessairement active. Vous pouvez utiliser cette commande pour copier des acteurs entre plusieurs animations ou applications.

Exemples

L'instruction suivante copie l'acteur Chaise dans le Presse-papiers :

```
member("Chaise").copyToClipboard()
```

L'instruction suivante copie l'acteur numéro 5 dans le Presse-papiers :

```
member(5).copyToClipboard()
```

Voir aussi

`pasteClipBoardInto`

cos()

Syntaxe

```
(angle).cos
cos (angle)
```

Description

Fonction ; calcule le cosinus de l'angle spécifié, exprimé en radians.

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()

Syntaxe

```
liste.count
count (liste)
count(quelObjet)
quelObjet.count
expressionTexte.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`.

Vous pourrez voir un exemple de `count()` dans une animation en consultant l'animation Text du dossier Learning/Lingo_Examples, 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

Syntaxe

```
member(quelActeur).light.count
member(quelActeur).camera.count
member(quelActeur).modelResource(quelleRessDeMod).\
  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(quelleRessDeMod).\
  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.

Exemples

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

Syntaxe

```
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 le Macintosh.

Exemple

L'instruction suivante entraîne Director à libérer le processeur tous les 6 battements, ce qui représente un dixième de seconde :

```
the cpuHogTicks = 6
```

Voir aussi

`ticks`

creaseAngle

Syntaxe

```
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

Syntaxe

```
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`

createMask()

Syntaxe

`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. 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 Dégrad2 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()

Syntaxe

`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()`. 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()`

creationDate

Syntaxe

```
quelActeur.creationDate
the creationDate of quelActeur
```

Description

Cette propriété d'acteur enregistre la date et l'heure de la création initiale de l'acteur, à l'aide de l'heure système de l'ordinateur. Vous pouvez utiliser cette propriété pour programmer un projet ; elle n'est pas utilisée par Director.

Cette propriété peut être testée et définie.

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 :

```
put member(1).creationDate
  -- date( 1999, 12, 8 )
```

Voir aussi

`comments`, `modifiedBy`, `modifiedDate`

crop() (commande d'objet image)

Syntaxe

```
objetImage.crop(rectDeRecadrage)
```

Description

Utilisée avec un objet image, renvoie un nouvel objet image contenant une copie de l'objet image concerné, recadrée dans le rectangle donné. L'objet image d'origine reste inchangé. Le nouvel objet image n'appartient plus à aucun acteur et n'est plus associé à la scène. Pour l'affecter à un acteur, vous devez définir la propriété `image` de cet acteur.

Cette opération diffère de l'application à un acteur de la propriété `crop`, qui recadre l'acteur même, en modifiant l'original.

Exemples

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 :

```
set imageDeLaScène = (the stage).image
set imageDLimageObjet = imageDeLaScène.crop(sprite(10).rect)
member("instantané de l'image-objet").image = imageDeLimageObjet
```

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)
```

crop() (commande d'acteur)

Syntaxe

```
member(quelActeur).crop(rectDeRecadrage)
crop member quelActeur, rectDeRecadrage
```

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.

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 :

```
member("image de la scène").picture = (the stage).picture
member("image de la scène").crop(sprite(10).rect)
```

Voir aussi

`picture (propriété d'acteur)`

crop (propriété d'acteur)

Syntaxe

```
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 point :

```
member("Interview").crop = TRUE
```

Syntaxe verbose :

```
set the crop of member "Interview" to TRUE
```

Voir aussi

`center`

cross

Syntaxe

```
vecteur1.cross(vecteur2)
```

Description

Méthode de vecteur 3D ; 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()

Syntaxe

```
vecteur1.crossProduct(vecteur2)
```

Description

Méthode de vecteur 3D ; 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`

on cuePassed

Syntaxe

```
on cuePassed(IDdePiste, numéroDeRepère,nomDeRepère)
  instruction(s)
end
on cuePassed(me,IDdePiste, numéroDeRepère,nomDeRepère)
  instruction(s)
end
```

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 :

```
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
```

Voir aussi

`scriptInstanceList`, `cuePointNames`, `cuePointTimes`, `isPastCuePoint()`

cuePointNames

Syntaxe

```
member(quelActeur).cuePointNames
the cuePointNames of member quelActeur
```

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 supportée 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 à point :

```
put member("symphonie").cuePointNames[3]
```

Syntaxe verbose :

```
put (getAt(the cuePointNames of member "symphonie",3))
```

Voir aussi

`cuePointTimes`, `mostRecentCuePoint`

cuePointTimes

Syntaxe

```
member(quelActeur).cuePointTimes
the cuePointTimes of member quelActeur
```

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 supportée par les acteurs SoundEdit, les acteurs vidéo numérique QuickTime et les acteurs Xtras supportant 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 à point :

```
put member("symphonie").cuePointTimes[3]
```

Syntaxe verbose :

```
put (getAt(the cuePointTimes of member "symphonie",3))
```

Voir aussi

`cuePointNames`, `mostRecentCuePoint`

currentLoopState

Syntaxe

```
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 (propriété d'acteur)`, `play() (3D)`, `queue() (3D)`, `playlist`

currentSpriteNum

Syntaxe

```
the currentSpriteNum
```

Description

Propriété d'animation ; indique le numéro de piste de l'image-objet dont le script est en cours d'exécution. Elle 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 `spriteNum of me`, mais ne nécessite pas la présence de la référence `me`.

Cette propriété peut être testée, mais pas définie.

Remarque Cette propriété était plus utile pour le passage des anciennes animations à Director 6, à l'apparition des comportements. Elle permettait une fonction de type comportement sans obliger à réécrire complètement le code Lingo. 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` :

```
on mouseDown
  sprite(the currentSpriteNum).member = member "imageCliquée"
end
```

Voir aussi

`me`, `spriteNum`

currentTime

Syntaxe

```
sprite(quelleImageObjet).currentTime
the currentTime of sprite quelleImageObjet
sound(numéroDePiste).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 quel 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 à point :

```
member("position").text = string(sprite (10).currentTime/ 1000)
```

Syntaxe verbose :

```
set the text of member "position" to (the currentTime of sprite 10) / 1000
```

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 :

```
sound(2).currentTime = 2700
```

Voir aussi

`movieTime, duration`

currentTime (3D)

Syntaxe

```
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 (RealMedia)

Syntaxe

```
sprite(quelleImageObjet).currentTime
member(quelActeur).currentTime
sprite(quelleImageObjet).currentTime = positionEnMillisecondes
member(quelActeur).currentTime = positionEnMillisecondes
```

Description

Propriété d'image-objet ou d'acteur RealMedia ; permet d'obtenir ou de définir la position du train 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 train 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 train, avant que la lecture ne reprenne à la nouvelle position définie. Si le train est interrompu (valeur `#paused` `mediaStatus`) lors de la définition ou de la modification de la propriété `currentTime`, le train redessine l'image à la nouvelle position, et la lecture reprend si la propriété `pausedAtStart` a pour valeur `FALSE`. Lorsque le train est interrompu ou arrêté dans la fenêtre RealMedia, la propriété `mediaStatus` a pour valeur `#paused`. Lorsque le train 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 train.

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 train.

Exemples

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 train.

```
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 train de l'image-objet 2 et de l'acteur Real.

```
sprite(2).currentTime = 20000
member("Real").currentTime = 20000
```

Voir aussi

`duration (RealMedia)`, `seek`, `mediaStatus`

cursor (commande)

Syntaxe

```
cursor [numéroDacteur, numéroDacteurMasque]
cursor quelCurseur
cursor (member quelActeurCurseur)
```

Description

Commande ; change l'acteur ou le curseur intégré utilisé comme pointeur et reste en vigueur jusqu'à ce que vous la désactiviez en lui affectant la valeur 0.

- Utilisez la syntaxe `cursor [numéroDacteur, numéroDacteurMasque]` 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 `cursor quelCurseur` pour spécifier les curseurs par défaut fournis par le système. Le terme `quelCurseur` doit être l'un des nombres entiers suivants :

`0*`	Aucun curseur défini
`-1`	Flèche
`1`	I
`2`	Croix
`3*`	Croix épaisse
`4`	Montre (Macintosh uniquement)
`200*`	Curseur vide (masque le curseur)

* Le lecteur Director pour Java ne supporte pas ces types de curseurs et affiche à la place un curseur flèche.

- Utilisez la syntaxe `cursor (member quelActeurCurseur)` pour les curseurs personnalisés proposés par l'Xtra Cursor.

Veillez à ne pas confondre les syntaxes `cursor 1` et `cursor [1]`. La première sélectionne un curseur en I dans le jeu de curseurs système ; la deuxième utilise l'acteur 1 comme curseur personnalisé.

Remarque L'Xtra Cursor accepte différents types d'acteurs comme curseurs, mais pas les acteurs texte.

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 de toute nouvelle animation chargée dans une présentation utilisant un curseur personnalisé pour plusieurs animations, stockez donc 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.

Exemple

L'instruction suivante change le curseur en curseur montre sur un Macintosh et en sablier sous Windows, si la valeur de la variable *Etat* est égale à 1 :

```
if Etat = 1 then 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 :

```
on monCurseur unActeur
  if the depth of member unActeur = 1 then
    cursor[unActeur]
      else
    beep
  end if
end
```

Voir aussi

cursor (propriété d'image-objet), rollOver()

cursor (propriété d'image-objet)

Syntaxe

```
sprite(quelleImageObjet).cursor = [numéroDacteur, numéroDacteurMasque]
set the cursor of sprite quelleImageObjet to [numéroDacteur,
  numéroDacteurMasque]
sprite(quelleImageObjet).cursor = quelCurseur
set the cursor of sprite quelleImageObjet to quelCurseur
```

Description

Propriété d'image-objet ; détermine le curseur utilisé lorsque le pointeur se trouve au-dessus de l'image-objet spécifiée par l'expression entière *quelleImageObjet*. 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 `cursor of sprite...[numéroDacteur, numéroDacteurMasque]` pour spécifier le numéro d'acteur à utiliser comme curseur et son masque facultatif.
- Utilisez la syntaxe `cursor of sprite...quelCurseur` pour spécifier les curseurs par défaut fournis par le système. Le terme *quelCurseur* doit être l'un des nombres entiers suivants :

0*	Aucun curseur défini
-1	Flèche
1	I
2	Croix
3*	Croix épaisse
4	Montre (Macintosh uniquement)
200*	Curseur vide (masque le curseur)

* Le lecteur Director pour Java ne supporte pas ces types de curseurs et affiche à la place un curseur flèche.

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.

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 remplaçant *`quelCurseur`* par le numéro de la ressource de curseur.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante change en montre le curseur qui apparaît sur l'image-objet 20 :

Syntaxe à point :

```
sprite(20).cursor = 4
```

Syntaxe verbose :

```
set the cursor of sprite 20 to 4
```

Voir aussi

`cursor (commande)`

cursorSize

Syntaxe

```
member(quelActeurCurseur).cursorSize
the cursorSize of member quelActeurCurseur
```

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 à point :

```
member(20).cursorSize = 32
```

Syntaxe verbose :

```
set the cursorSize of member 20 = 32
```

curve

Syntaxe

acteur.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`.

Exemples

L'instruction suivante affiche la liste des sommets de la troisième courbe de l'acteur forme vectorielle CourbesSimples :

```
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 :

```
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.

```
emplacementDuSommet = member(1).curve[1].vertex[1]
emplacementDeLimageObjet = mapMemberToStage(sprite(3), emplacementDuSommet)
sprite(7).loc = spriteLoc
```

Voir aussi

`vertex, vertexList`

date() (horloge du système)

Syntaxe

```
the abbr date
the abbrev date
the abbreviated date
the date
the long date
the short date
```

Description

Fonction ; renvoie la date courante fournie par l'horloge du système dans l'un des trois formats suivants : `abbreviated`, `long` ou `short` (valeur par défaut). Le format abrégé peut également être abrégé en `abbrev` et `abbr`.

La fonction `date` est disponible dans Java, mais n'accepte pas les variantes `abbrev`, `long` ou `short`. Lorsque l'animation est lue dans une applet, le format de date est `JJ/MM/AA`, où `JJ` représente le jour, `MM` le mois et `AA` les deux derniers chiffres de l'année en cours. Pour les mois de janvier à septembre, la valeur de `MM` est un seul chiffre.

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 que les parties de la date au format court indiquent.
- Sur le Macintosh, vous pouvez personnaliser l'affichage de la date dans le tableau de bord Date et heure.

Exemples

L'instruction suivante renvoie la date au format abrégé :

```
put the abbreviated date
-- "Sam. 7 sep 2002"
```

L'instruction suivante renvoie la date au format long :

```
put the long date
-- "Samedi 7 septembre 2002"
```

L'instruction suivante renvoie la date au format court :

```
put the short date
-- "7/9/02"
```

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 :

```
if char 1 to 4 of the date = "1/1/" then alert "Bonne année !"
```

Remarque Les trois formats de date varient en fonction du pays pour lequel le système d'exploitation a été conçu. Utilisez l'objet `date` pour créer et manipuler les dates en format standard.

Voir aussi

`time()`, `date() (formats)`, `systemDate`

date() (formats)

Syntaxe

```
date(chaîneAuFormatISO)
date(entierAuFormatISO)
date(jourEntierAuFormatISO, moisEntierAuFormatISO, annéeEntierAuFormatISO)
```

Description

Fonction et type de données ; crée une instance d'objet de date au format standard que vous pouvez utiliser avec d'autres instances de date dans des opérations mathématiques ou pour manipuler les dates d'une plate-forme à l'autre, ainsi que dans des formats de pays différents.

Pour créer la date, utilisez des années sur quatre chiffres, des mois sur deux chiffres et des jours sur deux chiffres. Les expressions suivantes sont équivalentes :

entier :	`set débutDesVacances = date(06181998)`
chaîne :	`set débutDesVacances = date("06181998")`
séparation par virgules :	`set débutDesVacances = date(06, 18, 1998)`

Les opérations d'addition et de soustraction sur la date sont interprétées comme l'addition et la soustraction de jours.

Les différentes propriétés de l'instance d'objet de date renvoyée sont :

#year	Nombre entier représentant l'année
#month	Nombre entier représentant le mois de l'année
#day	Nombre entier représentant le jour du mois

Exemples

Les instructions suivantes créent et déterminent le nombre de jours séparant deux dates :

```
maDateDeNaissance = date(12071965)
taDateDeNaissance = date(29051945)
put "Il y a" && abs(taDateDeNaissance - maDateDeNaissance) && "jours entre nos
  dates de naissances."
```

Les instructions suivantes permettent d'accéder à la propriété individuelle d'une date :

```
maDateDeNaissance = date(12071965)
put "Je suis né au mois numéro"&&maDateDeNaissance.month
```

Voir aussi

date() (horloge du système)

deactivateApplication

Syntaxe

```
on deactivateApplication
```

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 :

```
on deactivateApplication
  sound(1).queue(member("sonDeFermeture"))
  sound(1).play()
end
```

Voir aussi

`add (texture 3D), activeCastLib, on deactivateWindow`

on deactivateWindow

Syntaxe

```
on deactivateWindow
  instruction(s)
end
```

Description

Message système et gestionnaire d'événements ; contient des instructions 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 Lingo 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 :

```
on deactivateWindow
  puppetSound 2, "Ronflement"
end
```

debug

Syntaxe

`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

Syntaxe

```
the debugPlaybackEnabled
```

Description

Propriété ; sous Windows, ouvre une fenêtre Messages à des fins de débogage dans Shockwave et les projections. Elle 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 ou projection particulière. Si plusieurs animations Shockwave utilisent ce code Lingo dans un seul navigateur, seule la première ouvrira une fenêtre Messages, qui sera liée uniquement à cette animation.

Sur le Macintosh, c'est un fichier journal qui est ouvert et dans lequel des instructions `put` sont générées pour placer des données à des fins de débogage. Ce fichier se trouve dans le dossier Shockwave 8, 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 ou dans une projection :

```
the debugPlaybackEnabled = TRUE
```

decayMode

Syntaxe

```
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

Syntaxe

```
member(quelActeurFlashOuFormeVectorielle).defaultRect
the defaultRect of member quelActeurFlashOuFormeVectorielle
```

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.

```
on réglerLeRectFlashParDéfaut quelleDistribution, quelRect
  repeat with i = 1 to the number of members of castLib quelleDistribution
    if member(i, quelleDistribution).type = #flash then
      member(i, quelleDistribution).defaultRect = quelRect
    end if
  end repeat
end
```

Voir aussi

`defaultRectMode`, `flashRect`

defaultRectMode

Syntaxe

```
member(quelActeurVecteurOuFlash).defaultRectMode
the defaultRectMode of member quelActeurVecteurOuFlash
```

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)`.

```
on réglerLaTailleParDéfautDeRect quelleDistribution
  repeat with i = 1 to the number of members of castLib quelleDistribution
    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
```

Voir aussi

`flashRect`, `defaultRect`

delay

Syntaxe

`delay` *`nombreDeBattements`*

Description

Commande ; immobilise la tête de lecture pendant une durée donnée. L'expression entière *`nombreDeBattements`* spécifie le nombre de battements de l'attente, où chaque battement correspond à 1/60ème de seconde. La seule activité de souris et de clavier possible à ce moment-là est d'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 commande `delay` ne peut être appliquée que lorsque la tête de lecture est en mouvement. Les gestionnaires peuvent cependant toujours être exécutés pendant la durée de la commande `delay` : seule la tête de lecture est immobilisée, alors que l'exécution du script ne l'est pas. Placez les scripts utilisant la commande `delay` dans un gestionnaire `on enterFrame` ou `on exitFrame`.

Pour simuler une interruption dans un gestionnaire lorsque la tête de lecture est immobile, utilisez la commande `startTimer` ou affectez la valeur courante de la propriété `timer` à une variable et attendez un certain temps avant de sortir de l'image.

Exemples

Le gestionnaire suivant interrompt l'animation pendant 2 secondes lorsque la tête de lecture quitte l'image en cours :

```
on exitFrame
  delay 2 * 60
end
```

Le gestionnaire suivant, qui peut être placé dans un script d'image, retarde l'animation d'un nombre aléatoire de battements :

```
on keyDown
  if the key = RETURN then delay random(180)
end
```

Exemple

Le premier de ces gestionnaires démarre un compteur lorsque la tête de lecture quitte une image. Le second gestionnaire, affecté à l'image suivante, la met en boucle jusqu'à ce que le temps spécifié soit écoulé :

```
--script de la première image

on exitFrame
  global gCompteur
  set gCompteur = the ticks
end

--script de la seconde image

on exitFrame
  global gCompteur
  if the ticks < gCompteur + (10 * 60) then
    go to the frame
  end if
end
```

Voir aussi

`startTimer`, `ticks`, `timer`

delete

Syntaxe

`delete` *`expressionSousChaîne`*

Description

Commande ; supprime la sous-chaîne (caractère, mot, élément ou ligne) spécifiée d'un texte source. Les sources de chaînes incluent les acteurs champ et les variables contenant des chaînes.

Vous pourrez voir un exemple de `delete` dans une animation en consultant l'animation Text du dossier Learning/Lingo_Examples, lui-même dans le dossier de Director.

Exemples

L'instruction suivante supprime le premier mot de la troisième ligne de l'acteur champ Adresse :

```
delete word 1 of line 3 of member "Adresse"
```

La syntaxe suivante permet également de supprimer la même sous-chaîne :

```
delete member("Adresse").line[3].word[1]
```

L'instruction suivante supprime le premier caractère de la chaîne dans la variable *`montantDuDevis`* si ce caractère est le signe $:

```
if montantDuDevis.char[1] = "$" then delete montantDuDevis.char[1]
```

Voir aussi

`char...of`, `field`, `item...of`, `line...of`, `word...of`, `hilite (propriété d'acteur)`, `paragraph`

deleteAll

Syntaxe

```
liste.deleteAll()
deleteAll liste
```

Description

Commande de liste ; supprime tous les éléments présents dans la liste spécifiée sans changer le type de cette liste.

Exemple

L'instruction suivante supprime chaque élément de la liste listeDesPropriétés :

```
listeDesPropriétés.deleteAll()
```

deleteAt

Syntaxe

```
liste.deleteAt(nombre)
deleteAt liste, nombre
```

Description

Commande de liste ; supprime l'élément se trouvant à la position spécifiée par *nombre* dans la liste linéaire ou de propriétés spécifiée par `liste`.

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.

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

Syntaxe

```
member(quelActeur).deleteCamera(nomDeCaméra)
member(quelActeur).deleteCamera(index)
sprite(quelleImageObjet).deleteCamera(cameraOrIndex)
```

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.

Exemples

L'instruction suivante supprime deux caméras de l'acteur Pièce : tout d'abord la caméra Caméra06, puis la caméra 3.

```
member("Pièce").deleteCamera("Caméra06")
member("Pièce").deleteCamera(3)
```

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

Syntaxe

```
deleteFrame
```

Description

Commande ; supprime l'image courante et fait de l'image suivante la nouvelle image courante pendant une session de création du scénario uniquement.

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 :

```
on testImageObjet
  beginRecording
    if sprite(10).locH > 640 then deleteFrame
  endRecording
end
```

Voir aussi

`beginRecording`, `endRecording`, `updateFrame`

deleteGroup

Syntaxe

```
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 Univers, qui est le groupe par défaut.

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, parent

deleteLight

Syntaxe

```
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.

Exemples

Les exemples suivants suppriment les lumières de l'acteur Pièce.

```
member("Pièce").deleteLight("lumièreAmbiante")
member("Pièce").deleteLight(6)
```

Voir aussi

newLight

deleteModel

Syntaxe

```
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.

Exemples

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

Syntaxe

```
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.

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

Syntaxe

```
member(quelActeur).deleteMotion(quelMouvement)
member(quelActeur).deleteMotion(index)
```

Description

Commande 3D ; supprime le mouvement de l'acteur.

Exemples

La première ligne de l'exemple suivant supprime le mouvement Saut de l'acteur scèneDePicnic. La deuxième ligne supprime le cinquième mouvement de scèneDePicnic.

```
member("scèneDePicnic").deleteMotion("Saut")
member("scèneDePicnic").deleteMotion(5)
```

Voir aussi

`newMotion()`, `removeLast()`

deleteOne

Syntaxe

```
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.

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

Syntaxe

```
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.

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

Syntaxe

```
member(quelActeur).deleteShader(quelMatériau)
member(quelActeur).deleteShader(index)
```

Description

Commande 3D ; retire le matériau de l'acteur.

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

Syntaxe

```
member(quelActeur).deleteTexture(quelleTexture)
member(quelActeur).deleteTexture(index)
```

Description

Commande 3D ; retire le matériau de l'acteur.

Exemple

La première ligne de cet exemple supprime la texture Ciel de l'acteur scèneDePicnic. La deuxième ligne supprime la cinquième texture de scèneDePicnic.

```
member("scèneDePicnic).deleteTexture("Ciel")
member("scèneDePicnic").deleteTexture(5)
```

Voir aussi

`newTexture`

deleteVertex()

Syntaxe

```
member(réfDacteur).deleteVertex(indexAsupprimer)
deleteVertex(member réfDacteur, indexAsupprimer)
```

Description

Commande de forme vectorielle ; supprime un sommet existant d'un acteur forme vectorielle à la position d'index spécifiée.

Exemple

La ligne suivante supprime le deuxième point de sommet de la forme vectorielle Archie :

```
member("Archie").deleteVertex(2)
```

Voir aussi

`addVertex`, `moveVertex()`, `originMode`, `vertexList`

density

Syntaxe

```
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("Scène").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("Scène").shader("gbMatériau").density = 100
```

Voir aussi

newShader

depth

Syntaxe

```
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.

Exemples

L'instruction suivante affiche le codage des couleurs de l'objet image stocké dans la variable *nouvelleImage*. Le résultat apparaît dans la fenêtre Messages.

```
put nouvelleImage.depth
```

L'instruction suivante affiche le codage des couleurs de l'acteur Temple dans la fenêtre Messages :

```
put member("Temple").depth
```

depth (3D)

Syntaxe

```
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 lors de l'utilisation du 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("Scène").model("Bébé").sds.depth = 3
```

Voir aussi

`sds (modificateur)`, `error`, `tension`

depthBufferDepth

Syntaxe

```
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

Syntaxe

```
the deskTopRectList
```

Description

Propriété système ; affiche la taille, et la position sur le bureau, des moniteurs connectés à l'ordinateur. 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.

Cette propriété peut être testée, mais pas définie.

Exemples

L'instruction suivante teste la taille des moniteurs connectés à l'ordinateur et affiche le résultat dans la fenêtre Messages :

```
put the deskTopRectList
-- [rect(0,0,1024,768), rect(1025, 0, 1665, 480]
```

Le résultat indique que le premier moniteur est de 1024x768 pixels et le second de 640x480 pixels.

Le gestionnaire suivant indique le nombre de moniteurs du système courant :

```
on nombreDeMoniteurs
  return the deskTopRectList.count
end
```

diffuse

Syntaxe

```
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

Syntaxe

```
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

Syntaxe

```
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

Syntaxe

```
the digitalVideoTimeScale
```

Description

Propriété système ; détermine l'échelle temporelle, en unités par seconde, que le système utilise pour mesurer la durée des acteurs vidéo numérique.

La propriété `digitalVideoTimeScale` peut recevoir n'importe quelle valeur.

La valeur de la 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.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante définit l'échelle temporelle utilisée par le système pour mesurer la vidéo numérique à 600 unités par seconde :

```
the digitalVideoTimeScale to 600
```

digitalVideoType

Syntaxe

```
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

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).\
  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` 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`

directionalColor

Syntaxe

```
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

Syntaxe

`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

Syntaxe

```
member(quelActeur).directToStage
the directToStage of member quelActeur
sprite(quelleImageObjet).directToStage
the directToStage of sprite quelleImageObjet
```

Description

Propriété d'image-objet et d'acteur; détermine le plan sur lequel un acteur vidéo numérique, GIF animé, forme vectorielle, 3D 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_Examples, 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 :

```
member("Résidents").directToStage = 1
```

disableImagingTransformation

Syntaxe

```
the disableImagingTransformation
```

Description

Propriété Lingo graphique ; lorsque `TRUE`, empêche Director de prendre en compte le défilement ou le zoom sur la scène lorsque `(the stage).image` est utilisé. Lorsque la propriété `disableImagingTransformation` 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. Lorsque cette propriété a pour valeur `TRUE`, le zoom et le défilement de la scène affectent l'apparence de l'image capturée par `(the stage).image`.

displayFace

Syntaxe

```
member(quelActeurTexte).displayFace
member(quelActeur3D).modelResource(quelleRessDeMod).\
  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("Scène").model("Slogan").resource.displayFace = \
  [#back, #tunnel]
```

Voir aussi

`extrude3D`, `displayMode`

displayMode

Syntaxe

```
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

Syntaxe

```
sprite(quelleImageObjet).displayRealLogo
member(quelActeur).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 train 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.

Exemples

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.

```
put 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.

```
sprite(2).displayRealLogo = FALSE
member("Real").displayRealLogo = FALSE
```

distanceTo()

Syntaxe

```
vecteur1.distanceTo(vecteur2)
```

Description

Méthode 3D de vecteur ; renvoie la distance, en unités d'univers, séparant deux positions vectorielles.

Exemple

L'exemple suivant compte trois vecteurs. La distance séparant Vec1 de Vec2 est 100.0000 unités de l'univers. La distance séparant Vec1 de Vec3 est 141.4214 unités de l'univers.

```
Vec1 = vector(100, 0, 0)
Vec2 = vector(100, 100, 0)
Vec3 = vector(100, 100, 100)
put Vec1.distanceTo(Vec2)
-- 100.0000
put Vec1.distanceTo(Vec3)
-- 141.4214
```

Voir aussi

magnitude

distribution

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).\
  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

Syntaxe

```
member(quelActeur).dither
the dither of member quelActeur
```

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`

do

Syntaxe

```
do expressionChaîne
```

Description

Commande ; évalue *`expressionChaîne`* et exécute le résultat sous la forme d'une instruction Lingo. 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 les variables locales en avance.

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.

Exemple

L'instruction suivante exécute l'instruction placée entre guillemets :

```
do "beep 2"
do listeDeCommandes[3]
```

doneParsing()

Syntaxe

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.

Voir aussi

parseURL()

dot()

Syntaxe

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 arriver manuellement 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().

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 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.dot(norm2)
-- 0.7071
```

Voir aussi

dotProduct(), getNormalized, normalize

dotProduct()

Syntaxe

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, le produit est le cosinus de l'angle entre les deux vecteurs.

Pour arriver manuellement 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()`.

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`

doubleClick

Syntaxe

```
the doubleClick
```

Description

Fonction ; 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`).

Exemples

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 :

```
if the doubleClick then go to frame "Entrée de l'offre"
```

Le gestionnaire suivant vérifie si un double-clic a eu lieu. Lorsque l'utilisateur clique sur le bouton de la souris, une boucle de répétition est exécutée pendant l'intervalle déterminé pour un double-clic (20 battements dans ce cas). Si un second clic se produit pendant cet intervalle, le gestionnaire `actionDeDoubleClic` est exécuté. Dans le cas contraire, le gestionnaire `actionDeClicSimple` est exécuté :

```
on mouseUp
  if the doubleClick then exit
  startTimer
  repeat while the timer < 20
    if the mouseDown then
      actionDeDoubleClic
      exit
    end if
  end repeat
  actionDeClicSimple
end mouseUp
```

Voir aussi

`clickOn`, `the mouseDown (propriété système)`, `the mouseUp (propriété système)`

downloadNetThing

Syntaxe

`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.

- *`URL`* – Nom de fichier de n'importe quel objet pouvant être téléchargé : par exemple un serveur FTP ou HTTP, une page HTML, un acteur externe, une animation Director, un graphique, etc.
- *`fichierLocal`* – Chemin d'accès et nom du fichier sur le disque local.

Les animations Director en mode auteur et les projections supportent la commande `downLoadNetThing`, mais celle-ci n'est pas supportée par Shockwave. 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`.

Remarque Director pour Java ne supporte pas la commande `downloadNetThing`.

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()`

drag

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).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`

draw()

Syntaxe

```
objetImage.draw(x1, y1, x2, y2, objetCouleurOuListedeParamètres)
objetImage.draw(point(x, y), point(x, y), objetCouleurOuListedeParamètres)
objetImage.draw(rect, objetCouleurOuListedeParamètres)
```

Description

Cette fonction dessine une ligne ou une forme vide de couleur *objetCouleur* dans une zone rectangulaire de l'objet image donné, tel que spécifié dans les trois méthodes présentées. La fonction `draw` renvoie une valeur de 1 si aucune erreur n'est détectée. Vous pouvez utiliser la fonction facultative *ListeDeParamètres* de liste de propriétés pour spécifier les propriétés de forme 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.

En cas d'absence de liste de paramètres, cette fonction dessine un trait 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 une valeur inférieure), l'*objetCouleur* 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 fonction `fill()`.

Exemples

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.

```
member("Joyeux").image.draw(0, 0, 128, 86, rgb(150,0,0))
```

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).

```
member("Joyeux").image.draw(0, 0, 128, 86, [#shapeType:#oval, #lineSize:3, \
#color: rgb(150, 0, 0)])
```

Voir aussi

`color()`, `copyPixels()`, `fill()`, `setPixel()`

drawRect

Syntaxe

```
window nomDeFenêtre.drawRect
the drawRect of window nomDeFenêtre
```

Description

Propriété de fenêtre ; identifie les coordonnées rectangulaires de la scène de l'animation qui apparaît dans la fenêtre. 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.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante affiche les coordonnées courantes de la fenêtre d'animation intitulée Tableau de commande :

```
put the drawRect of window "Tableau de commande"
-- rect(10, 20, 200, 300).
```

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.

```
set the drawRect of window "Tableau de commande" to rectangleDanimation
```

Les lignes suivantes entraînent le remplissage de la zone principale du moniteur par la scène :

```
(the stage).drawRect = the desktopRectList[1]
(the stage).rect = the desktopRectList[1]
```

Voir aussi

deskTopRectList, rect (caméra), sourceRect

dropShadow

Syntaxe

```
member(quelActeur).dropShadow
the dropShadow of member quelActeur
```

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 :

```
member("Commentaire").dropShadow = 5
```

duplicate

Syntaxe

```
référenceDeVecteur.duplicate()
référenceDeTransformation.duplicate()
```

Description

Méthode 3D de vecteur et transformation ; renvoie une copie du vecteur ou de la transformation.

Exemple

L'instruction suivante crée une copie de la position du modèle 1 et l'enregistre dans la variable *zz*.

```
zz = member("maSalle").model[1].transform.position.duplicate()
```

Voir aussi

clone

duplicateFrame

Syntaxe

```
duplicateFrame
```

Description

Commande ; copie l'image courante et son contenu, insère la copie après l'image courante et fait de la copie l'image courante. Cette commande peut être utilisée pendant la création du scénario uniquement.

La commande duplicateFrame a la même fonction que la commande insertFrame.

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`.

```
on animBalle nombreDimages
  beginRecording
    sprite(20).member = member("Balle", "Jouets")
    repeat with i = 0 to nombreDimages
      duplicateFrame
    end repeat
  endRecording
end
```

duplicate() (fonction de liste)

Syntaxe

```
(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.

Vous pourrez voir un exemple de `duplicate() (fonction de liste)` dans une animation en consultant l'animation Vector Shapes du dossier Learning/Lingo_Examples, lui-même dans le dossier de Director.

Exemple

L'instruction suivante copie la liste ClientsDuJour et l'affecte à la variable *`ListeDesClients`* :

```
ListeDesClients = ClientsDuJour.duplicate()
```

duplicate() (fonction d'image)

Syntaxe

```
objetImage.duplicate()
```

Description

Cette fonction crée et renvoie une copie de l'*`objetImage`* donné. La nouvelle image est entièrement indépendante de l'originale 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.

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

`duplicate member`

duplicate member

Syntaxe

```
member(acteurDorigine).duplicate()
member(acteurDorigine).duplicate({nouvellePosition})
duplicate member acteurDorigine {, nouvellePosition}
```

Description

Commande ; copie l'acteur spécifié par *acteurDorigine*. Le paramètre facultatif *nouvellePosition* spécifie une position dans la fenêtre de distribution pour l'acteur copié. Si le paramètre *nouvellePosition* n'est pas inclus, la copie de l'acteur est placée dans la première position disponible de la fenêtre Distribution.

Cette commande trouve sa meilleure utilisation en programmation plutôt qu'à l'exécution, car elle crée un autre acteur en mémoire, ce qui risque d'entraîner des problèmes de mémoire. Utilisez la commande pendant les étapes de la programmation pour que les modifications apportées à la distribution soient enregistrées avec le fichier.

Exemples

L'instruction suivante copie l'acteur Bureau et le place dans la première position vide de la fenêtre Distribution :

```
member("Bureau").duplicate()
```

L'instruction suivante copie l'acteur Bureau et le place à la position 125 :

```
member("Bureau").duplicate(125)
```

duration

Syntaxe

```
member(quelActeur).duration
the duration of member quelActeur
```

Description

Propriété d'acteur ; détermine la durée des acteurs Shockwave Audio (SWA), transition et QuickTime spécifiés.

- Si *quelActeur* 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 *quelActeur* est un acteur vidéo numérique, cette propriété indique la durée de la vidéo numérique. La valeur est exprimée en battements.
- Si *quelActeur* 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 supportent, 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_Examples, lui-même dans le dossier de Director.

Exemples

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 :

```
on exitFrame
  if member("Louie Prima").state = 2 then
    member("Affichage de la durée").text = member("Louie Prima").duration
  end if
end
```

Vous pouvez utiliser un comportement sur une image-objet vidéo numérique pour que la tête de lecture effectue une boucle sur l'image courante jusqu'à la fin de la lecture de la séquence, lui permettant de continuer après la fin :

```
property spriteNum

on exitFrame me
  monActeur = sprite(spriteNum).member
  maDurée = member(monActeur).duration
  duréeDeMonAnimation = sprite(spriteNum).movieTime
  if maDurée > duréeDeMonAnimation then
    go to the frame
  else
    go to the frame + 1
  end if
end
```

duration (3D)

Syntaxe

```
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 (RealMedia)

Syntaxe

```
sprite(quelleImageObjet).duration
member(quelActeur).duration
```

Description

Propriété d'acteur ou image-objet RealMedia ; indique la durée du train RealMedia, en millisecondes. La durée du train RealMedia reste inconnue juqu'au démarrage de la lecture de l'acteur. Si le train provient d'un train en direct ou n'a pas été lu, la valeur de cette propriété est 0. Cette propriété peut être testée, mais pas définie.

Exemples

Les exemples suivants indiquent que l'état du train RealMedia de l'image-objet 2 et de l'acteur Real est 100 500 millisecondes (100,50 secondes).

```
put sprite(2).duration
-- 100500

put member("Real").duration
-- 100500
```

Voir aussi

play (RealMedia), seek, currentTime (RealMedia)

editable

Syntaxe

```
member(quelActeur).editable
the editable of member quelActeur
sprite(quelleImageObjet).editable
the editable of sprite quelleImageObjet
```

Description

Propriété d'acteur et d'image-objet ; détermine si l'acteur champ spécifié peut être modifié sur la scène (TRUE) ou non (FALSE).

Lorsque la propriété d'acteur est définie, le paramètre est appliqué à toutes les images-objets contenant le champ. Lorsque la propriété d'image-objet est définie, seule l'image-objet spécifiée est affectée.

Vous pouvez également rendre un acteur 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 Lingo dure au-delà de l'image-objet courante, celle-ci doit être un esclave.

Cette propriété peut être testée et définie.

Exemples

L'instruction suivante rend l'acteur champ Réponse modifiable :

```
member("Réponse").editable = TRUE
```

Le gestionnaire suivant fait de la piste d'image-objet un esclave et rend l'image-objet champ modifiable :

```
on mesNotes
  puppetSprite 5, TRUE
  sprite(5).editable = TRUE
end
```

L'instruction suivante vérifie si une image-objet champ est modifiable et affiche un message le cas échéant :

```
if sprite(13).editable = TRUE then
  member("Notice").text = "Veuillez saisir votre réponse ci-dessous."
end if
```

editShortCutsEnabled

Syntaxe

```
the editShortCutsEnabled
```

Description

Propriété d'animation ; détermine si les opérations Couper, Copier et Coller, ainsi que leurs raccourcis clavier, fonctionnent dans l'animation courante. Ces opérations de texte fonctionnent si elles sont définies sur `TRUE`. Ces opérations ne sont pas autorisées si elles sont définies sur `FALSE`.

Cette propriété peut être testée et définie. 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 :

```
the editShortCutsEnabled = 0
```

elapsedTime

Syntaxe

```
sound(numéroDePiste).elapsedTime
the elapsedTime of sound numéroDePiste
```

Description

Cette propriété en lecture seule indique la durée écoulée, en millisecondes, depuis le début de la lecture de l'acteur son courant dans la piste audio indiquée. Elle 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 `on idle` affiche le temps écoulé pour la piste audio 4 dans un champ de la scène en période d'inactivité :

```
on idle
  member("durée").text = string(sound(4).elapsedTime)
end idle
```

Voir aussi

`currentTime`, `loopCount`, `loopsRemaining`, `rewind()`

emissive

Syntaxe

```
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

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).\
  emitter.numParticles
member(quelActeur).modelResource(quelleRessDeMod).\
  emitter.mode
member(quelActeur).modelResource(quelleRessDeMod).\
  emitter.loop
member(quelActeur).modelResource(quelleRessDeMod).\
  emitter.direction
member(quelActeur).modelResource(quelleRessDeMod).\
  emitter.region
member(quelActeur).modelResource(quelleRessDeMod).\
  emitter.distribution
member(quelActeur).modelResource(quelleRessDeMod).\
  emitter.angle
member(quelActeur).modelResource(quelleRessDeMod).\
  emitter.path
member(quelActeur).modelResource(quelleRessDeMod).\
  emitter.pathStrength
member(quelActeur).modelResource(quelleRessDeMod).\
  emitter.minSpeed
member(quelActeur).modelResource(quelleRessDeMod).\
  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`, `path`, `pathStrength`, `minSpeed`, `maxSpeed`

EMPTY

Syntaxe

```
EMPTY
```

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` :

```
member("Notice").text = EMPTY
```

emulateMultiButtonMouse

Syntaxe

```
the emulateMultiButtonMouse
```

Description

Propriété système ; 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`).

Les clics du bouton droit n'ont aucun équivalent direct sur le 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

L'instruction suivante vérifie si l'animation est lue sur un Macintosh et, le cas échéant, donne à la propriété `emulateMultiButtonMouse` la valeur `TRUE` :

```
if the platform contains "Macintosh" then the emulateMultiButtonMouse = TRUE
```

Voir aussi

`keyPressed()`, `rightMouseDown (propriété système)`, `rightMouseUp (propriété système)`

enabled

Syntaxe

```
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.

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)

Syntaxe

```
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)

Syntaxe

```
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)

Syntaxe

```
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`

enableHotSpot

Syntaxe

```
enableHotSpot(sprite quelleImageObjetQTVR, idDeZoneRéférencée, trueOuFalse)
```

Description

Commande QTVR ; détermine si la zone référencée spécifiée pour l'image-objet QTVR spécifiée est activée (`TRUE`) ou désactivée (`FALSE`).

end

Syntaxe

```
end
```

Description

Mot-clé ; marque la fin des gestionnaires et des structures de contrôle à plusieurs lignes.

end case

Syntaxe

```
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 the key
    of "A": go to frame "Pomme"
    of "B", "C" :
      puppetTransition 99
      go to frame "Mangue"
    otherwise beep
  end case
end keyDown
```

Voir aussi

`case`

endAngle

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).\
  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 demi arc de 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

Syntaxe

`the endColor of member` *`quelActeur`*

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_Examples, lui-même dans le dossier de Director.

Voir aussi

`color()`, `fillColor`, `fillMode`

endFrame()

Syntaxe

`sprite(`*`quelleImageObjet`*`).endFrame`

Description

Fonction ; renvoie le numéro de la dernière image de l'étendue de l'image-objet.

Cette fonction est pratique pour déterminer l'étendue d'une image-objet particulière dans le scénario. Cette fonction est uniquement disponible dans une image contenant l'image-objet. Elle ne peut pas être appliquée aux images-objets situées dans des images différentes de l'animation et sa valeur ne peut pas être définie.

Exemple

L'instruction suivante indique l'image finale de l'image-objet de la piste 5 dans la fenêtre Messages :

```
put sprite(5).endFrame
```

Voir aussi

`startFrame`

endRecording

Syntaxe

```
endRecording
```

Description

Mot-clé ; termine une session de mise à jour du scénario. Vous pouvez reprendre le contrôle des pistes du scénario en créant des esclaves après l'émission du mot-clé `endRecording`.

Exemple

Lorsque utilisé dans le gestionnaire suivant, le mot-clé `endRecording` termine la session de création de scénario :

```
on animBalle nombreDimages
  beginRecording
    horizontal = 0
    vertical = 100
    repeat with i = 1 to nombreDimages
      go to frame i
      sprite(20).member = member "Balle"
      sprite(20).locH = horizontal
      sprite(20).locV = vertical
      horizontal = horizontal + 3
      vertical = vertical + 2
      updateFrame
    end repeat
  endRecording
end
```

Voir aussi

`beginRecording`, `scriptNum`, `tweened`, `updateFrame`

end repeat

Consultez

`repeat while`, `repeat with`, `repeat with...in list`, `repeat with...down to`

on endSprite

Syntaxe

```
on endSprite
  instruction(s)
end
```

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 au delà de l'image.

Les commandes `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 :

```
on endSprite me
  -- nettoyage
  gNombreDeRequins = gNombreDeRequins - 1
  puppetSound(5,0)
end
```

Voir aussi

`on beginSprite`, `on exitFrame`

endTellTarget()

Consultez `tellTarget()`.

endTime

Syntaxe

```
sound(numéroDePiste).endTime
the endTime of sound numéroDePiste
```

Description

Cette propriété constitue la position temporelle de fin du son en cours, en pause ou en file d'attente. 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 commandes `queue()` ou `setPlayList()`.

Exemple

L'instruction Lingo suivante vérifie si l'acteur son Annonce est défini pour une lecture sans interruption dans la piste audio 1 :

```
if sound(1).startTime > 0 and sound(1).endTime < member("Annonce").duration
  then
  alert "Lecture d'un son incomplet"
end if
```

Voir aussi

`setPlaylist()`, `queue()`, `startTime`

ENTER

Syntaxe

```
ENTER
```

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

L'instruction suivante vérifie si la touche Entrée est enfoncée et, le cas échéant, envoie la tête de lecture sur l'image ajouterLaSomme :

```
on keyDown
  if the key = ENTER then go to frame "ajouterLaSomme"
end
```

Voir aussi

`RETURN (constante)`

on enterFrame

Syntaxe

```
on enterFrame
  instruction(s)
end
```

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 :

```
on enterFrame
  repeat with i = 1 to 5
    puppetSprite i, FALSE
  end repeat
end
```

environment

Syntaxe

```
the environment
```

```
the environment.nomDePropriété
```

Description

Propriété système ; cette propriété contient une liste d'informations concernant l'environnement dans lequel le contenu Director est lu.

Ceci permet à Macromedia d'ajouter ultérieurement des informations à la propriété `environment`, 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), `Plugin` (module de navigateur web) ou `Java Applet` (applet Java). 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 :

```
put the environment
-- [#shockMachine: 0, #shockMachineVersion: "", #platform:
   "Macintosh,PowerPC", #runMode: "Author", #colorDepth: 32,
   #internetConnected: #online, #uiLanguage: "English", #osLanguage: "English",
   #productBuildVersion: "151"]
```

Voir aussi

`colorDepth`, `platform`, `runMode`

erase member

Syntaxe

```
member(quelActeur).erase()
erase member quelActeur
```

Description

Commande ; supprime l'acteur spécifié et laisse son emplacement vide dans la fenêtre Distribution.

Pour de meilleurs résultats, utilisez cette commande pendant la programmation et non dans les projections, car cela peut entraîner des problèmes de mémoire.

Exemples

L'instruction suivante supprime l'acteur Engrenage de la distribution Matériel :

```
member("Engrenage", "Matériel").erase()
```

Le gestionnaire suivant supprime les acteurs du début à la fin :

```
on effacerLesActeurs start, finish
  repeat with i = start to finish
    member(i).erase()
  end repeat
end on effacerLesActeurs
```

Voir aussi

`new()`

error

Syntaxe

```
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 se combinent 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`

on EvalScript

Syntaxe

```
on EvalScript unParamètre
  instruction(s)
end
```

Description

Message système et gestionnaire d'événement ; dans une animation Shockwave, contient des instructions exécutées lorsque le gestionnaire reçoit un message `EvalScript` du 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.

  Le lecteur Director pour Java ne supporte pas le gestionnaire `on EvalScript`. Pour activer la communication entre une applet et un navigateur, utilisez Java, JavaScript ou VBScript.

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.

Exemples

L'exemple suivant indique comment faire passer la tête de lecture à une image spécifique en fonction de l'image passée comme paramètre :

```
on EvalScript unParamètre
  go frame unParamètre
end
```

Le gestionnaire suivant exécute l'instruction `go frame unParamètre` s'il reçoit un message `EvalScript` contenant les arguments `chien`, `chat` ou `arbre` :

```
on EvalScript unParamètre
  case unParamètre of
    "chien", "chat", "arbre": go frame unParamètre
  end case
end
```

Une instruction d'appel possible dans JavaScript serait `EvalScript ("chien")`.

Le gestionnaire suivant prend un argument qui pourrait être un nombre ou un symbole :

```
on EvalScript unParamètre
  if word 1 of unParamètre = "monGestionnaire" then
    do unParamètre
  end if
end
```

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` :

```
on monGestionnaire unParamètre
  go to frame string(unParamètre)
end
```

Voir aussi

`externalEvent`, `return (mot-clé)`

eventPassMode

Syntaxe

```
sprite(quelleImageObjetFlash).eventPassMode
the eventPassMode of sprite quelleImageObjetFlash
member(quelActeurFlash).eventPassMode
the eventPassMode of member quelActeurFlash
```

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.

```
on enterFrame
  if sprite(5).buttonsEnabled = TRUE then
    sprite(5).eventPassMode= #passNotButton
  else
    sprite(5).eventPassMode = #passAlways
  end if
end
```

exit

Syntaxe

```
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 the colorDepth = 1 then exit
  sprite(1).foreColor = 35
end
```

Voir aussi

`abort`, `halt`, `quit`, `pass`, `return (mot-clé)`

on exitFrame

Syntaxe

```
on exitFrame
  instruction(s)
end
```

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 Lingo 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`.

Exemples

Le gestionnaire suivant désactive toutes les conditions d'asservissement lorsque la tête de lecture quitte l'image :

```
on exitFrame me
  repeat with i = 48 down to 1
    sprite(i).puppet = FALSE
  end repeat
end
```

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 :

```
global vTotal

on exitFrame
  if vTotal > 1000 then go to frame "Terminé"
end
```

Voir aussi

`on enterFrame`

exitLock

Syntaxe

```
the 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`).

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).

Cette propriété peut être testée et définie.

Exemples

L'instruction suivante donne à la propriété `exitLock` la valeur `TRUE` :

```
set the exitLock to 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 :

```
on checkExit
  if the commandDown and (the key = "." or the key = "q") and the exitLock =
  TRUE then go to frame "séquence de sortie"
end
```

exit repeat

Syntaxe

```
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`

exp()

Syntaxe

```
(entierOuVirguleFlottante).exp
exp(entierOuVirguleFlottante)
```

Description

Fonction ; calcule e, la base logarithmique naturelle, à la puissance spécifiée par *`entierOuVirguleFlottante`*.

Exemple

L'instruction suivante calcule la valeur de e à la puissance 5 :

```
put (5).exp
-- 148.4132
```

externalEvent

Syntaxe

```
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. La chaîne envoyée par `externalEvent` doit être dans un langage script supporté par le navigateur.

Cette commande fonctionne uniquement pour les animations dans des navigateurs. Pour activer la communication entre une applet et un navigateur, utilisez Java, JavaScript ou VBScript.

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.

Exemples

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`

externalParamCount()

Syntaxe

`externalParamCount()`

Description

Fonction ; renvoie le nombre de paramètres passés par une balise HTML <EMBED> ou <OBJECT> à une animation Shockwave.

Cette fonction est valide uniquement pour les animations Shockwave exécutées dans un navigateur web. Elle ne fonctionne pas pour les animations en cours de création ou dans des projections.

Exemple

Le gestionnaire suivant détermine si une balise <OBJECT> ou <EMBED> passe des paramètres externes à une animation Shockwave et exécute des instructions Lingo si les paramètres sont passés :

```
if externalParamCount() > 0 then
  -- une action quelconque
end if
```

Voir aussi

`externalParamName()`, `externalParamValue()`

externalParamName()

Syntaxe

`externalParamName(`*n*`)`

Description

Fonction ; renvoie le nom d'un paramètre spécifique dans la liste de paramètres externes d'une balise HTML <EMBED> ou <OBJECT>. Cette fonction est valide uniquement pour les animations Shockwave exécutées dans un navigateur web. Elle ne peut pas être utilisée avec des animations Director ou des projections.

- Si *n* est un nombre entier, `externalParamName` renvoie le *n*ième nom de paramètre dans la liste.
- Si *n* est une chaîne, `externalParamName` renvoie `n` si un des noms de paramètre externe correspond à *n*. La correspondance n'est pas sensible à la hauteur de casse. Si aucun nom de paramètre correspondant n'est trouvé, `externalParamName` renvoie `VOID`.

Exemple

L'instruction suivante place la valeur d'un paramètre externe donné dans la variable *maVariable* :

```
if externalParamName ("chaîneURLsw") = "chaîneURLsw" then
  maVariable = externalParamValue ("chaîneURLsw")
end if
```

Voir aussi

externalParamCount(), externalParamValue()

externalParamValue()

Syntaxe

externalParamValue(*n*)

Description

Fonction ; renvoie une valeur spécifique de la liste des paramètres externes d'une balise HTML <EMBED> ou <OBJECT>. Cette fonction est valide uniquement pour les animations Shockwave exécutées dans un navigateur web. Elle ne peut pas être utilisée avec des animations exécutées dans l'environnement auteur de Director ou dans des projections.

- Si *n* est un nombre entier, externalParamValue renvoie la *é*n*i*ème valeur de paramètre de la liste des paramètres externes.
- Si *n* est une chaîne, externalParamValue renvoie la valeur associée au premier nom qui correspond à *n*. La correspondance n'est pas sensible à la hauteur de casse. Si aucune valeur de paramètre correspondante n'existe, externalParamValue renvoie VOID.

Le comportement de cette fonction dans une applet diffère de son comportement dans d'autres animations Director. Dans une applet, externalParamValue a le résultat suivant :

- Renvoie les paramètres de l'applet au lieu des paramètres de la balise <EMBED>.
- Accepte uniquement les paramètres de chaîne.
- Renvoie une chaîne de longueur nulle plutôt que VOID.

Consultez Paramètres des balises EMBED et OBJECT et Paramètres accessibles avec Lingo sur le site du centre des développeurs Director.

Exemple

L'instruction suivante place la valeur d'un paramètre externe dans la variable *maVariable* :

```
if externalParamName ("chaîneURLsw") = "chaîneURLsw" then
  maVariable = externalParamValue ("chaîneURLsw")
end if
```

Voir aussi

externalParamCount(), externalParamName()

extractAlpha()

Syntaxe

```
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()
```

Voir aussi

setAlpha(), useAlpha

extrude3D

Syntaxe

```
member(quelActeurTexte).extrude3D(member(quelActeur3D))
```

Description

Commande 3D ; crée une ressource de modèle #extruder dans l'acteur 3D *quelActeur3D* à partir du texte *quelActeurTexte*.

Remarquez qu'il ne s'agit pas de la même chose que l'utilisation de 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)
```

Exemple

Dans l'exemple suivant, Logo est un acteur texte et Séquence est un acteur 3D. La première ligne crée une ressource de modèle dans Séquence, 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("Séquence"))
member("Séquence").newModel("Logo3D", maRessourceDeModèleTexte)
```

Voir aussi

bevelDepth, bevelType, displayFace, smoothness, tunnelDepth, displayMode

face

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).\
  face.count
member(quelActeur).modelResource(quelleRessDeMod).\
  face[index].colors
member(quelActeur).modelResource(quelleRessDeMod).\
  face[index].normals
member(quelActeur).modelResource(quelleRessDeMod).\
  face[index].shader
member(quelActeur).modelResource(quelleRessDeMod).\
  face[index].textureCoordinates
member(quelActeur).modelResource(quelleRessDeMod).\
  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[indexDeFace]
member(quelActeur).model(quelModèle).meshdeform.\
  mesh[indexDeMaille].face[indexDeFace].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[ ]

Syntaxe

```
member(quelActeur).model(quelModèle).meshdeform.\
   mesh[indexDeMaille].face[indexDeFace]
```

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`

fadeIn()

Syntaxe

```
sound(numéroDePiste).fadeIn({millisecondes})
fadeIn(sound(numéroDePiste) {, millisecondes })
```

Description

Cette fonction définit immédiatement le `volume` de la piste audio *`numéroDePiste`* sur zéro pour le ramener ensuite sur le volume courant en fonction du nombre de millisecondes défini. La valeur par défaut est 1 000 millisecondes (1 seconde).

Le paramètre de balance courant est conservé pour l'ensemble du fondu.

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 :

```
sound(3).play(member("intro2"))
sound(3).fadeIn(3000)
```

Voir aussi

`fadeOut()`, `fadeTo()`, `pan (propriété audio)`, `volume (piste audio)`

fadeOut()

Syntaxe

```
sound(numéroDePiste).fadeOut({millisecondes})
fadeOut(sound(numéroDePiste) {, millisecondes })
```

Description

Cette fonction réduit progressivement le `volume` de la piste audio *`numéroDePiste`* jusque zéro en fonction du nombre donné de millisecondes, ou de 1000 millisecondes (1 seconde) si aucune valeur n'est définie.

Le paramètre de balance courant est conservé pour l'ensemble du fondu.

Exemple

L'instruction suivante diminue le son de la piste 3 sur 5 secondes :

```
sound(3).fadeOut(5000)
```

Voir aussi

`fadeIn()`, `fadeTo()`, `pan (propriété audio)`, `volume (piste audio)`

fadeTo()

Syntaxe

```
sound(numéroDePiste).fadeTo(volume {, millisecondes })
fadeTo(sound(numéroDePiste), volume {, millisecondes })
```

Description

Cette fonction change progressivement le `volume` de la piste audio `numéroDePiste` jusqu'au volume spécifié en fonction du nombre donné de millisecondes, ou de 1000 millisecondes (1 seconde) si aucune valeur n'est définie. La plage de valeurs du volume s'étend de 0 à 255.

Le paramètre de balance courant est conservé pour l'ensemble du fondu.

Vous pourrez voir un exemple de `fadeTo()` dans une animation en consultant l'animation Sound Control du dossier Learning/Lingo_Examples, lui-même dans le dossier de Director.

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.

```
sound(4).fadeTo(150, 2000)
```

Voir aussi

`fadeIn()`, `fadeOut()`, `pan (propriété audio)`, `volume (piste audio)`

FALSE

Syntaxe

```
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` :

```
the soundEnabled = FALSE
```

Voir aussi

`if`, `not`, `TRUE`

far (brouillard)

Syntaxe

```
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)`

field

Syntaxe

```
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.

Exemples

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 go to "devisBureau"
```

Voir aussi

`char...of`, `item...of`, `line...of`, `word...of`

fieldOfView

Syntaxe

```
sprite(quelleImageObjetQTVR).fieldOfView
the fieldOfView of sprite quelleImageObjetQTVR
```

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)

Syntaxe

```
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 rect 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`

fileName (propriété de distribution)

Syntaxe

```
castLib(quelleDistribution).fileName
the fileName of castLib quelleDistribution
```

Description

Propriété ; spécifie le nom de fichier de la distribution spécifiée.

- Pour une distribution externe, `fileName` donne le chemin d'accès complet et le nom de la distribution.
- Pour une distribution interne, la propriété `fileName castLib` dépend de la distribution interne spécifiée. Pour la première distribution interne, la propriété `fileName castLib` spécifie le nom de l'animation. Pour le reste des distribution internes, `fileName` est une chaîne vide.

La propriété `fileName of castLib` accepte les URL comme références. Cependant, pour utiliser une distribution depuis Internet et minimiser la durée de téléchargement, utilisez la commande `downloadNetThing` ou `preloadNetThing` pour télécharger le fichier de la distribution sur un disque local, puis définissez `fileName castLib` 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 de projection.

Cette propriété peut être testée et définie pour les distributions externes. Elle peut être testée uniquement pour les distributions internes.

Remarque Director pour Java ne supporte pas la commande `downloadNetThing`.

Exemples

L'instruction suivante affiche le chemin d'accès et le nom de fichier de la distribution externe Boutons dans la fenêtre Messages :

```
put castLib("Boutons").fileName
```

L'instruction suivante affecte le nom de fichier Contenu.cst à la distribution externe Boutons :

```
castLib("Boutons").fileName = the moviePath & "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 :

```
downLoadNetThing("http://www.cbDeMille.com Milliers.cst", the  \
  applicationPath&"Milliers.cst")
castLib("Distribution").fileName = the applicationPath & "Milliers.cst"
```

Voir aussi

`downloadNetThing`, `preloadNetThing()`

fileName (propriété d'acteur)

Syntaxe

```
member(quelActeur).fileName
the fileName of member quelActeur
```

Description

Propriété d'acteur ; fait référence au nom du fichier affecté à l'acteur lié spécifié par *quelActeur*. 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.

La propriété d'acteur `fileName` accepte les URL comme références. Cependant, pour utiliser un fichier d'une URL et minimiser la durée de téléchargement, utilisez la commande `downloadNetThing` ou `preloadNetThing` pour télécharger le fichier sur un disque local, puis définissez la propriété d'acteur `fileName` comme fichier sur le disque local.

Le lecteur Director pour Java ne supportant pas la commande `downLoadNetThing`, le lecteur ne peut pas télécharger des fichiers en tâche de fond avant d'affecter un nouveau fichier à un acteur. La modification de la propriété d'acteur `fileName` dans une animation lue comme applet peut forcer l'applet à attendre la fin du téléchargement du nouveau fichier.

Cette propriété peut être testée et définie. 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 :

```
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 :

```
downLoadNetThing("http://www.cbDeMille.com/CinémaParlant.AIF",the  \
  applicationPath&"CinémaParlant.AIF")
member("Norma").fileName = the applicationPath & "FilmsAvecSon.AIF"
```

Voir aussi

`downloadNetThing`, `preloadNetThing()`

fileName (propriété de fenêtre)

Syntaxe

```
window quelleFenêtre.fileName
the fileName of window quelleFenêtre
```

Description

Propriété de fenêtre ; fait référence au nom de fichier de l'animation affectée à la fenêtre spécifiée par *quelleFenêtre*. 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 de fenêtre `fileName` au nom de fichier de l'animation.

La propriété `fileName of window` accepte les URL comme références. Cependant, pour utiliser un fichier d'animation depuis une URL et minimiser la durée de téléchargement, utilisez la commande `downloadNetThing` ou `preloadNetThing` pour télécharger le fichier d'animation sur un disque local, puis définissez la propriété de fenêtre `fileName` comme le fichier sur le disque local.

Cette propriété peut être testée et définie.

Exemples

L'instruction suivante affecte le fichier appelé Tableau de commande à la fenêtre Boîte à outils :

```
window("Boîte à outils").fileName = "Tableau de commande"
```

L'instruction suivante affiche le nom de fichier affecté à la fenêtre intitulée Navigateur :

```
put 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 :

```
downLoadNetThing("http://www.cbDeMille.com/Finale.DIR",the  \
  applicationPath&"Finale.DIR")
window("Mon gros plan").fileName = the applicationPath&"Finale.DIR"
```

Voir aussi

`downloadNetThing`, `preloadNetThing()`

fill()

Syntaxe

```
objetImage.fill(gauche, haut, droite, bas, objetCouleurOuListedeParamètres)
objetImage.fill(point(x, y), point(x, y), objetCouleurOuListedeParamètres)
objetImage.fill(rect, objetCouleurOuListedeParamètres)
```

Description

Cette fonction remplit une région rectangulaire avec la couleur *objetCouleur* dans l'objet image donné. Vous pouvez définir le rectangle par l'une des trois méthodes présentées. Les points spécifiés le sont par rapport au coin supérieur gauche de l'objet image donné. La valeur renvoyée est 1 si aucune erreur n'est détectée, et zéro en cas d'erreur.

Si vous fournissez une *listeDeParamètres* plutôt qu'un simple *objetCouleur*, le rectangle est rempli avec une forme définie à l'aide de ces paramètres :

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 remplissage de la forme.
#bgColor	Un objet couleur, qui détermine la couleur de la bordure de la forme.

Pour optimiser les performances avec des images 8 bits (ou une valeur inférieure), l'*objetCouleur* doit contenir une valeur de couleur indexée. Pour les images 16 ou 32 bits, utilisez une valeur de couleur rvb.

Exemples

L'instruction suivante rend l'objet image dans la variable *monImage* complètement noir :

```
monImage.fill(monImage.rect, rgb(0, 0, 0))
```

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.

```
imageTest.fill(0, 0, 100, 100, [#shapeType: #oval, #lineSize: 5, #color:
   rgb(0, 255, 0), \
#bgColor: rgb(255, 0, 0)])
```

Voir aussi

`color()`, `draw()`

fillColor

Syntaxe

`member(`*`quelActeur`*`).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_Examples, 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 :

```
member("Archie").fillColor = rgb( 24, 15, 153)
```

Voir aussi

`endColor`, `fillMode`

fillCycles

Syntaxe

`member(`*`quelActeur`*`).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 valide 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_Examples, lui-même dans le dossier de Director.

Exemple

L'instruction suivante donne à `fillCycles` de l'acteur Archie la valeur 3 :

```
member("Archie").fillCycles = 3
```

Voir aussi

`endColor`, `fillColor`, `fillMode`

fillDirection

Syntaxe

```
member(quelActeur).fillDirection
```

Description

Propriété d'acteur forme vectorielle ; spécifie le degré de rotation du remplissage de la forme.

Cette propriété n'est valide 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_Examples, lui-même dans le dossier de Director.

Voir aussi

`fillMode`

filled

Syntaxe

```
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("Cible").filled = TRUE
member("Cible").pattern = 1
```

Voir aussi

`fillColor`, `fillMode`

fillMode

Syntaxe

```
member(quelActeur).fillMode
```

Description

Propriété d'acteur forme vectorielle ; indique la méthode de remplissage pour 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_Examples, lui-même dans le dossier de Director.

Exemple

L'instruction suivante donne à `fillMode` de l'acteur Archie la valeur de dégradé :

```
member("Archie").fillMode = #gradient
```

Voir aussi

`endColor`, `fillColor`

fillOffset

Syntaxe

```
member(quelActeur).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 valide 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_Examples, 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 :

```
member("Miette").fillOffset = point(33, 27)
```

Voir aussi

`defaultRect`, `fillMode`

fillScale

Syntaxe

```
member(quelActeur).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 valide 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_Examples, lui-même dans le dossier de Director.

Exemple

L'instruction suivante donne à `fillScale` de l'acteur Archie la valeur 33 :

```
member("Archie").fillScale = 33.00
```

Voir aussi

`fillMode`

findEmpty()

Syntaxe

```
findEmpty(member quelActeur)
```

Description

Fonction ; pour la distribution courante uniquement, affiche la position d'acteur vide suivante ou la position suivant l'acteur spécifié par *quelActeur*.

Exemple

L'instruction suivante recherche le premier acteur vide à partir de l'acteur 100 :

```
put findEmpty(member 100)
```

findLabel()

Syntaxe

```
sprite(quelleImageObjetFlash).findLabel(quelNomDétiquette)
findLabel(sprite quelleImageObjetFlash, quelNomDétiquette)
```

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'est pas encore transférée en mémoire.

findPos

Syntaxe

```
liste.findPos(propriété)
findPos(liste, propriété)
```

Description

Commande de liste ; identifie la position de la propriété spécifiée par *propriété* dans la liste de propriétés spécifiée par *liste*.

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.

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

Syntaxe

`listeTriée``.findPosNear(`*`valeurOuPropriété`*`)`
`findPosNear(`*`listeTriée`*`, `*`valeurOuPropriété`*`)`

Description

Commande de liste ; pour les listes triées uniquement, identifie la position de l'élément spécifié par *`valeurOuPropriété`* dans la liste triée indiquée.

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.

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

Syntaxe

`finishIdleLoad` *`baliseDeChargement`*

Description

Commande ; force le chargement de tous les acteurs possédant la balise de chargement spécifiée.

Exemple

L'instruction suivante termine le chargement de tous les acteurs possédant la balise de chargement 20 :

```
finishIdleLoad 20
```

Voir aussi

`idleHandlerPeriod`, `idleLoadDone()`, `idleLoadMode`, `idleLoadPeriod`, `idleLoadTag`, `idleReadChunkSize`

firstIndent

Syntaxe

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 :

```
member("Bureau").firstIndent = 0
```

Voir aussi

leftIndent, rightIndent

fixedLineSpace

Syntaxe

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 :

```
member("Bureau").fixedLineSpace = 24
```

fixedRate

Syntaxe

```
sprite(quelleImageObjetFlashOuGIF).fixedRate
the fixedRate of sprite quelleImageObjetFlashOuGIF
member(quelleImageObjetFlashOuGIF).fixedRate
the fixedRate of member quelleImageObjetFlashOuGIF
```

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.

```
on ajustementDeLaCadenceFixée quelleImageObjet, typeDajustement, Decombien
  case typeDajustement of
    #accéleration:
      sprite(quelleImObj).fixedRate = sprite(quelleImObj).fixedRate + combien
    #décélération:
      sprite(quelleImObj).fixedRate = sprite(quelleImObj).fixedRate - combien
  end case
end
```

Voir aussi

`playBackMode`

fixStageSize

Syntaxe

```
the 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.

La propriété `fixStageSize` ne peut pas changer la taille de la scène d'une animation en cours de lecture.

Cette propriété peut être testée et définie.

Exemples

L'instruction suivante détermine si la propriété `fixStageSize` est activée. Si `fixStageSize` est `FALSE`, elle envoie la tête de lecture vers une image spécifiée.

```
if the fixStageSize = FALSE then go to frame "taille correcte"
```

L'instruction suivante donne à la propriété `fixStageSize` la valeur opposée au paramètre courant :

```
the fixStageSize = not the fixStageSize
```

Voir aussi

`centerStage`

flashRect

Syntaxe

```
member(quelActeurVecteurOuFlash).flashRect
the flashRect of member quelActeurVecteurOuFlash
```

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 :

```
on beginSprite me
  sprite(me.spriteNum).rect = sprite(me.spriteNum).member.FlashRect
end
```

Voir aussi

`defaultRect`, `defaultRectMode`, `state (Flash, SWA)`

flashToStage()

Syntaxe

```
sprite(quelleImageObjetFlash).flashToStage(pointDeLanimationFlash)
flashToStage (sprite quelleImageObjetFlash, 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.

Exemple

Le gestionnaire suivant accepte une valeur de point et une référence d'image-objet comme paramètre, puis donne à la coordonnée supérieure gauche de l'image-objet spécifiée le point indiqué dans une image-objet d'animation Flash de la piste 10 :

```
on déplacImObj quelPointFlash, quelleImageObjet
  sprite(quelleImageObjet).loc =
  sprite(1).FlashToStage(pointDeLanimationFlash)
  updatestage
end
```

Voir aussi

`stageToFlash()`

flat

Syntaxe

```
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

Syntaxe

```
sprite(quelNuméroDimageObjet).flipH
the flipH of sprite quelNuméroDimageObjet
```

Description

Propriété d'image-objet ; indique si l'image d'une image-objet a été renversée horizontalement sur la scène (TRUE) ou non (FALSE).

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 :

```
put sprite (5).flipH
```

Voir aussi

flipV, rotation, skew

flipV

Syntaxe

```
sprite(quelNuméroDimageObjet).flipV
the flipV of sprite quelNuméroImageObjet
```

Description

Propriété d'image-objet ; indique si l'image d'une image-objet a été renversée verticalement sur la scène (`TRUE`) ou non (`FALSE`).

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 :

```
sprite (5).flipV = 1
```

Voir aussi

`flipH`, `rotation`, `skew`

float()

Syntaxe

```
(expression).float
float (expression)
```

Description

Fonction ; convertit une expression en 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`.

Exemples

L'instruction suivante convertit le nombre entier 1 en 1 à virgule flottante :

```
put (1).float
-- 1.0
```

Les opérations mathématiques peuvent être réalisées à l'aide de `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()

Syntaxe

```
(expression).floatP
floatP(expression)
```

Description

Fonction ; indique si la valeur spécifiée par *expression* est un nombre à virgule flottante (1 ou TRUE) ou non (0 ou FALSE).

Le *P* dans floatP signifie *prédicat*.

Exemples

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()

floatPrecision

Syntaxe

```
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.

Exemples

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
```

flushInputEvents

Syntaxe

```
flushInputEvents()
```

Description

Cette commande purge les événements clavier ou souris de la file d'attente de Director. Cette commande est pratique lorsque Lingo exécute une boucle de répétition 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 commande n'est effective qu'en cours d'exécution et n'a aucun effet pendant la programmation.

Exemple

L'instruction Lingo suivante désactive les événements souris et clavier lors de l'exécution d'une boucle de répétition :

```
repeat with i = 1 to 10000
  flushInputEvents()
  sprite(1).loc = sprite(1).loc + point(1, 1)
end repeat
```

Voir aussi

mouseDown, mouseUp, on keyDown, on keyUp

fog

Syntaxe

```
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.

La section « Voir aussi » de cette entrée contient une liste complète des propriétés de brouillard. Pour plus d'informations, consultez les entrées des différentes propriétés.

Voir aussi

color (brouillard), decayMode, enabled (brouillard), far (brouillard), near (brouillard)

font

Syntaxe

```
member(quelActeur).font
the font of member quelActeur
```

Description

Propriété d'acteur texte et champ ; détermine 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.

Le lecteur Director pour Java ne fait pas correspondre les polices lors de la conversion d'une animation ; Java substitue la police par défaut aux polices qui ne sont pas supportées. Utilisez uniquement les polices supportées par Java comme valeurs pour la propriété d'acteur `font` dans une animation lue comme applet.

Seules les polices suivantes offertes par Java sont compatibles entre plates-formes :

Nom de police Java	Police Windows correspondante	Police Macintosh correspondante
Helvetica	Arial	Helvetica
TimesRoman	Times New Roman	Times
Courier	Courier-New	Courier
Dialog	MS Sans Serif	Chicago ou Charcoal
DialogInput	MS Sans Serif	Geneva
ZapfDingbats	WingDings	Zapf Dingbats
Valeur par défaut	Arial	Helvetica

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_Examples, 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 :

```
anciennePolice = member("Pierre").font
```

Voir aussi

`text`, `alignment`, `fontSize`, `fontStyle`, `lineHeight (propriété d'acteur)`

fontSize

Syntaxe

```
member(quelActeur).fontSize
the fontSize of member quelActeur
```

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_Examples, lui-même dans le dossier de Director.

Exemples

L'instruction suivante donne à la variable *`ancienneTaille`* la valeur `fontSize of member` courante de l'acteur champ Pierre :

```
ancienneTaille = member("Pierre").fontSize
```

L'instruction suivante définit la troisième ligne de l'acteur texte monMenu sur 12 points :

```
member("monMenu").fontSize = 12
```

Voir aussi

`text`, `alignment`, `font`, `fontStyle`, `lineHeight (propriété d'acteur)`

fontStyle

Syntaxe

```
member(quelActeur).fontStyle
the fontStyle of member quelActeur
member(quelActeur).char[quelCaractère].fontStyle
the fontStyle of char quelCaractère
member(quelActeur).word[quelMot].fontStyle
the fontStyle of word quelMot
member(quelActeur).line[quelleLigne].fontStyle
the fontStyle of line quelleLigne
```

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 le Macintosh.

Utilisez le style normal pour supprimer tous les styles déjà appliqués. Le paramètre *`quelActeur`* peut être un nom ou un numéro d'acteur.

Pour une animation lue comme applet, normal, gras et italique sont les seules styles valides pour la propriété d'acteur `fontStyle`. Le lecteur Director pour Java ne supporte pas les styles de police souligné, ombré, étendu ou condensé.

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_Examples, lui-même dans le dossier de Director.

Exemples

L'instruction suivante donne à la variable *`ancienStyle`* la valeur de la propriété `fontStyle` courante de l'acteur champ Pierre :

```
ancienStyle = member("Pierre").fontStyle
```

L'instruction suivante affecte les valeurs gras et italique à la propriété d'acteur `fontStyle` de l'acteur champ Poème :

```
member("Poème").fontStyle = [#bold, #italic]
```

L'instruction suivante affecte la valeur italique à la propriété `fontStyle` du troisième nom de l'acteur champ Noms :

```
member("Noms").word[3].fontStyle = [#italic]
```

L'instruction suivante affecte la valeur gras et italique à la propriété `fontStyle` des mots 1 à 4 de l'acteur texte mesNotes :

```
member("mesNotes").word[1..4].fontstyle = [#bold, #italic]
```

Voir aussi

`text`, `alignment`, `fontSize`, `font`, `lineHeight (propriété d'acteur)`

foreColor

Syntaxe

```
member(nomDacteur).foreColor = numéroDeCouleur
set the foreColor of member nomDacteur to numéroDeCouleur
sprite quelleImageObjet.foreColor
the foreColor of sprite quelleImageObjet
```

Description

Propriété d'acteur ; définit la couleur de premier plan d'un acteur champ.

Pour une animation lue comme applet, indiquez les couleurs de la propriété d'image-objet `foreColor` sous la forme d'équivalent décimal des valeurs hexadécimales 24 bits utilisées dans un document HTML.

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`.

Exemples

La valeur hexadécimale du rouge pur, FF0000, est l'équivalent de 16711680 en nombres décimaux. L'instruction suivante spécifie le rouge pur comme couleur de premier plan de l'acteur :

```
member(20).foreColor = 16711680
```

L'instruction suivante remplace la couleur de l'acteur champ 1 par la couleur numéro 250 de la palette :

```
member(1).foreColor = 250
```

L'instruction suivante donne à la variable *ancienneCouleur* la couleur de premier plan de l'image-objet 5 :

```
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 :

```
sprite(10 + random(3)).foreColor = 36
```

L'instruction suivante définit la propriété `foreColor` du troisième mot de la deuxième ligne de l'acteur texte maDescription sur une valeur de 27 :

```
member("maDescription").line[2].word[3].forecolor = 27
```

Voir aussi

`backColor`, `color (propriété d'image-objet et d'acteur)`

forget()

Syntaxe

```
timeout("nomDeTemporisation").forget()
forget(timeout("nomDeTemporisation"))
```

Description

Cette fonction d'objet de temporisation supprime l'*`objetDeTemporisation`* donné de la liste `timeoutList` et l'empêche d'envoyer d'autres événements de temporisation.

Exemple

L'instruction suivante supprime l'objet de temporisation Réveil de la liste `timeoutList` :

```
timeout("Réveil").forget()
```

Voir aussi

`timeout()`, `timeoutHandler`, `timeoutList`, `new()`

forget

Syntaxe

```
window(quelleFenêtre).forget()
forget window quelleFenêtre
```

Description

Propriété de fenêtre ; indique à Lingo de fermer et supprimer la fenêtre spécifiée par *`quelleFenêtre`* lorsqu'elle n'est plus utilisée et qu'aucune autre variable n'y fait référence.

Lorsqu'une commande `forget window` est donnée, la fenêtre et l'animation dans une fenêtre disparaissent sans appeler les gestionnaires `on stopMovie`, `on closeWindow` ou `on deactivateWindow`.

S'il existe plusieurs références globales à l'animation dans une fenêtre, la fenêtre ne répond pas à la commande `forget`.

Exemple

L'instruction suivante indique à Lingo de supprimer la fenêtre Tableau de commande lorsque l'animation ne l'utilise plus :

```
window("Tableau de commande").forget()
```

Voir aussi

`close window`, `open window`

frame() (fonction)

Syntaxe

```
the frame
```

Description

Fonction ; renvoie le numéro de l'image courante de l'animation.

Exemple

L'instruction suivante envoie la tête de lecture à l'image qui précède l'image en cours :

```
go to (the frame - 1)
```

Voir aussi

go, label(), marker()

frame (propriété d'image-objet)

Syntaxe

```
sprite(quelleImageObjetFlash).frame
the frame of sprite quelleImageObjetFlash
```

Description

Propriété d'image-objet ; contrôle l'image affichée de l'animation Flash. La valeur par défaut est 1.

Cette propriété peut être testée et définie.

Exemple

Le script d'image suivant vérifie si la lecture d'une animation Flash est terminée (en vérifiant si l'image courante correspond au nombre total des images de l'animation). Si l'animation n'est pas terminée, la tête de lecture continue son exécution en boucle dans l'image courante ; une fois l'animation terminée, la tête de lecture passe à l'image suivante. Ce script considère que l'animation était conçue pour se terminer sur sa dernière image et qu'elle n'a pas été définie pour une lecture en boucle.

```
on exitFrame
  if sprite(5).frame < sprite(5).member.frameCount then
    go to the frame
  end if
end
```

frameCount

Syntaxe

```
member(quelActeurFlash).frameCount
the frameCount of member quelActeurFlash
```

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 :

```
property spriteNum

on beginSprite me
  put ""L'animation Flash de la piste"" && spriteNum && contient"" &&
  sprite(spriteNum).member.frameCount && ""images.""
end
```

frameLabel

Syntaxe

```
the frameLabel
```

Description

Propriété d'image ; identifie le libellé affecté à l'image courante. Lorsque l'image courante n'a pas de libellé, la valeur de la propriété `frameLabel` est 0.

Cette propriété peut être testée à tout moment. Elle peut être définie au cours d'une session de création de scénario.

Exemple

L'instruction suivante vérifie le libellé de l'image courante. Dans ce cas, la valeur de `frameLabel` est Démarrer :

```
put the frameLabel
-- "Démarrer"
```

Voir aussi

`labelList`

framePalette

Syntaxe

```
the framePalette
```

Description

Propriété d'image ; 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.

Le navigateur contrôlant la palette pour la page web toute entière, le lecteur Director pour Java utilise toujours la palette du navigateur. Pour obtenir des couleurs fiables pour la création d'une animation destinée au lecteur Director pour Java, utilisez la palette par défaut du système auteur. Lorsqu'un contrôle exact des couleurs est nécessaire, utilisez Shockwave au lieu de Java.

Cette propriété peut être testée. Elle peut également être définie au cours d'une session de création du scénario.

Exemples

L'instruction suivante vérifie la palette utilisée dans l'image courante. Dans ce cas, la palette est l'acteur 45.

```
put the framePalette
-- 45
```

L'instruction suivante fait de l'acteur palette 45 la palette de l'image courante :

```
the framePalette = 45
```

Voir aussi

puppetPalette

frameRate

Syntaxe

```
member(quelActeur).frameRate
the frameRate of member quelActeur
```

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.

Exemples

L'instruction suivante fixe la cadence d'image de l'acteur vidéo numérique QuickTime Chaise pivotante à 30 images par seconde :

```
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 :

```
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.

```
property spriteNum

on beginSprite me
  if sprite(spriteNum).member.frameRate < 15 then
    sprite(spriteNum).playBackMode = #fixed
    sprite(spriteNum).fixedRate = 15
  end if
end
```

Voir aussi

`fixedRate`, `movieRate`, `movieTime`, `playBackMode`

frameReady()

Syntaxe

```
frameReady(imageN)
frameReady(imageN, imageZ)
frameReady()
frameReady(sprite quelleImageObjetFlash, NuméroDimage)
```

Description

Fonction ; dans le cas d'une animation Flash, détermine si une animation lue en flux continu est prête à être affichée. Si une quantité suffisante de l'image-objet a été transférée en mémoire pour pouvoir afficher l'image (nombre entier pour un numéro d'image, chaîne pour un libellé) spécifiée dans le paramètre `numéroDimage`, cette fonction a la valeur `TRUE` ; autrement, elle a la valeur `FALSE`. Dans le cas d'une animation Director, cette fonction détermine si tous les acteurs de *`imageN`* (le numéro de l'image) ont été téléchargés à partir d'Internet et sont disponibles localement.

Cette fonction n'est utile que lors de la lecture en flux continu d'une animation, d'une plage d'images, d'une distribution ou d'un acteur lié. Pour activer la lecture en flux continu, réglez les propriétés de lecture de l'animation du menu Modification de façon à utiliser les médias disponibles ou à afficher les repères d'emplacement.

Pour les animations Director, les projections et les animations Shockwave :

- `frameReady (`*`imageN`*`)` – Détermine si les acteurs de l'*`imageN`* ont été téléchargés.
- `frameReady (`*`imageN, imageZ`*`)` – Détermine si les acteurs des images *`imageN`* à *`imageZ`* ont été téléchargés.
- `frameReady()` – Détermine si les acteurs utilisés dans les images du scénario ont été téléchargés.

Pour voir comment la fonction `frameReady` est utilisée avec une animation Director, consultez l'animation Shockwave et flux continu dans l'Aide de Director.

Cette fonction peut être testée, mais pas définie.

Exemples

L'instruction suivante détermine si les acteurs de l'image 20 sont téléchargés et prêts à être affichés :

```
on exitFrame
  if frameReady(20) then
    -- passer à l'image 20 si tous les acteurs
    -- nécessaires sont disponibles localement
    go to frame 20
  else
    -- reprendre la boucle pendant le
    -- téléchargement de l'arrière-plan
    got to frame 1
  end if
end
```

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.

```
on exitFrame
  if the frameReady(sprite 5, 25) = FALSE then
    go to the frame
  else
    play sprite 5
  end if
end
```

Voir aussi

`mediaReady`

frameScript

Syntaxe

```
the frameScript
```

Description

Propriété d'image ; contient le numéro d'acteur unique du script d'image affecté à l'image courante.

La propriété `frameScript` peut être testée. Au cours d'une session d'enregistrement de scénario, vous pouvez également affecter un script à l'image courante en définissant la propriété `frameScript`.

Exemples

L'instruction suivante affiche le numéro du script affecté à l'image courante. Dans ce cas, le numéro de script est 25.

```
put the frameScript
-- 25
```

L'instruction suivante fait de l'acteur script Réponse des boutons le script d'image pour l'image courante :

```
the frameScript = member "Réponse des boutons"
```

frameSound1

Syntaxe

```
the frameSound1
```

Description

Propriété d'image ; détermine le numéro de l'acteur affecté à la première piste audio de l'image courante.

Cette propriété peut être testée et définie. 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 :

```
the frameSound1 = member("Jazz").number
```

frameSound2

Syntaxe

```
the frameSound2
```

Description

Propriété d'image ; détermine le numéro de l'acteur affecté à la seconde piste audio de l'image courante.

Cette propriété peut être testée et définie. 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 :

```
the frameSound2 = member("Jazz").number
```

framesToHMS()

Syntaxe

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.

- *images* – Expression entière spécifiant le nombre d'images.
- *cadence* – Expression entière spécifiant le nombre d'images par seconde.
- *compensé* – Compense la cadence NTSC couleur qui n'est pas exactement de 30 images par seconde et n'est utile que pour une cadence de 30 images par seconde. Normalement, cet argument a pour valeur FALSE.
- *fractions* – Détermine si les images résiduelles sont converties au centième de seconde le plus proche (TRUE) ou renvoyées sous la forme d'un nombre d'images entier (FALSE).

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 *composé* a la valeur TRUE ; une espace est utilisée si *composé* a la valeur 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()

frameTempo

Syntaxe

```
the frameTempo
```

Description

Propriété d'image ; indique la cadence affectée à l'image courante.

Cette propriété peut être testée. Elle peut être définie au cours d'une session d'enregistrement du scénario.

Exemple

L'instruction suivante vérifie la cadence utilisée dans l'image courante. Dans ce cas, la cadence est de 15 images par seconde.

```
put the frameTempo
-- 15
```

Voir aussi

puppetTempo

frameTransition

Syntaxe

```
the frameTransition
```

Description

Propriété d'image ; spécifie le numéro de l'acteur de transition affecté à l'image courante.

Cette propriété peut être définie au cours d'une session d'enregistrement du scénario pour spécifier des transitions.

Exemple

Lorsque 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 :

```
set the frameTransition to member "Brouillard"
```

freeBlock()

Syntaxe

```
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.

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 :

```
if (the freeBlock < (10 * 1024)) then alert "Mémoire insuffisante !"
```

Voir aussi

`freeBytes()`, `memorySize`, `ramNeeded()`, `size`

freeBytes()

Syntaxe

```
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 le 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.

Exemple

L'instruction suivante vérifie si plus de 200 ko de mémoire est disponible 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

front

Syntaxe

member(*quelActeur*).modelResource(*quelleRessDeMod*).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

Syntaxe

the frontWindow

Description

Propriété système ; indique l'animation dans une fenêtre actuellement au premier plan de l'écran. Lorsque la scène est au premier plan, la fenêtre au premier plan est la scène. Lorsqu'un éditeur de média ou une palette flottante est au premier plan, frontWindow renvoie la valeur VOID.

Cette propriété peut être testée, mais pas définie.

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 :

```
if the frontWindow = "Musique" then window("Ecoutez ça").moveToFront
```

Voir aussi

activeWindow, on activateWindow, on deactivateWindow, moveToFront

generateNormals()

Syntaxe

```
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.

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

Syntaxe

```
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`.

Exemples

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 qu'un crochet d'accès est utilisé.

Voir aussi

`getAt`, `getOne()`, `getProp()`, `setaProp`, `setAt`

getAt

Syntaxe

```
getAt(liste, position)
liste [position]
```

Description

Commande de liste ; identifie l'élément à la position spécifiée par *position* dans la 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`.

Exemples

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.

```
premièrePersonne = getAt(listeDinfosDesEmployés, 1)
put premièrePersonne
-- ["Denis", "Conseil", 510]
serviceDeLaPremièrePersonne = getAt(premièrePersonne, 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 :

```
premièrePersonne = listeDinfosDesEmployés[1]
put premièrePersonne
-- ["Denis", "Conseil", 510]
serviceDeLaPremièrePersonne = premièrePersonne[2]
put serviceDeLaPremièrePersonne
-- "Conseil"
```

De même qu'avec `getAt`, les crochets peuvent être imbriqués :

```
serviceDeLaPremièrePersonne = listeDinfosDesEmployés[1][2]
```

Voir aussi

`getaProp`, `setaProp`, `setAt`

on getBehaviorDescription

Syntaxe

```
on getBehaviorDescription
   instruction(s)
end
```

Description

Message système et gestionnaire d'événement ; contient des instructions Lingo 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 :

```
on getBehaviorDescription
  return "Barre de défilement vertical de champ de texte multiligne"
end
```

Voir aussi

`on getPropertyDescriptionList`, `on getBehaviorTooltip`, `on runPropertyDialog`

on getBehaviorTooltip

Syntaxe

```
on getBehaviorTooltip
  instruction(s)
end
```

Description

Message système et gestionnaire d'événement ; contient des instructions Lingo 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 :

```
on getBehaviorTooltip
  return "Pièce de puzzle"
end
```

Voir aussi

`on getPropertyDescriptionList`, `on getBehaviorDescription`, `on runPropertyDialog`

getBoneID

Syntaxe

`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`

getError()

Syntaxe

```
member(quelActeurSWA).getError()
getError(member quelActeurSWA)
member(quelActeurFlash).getError()
getError(member quelActeurFlash)
```

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	mémoire
2	réseau
3	périphérique de lecture
99	autre

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.

Exemples

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 :

```
on exitFrame
  if member("Norma Desmond parle").getError() <> 0 then
    member("Affichage de l'erreur").text = member("Norma Desmond \
  parle").getErrorString()
  end if
end
```

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.

```
on vérifierEtatFlash
  testErreur = member("Dali").getError()
  if testErreur <> 0 then
    if testErreur = #memory then
      member("Dali").clearError()
      unloadCast
      go to frame ("Artistes")
    else
      go to frame ("Désolé")
    end if
  end if
end
```

Voir aussi

`clearError`, `getErrorString()`, `state (Flash, SWA)`

getError() (XML)

Syntaxe

`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.

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()

Syntaxe

```
member(quelActeur).getErrorString()
getErrorString(member quelActeur)
```

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	mémoire
2	réseau
3	périphérique de lecture
99	autre

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 :

```
on exitFrame
  if member("Norma Desmond parle").getError() <> 0 then
    member("Affichage de l'erreur").text = member("Norma Desmond \
  parle").getErrorString()
  end if
end
```

Voir aussi

`getError()`

getFlashProperty()

Syntaxe

```
sprite(numéroDimageObjet).getFlashProperty("NomDeCible", #propriété)
```

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.

`nomDeCible` est le nom du clip ou du niveau de l'animation dont vous souhaitez obtenir la propriété dans l'image-objet Flash donnée.

Le paramètre *`#propriété`* est le nom de la propriété à obtenir. Ces propriétés peuvent être testées : `#posX`, `#posY`, `#scaleX`, `#scaleY`, `#visible`, `#rotate`, `#alpha`, `#name`, `#width`, `#height`, `#target`, `#url`, `#dropTarget`, `#totalFrames`, `#currentFrame`, `#cursor` et `#lastframeLoaded`.

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.

Exemple

L'instruction suivante obtient la valeur de la propriété `#rotate` du clip Etoile dans l'acteur Flash de l'image-objet 3 :

```
sprite(3).setFlashProperty("Etoile", #rotate)
setFlashProperty()
```

getFrameLabel()

Syntaxe

```
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'est pas encore transférée en mémoire, cette fonction renvoie une chaîne vide.

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.

```
on exitFrame
  if sprite(1).getFrameLabel(15) = "Lions" then
    go "Lions"
  else
    go the frame
  end if
end
```

getHardwareInfo()

Syntaxe

```
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()

Syntaxe

```
sprite(quelleImageObjetQTVR).getHotSpotRect(idDeZoneRéférencée)
getHotSpotRect(quelleImageObjetQTVR, idDeZoneRéférencée)
```

Description

Fonction QuickTime VR ; renvoie un rectangle de délimitation approximative pour la zone référencée spécifiée par *idDeZoneRéférencée*. 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.

getLast()

Syntaxe

```
liste.getLast()
getLast(liste)
```

Description

Fonction de liste ; identifie la dernière valeur d'une liste linéaire ou de propriétés spécifiée par *liste*.

Exemples

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

Syntaxe

```
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.

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()`

getNetText()

Syntaxe

```
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 initie 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.

Pour une animation lue comme applet, la commande `getNetText` récupère le texte uniquement à partir du domaine contenant l'applet. Ce comportement est différent de Shockwave et est nécessaire en raison du modèle de sécurité de Java.

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.

Vous pourrez voir un exemple de `getNextText()` dans une animation en consultant l'animation Forms and Post du dossier Learning/Lingo_Examples, lui-même dans le dossier de Director.

Exemples

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

Syntaxe

```
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 est long d'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()

Syntaxe

```
getNthFileNameInFolder(cheminDeDossier, numéroDeFichier)
```

Description

Fonction ; renvoie un nom de fichier contenu dans un dossier d'après le chemin d'accès et le 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 le 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 le 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.

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)`

getOne()

Syntaxe

```
liste.getOne(valeur)
getOne(liste, valeur)
```

Description

Fonction de liste ; identifie la position (liste linéaire) ou propriété (liste de propriétés) associée à la valeur spécifiée par *`valeur`* dans la liste spécifiée par *`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`.

Exemples

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()`

getPixel()

Syntaxe

```
objetImage.getPixel(x, y {, #entier})
objetImage.getPixel(point(x, y) {, #entier})
```

Description

Cette fonction renvoie la valeur de la couleur du pixel au point spécifié dans l'objet image donné. En principe, cette valeur est renvoyée sous forme d'objet couleur rvb ou indexé, selon le codage des couleurs de l'image.

Toutefois, si vous avez inclus le paramètre facultatif *#entier*, il sera renvoyé comme nombre brut. Si vous définissez un grand nombre de pixels sur la couleur d'un autre pixel, il est plus rapide de les définir en tant que nombre entiers. 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 le divisant le nombre par 2^8+8+8.

`GetPixel()` renvoie 0 si le pixel donné se trouve en dehors de l'objet image spécifié.

Exemples

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.

```
maCouleur=member("Joyeux").image.getPixel(90, 20)
sprite(2).color=maCouleur
```

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.

```
alpha = monImage.getPixel(25, 33, #entier) / power(2, 8+8+8)
```

Voir aussi

`depth, color(), setPixel(), power()`

getPlaylist()

Syntaxe

```
sound(numéroDePiste).getPlaylist()
getPlaylist(sound(numéroDePiste))
```

Description

Cette fonction renvoie une copie de la liste des sons mis en attente pour *objetAudio*. Cette liste ne contient pas le son actuellement en cours.

La liste des sons placés en file d'attente ne peut pas être modifiée 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	Acteur son à lire. Cette propriété sera toujours présente ; toutes les autres sont facultatives.
#startTime	Position temporelle de départ de la lecture du son, en millisecondes. Pour plus d'informations, consultez `startTime`.
#endTime	Position temporelle de fin de la lecture du son, en millisecondes. Pour plus d'informations, consultez `endTime`.
#loopCount	Nombre de lectures d'une partie du son. Pour plus d'informations, consultez `loopCount`.
#loopStartTime	Position temporelle de départ d'une boucle, en millisecondes. Pour plus d'informations, consultez `loopStartTime`.
#loopEndTime	Position temporelle de fin d'une 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

Ce gestionnaire 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.

```
on lireLaMusique
  sound(2).queue([#member:member("carillons")])
  sound(2).queue([#member:member("intro"), #startTime:3000,\
  #endTime:10000, #loopCount:5,#loopStartTime:8000, #loopEndTime:8900])
  sound(2).play()
  put sound(2).getPlaylist()
end
-- [[#member: (member 12 of castLib 2), #startTime: 3000, #endTime: 10000,
  #loopCount: 5, #loopStartTime: 8000, #loopEndTime: 8900]]
```

Voir aussi

`endTime`, `loopCount`, `loopEndTime`, `loopStartTime`, `member (mot-clé)`, `preLoadTime`, `queue()`, `setPlaylist()`, `startTime`

getPos()

Syntaxe

```
liste.getPos(valeur)
getPos(liste, valeur)
```

Description

Fonction de liste ; identifie la position de la valeur spécifiée par *valeur* dans la liste spécifiée par *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.

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()

Syntaxe

```
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 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`.

Vous pourrez voir un exemple de `getPref()` dans une animation en consultant l'animation Read and Write Text du dossier Learning/Lingo_Examples, lui-même dans le dossier de Director.

Exemple

Le gestionnaire suivant récupère le contenu du fichier Test et en affecte le texte au champ Score :

```
on mouseUp
  theText = getPref("Test")
  member("Score").text = theText
end
```

Voir aussi

`setPref`

getProp()

Syntaxe

```
getProp(liste, propriété)
liste.propriété
```

Description

Fonction de liste de propriétés ; identifie la valeur associée à la propriété spécifiée par *propriété* dans la liste de propriétés spécifiée par *liste*.

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.

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()

Syntaxe

```
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 la position spécifiée par *index* dans la liste de propriétés indiquée par *liste*. 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.

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`.

on getPropertyDescriptionList

Syntaxe

```
on getPropertyDescriptionList
  instruction(s)
end
```

Description

Message système et gestionnaire d'événement ; contient des instructions Lingo 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 = [:]
  addProp description,#dynamic, [#default:1, #format:#boolean,
  #comment:"Dynamique"]
  addProp description,#fieldNum, [#default:1, #format:#integer,  \
  #comment:"Image-objet à faire défiler :"]
  addProp description, #extentSprite,[#default:1,#format:#integer,  \
  #comment: "Etendre l'image-objet :"]
  addProp description,#proportional,[#default:1,#format:#boolean, \
  #comment: "Proportionnel :"]
  return description
end
```

Voir aussi

`addProp`, `on getBehaviorDescription`, `on runPropertyDialog`

getRendererServices()

Syntaxe

```
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 utilisés par les modèles des acteurs 3D. Les valeurs possibles sont `#collision`, `#bonesPlayer`, `#keyframePlayer`, `#toon`, `#lod`, `#meshDeform`, `#sds`, `#inker` et des modificateur basés sur des Xtras publiés par d'autres éditeurs. 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.

Voir aussi

`renderer`, `preferred3DRenderer`, `active3dRenderer`, `rendererDeviceList`

getStreamStatus()

Syntaxe

```
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 train, s'il est connu. Cette valeur peut être zéro si le serveur HTTP ne 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()`.

Exemple

L'instruction suivante affiche dans la fenêtre Messages l'état du 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()

Syntaxe

```
sprite(numéroDimageObjetFlash).getVariable("nomDeVariable" {,
  valeurOuRéfRenvoyée})
getVariable(sprite numéroDimageObjetFlash, "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.

Exemples

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 :

```
valeurT = sprite(3).getVariable("gAutreVar",TRUE)

put 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 :

```
objetT = sprite(3).getVariable("gVar",FALSE)
```

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.

```
put getVariable(sprite 3, "URLactuelle")
-- "http://www.macromedia.com/software/flash/"
```

Voir aussi

setVariable()

getWorldTransform()

Syntaxe

```
member(quelActeur).node(quelNœud).getWorldTransform()
member(quelActeur).node(quelNœud).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 et est donc relative à l'univers.

Utilisez member(*quelActeur*).node(*quelNœud*).getWorldTransform().position pour trouver la propriété de position de la transformation du nœud par rapport à l'univers. Vous pouvez également utiliser worldPosition comme raccourci de getWorldTransform().position.

Utilisez member(*quelActeur*).node(*quelNœud*).getWorldTransform().rotation pour trouver la propriété de rotation de la transformation du nœud par rapport à l'univers.

Utilisez member(*quelActeur*).node(*quelNœud*).getWorldTransform().scale pour trouver la propriété d'échelle de la transformation du nœud par rapport à l'univers.

Ces propriétés peuvent être testées, mais pas définies.

Exemple

L'instruction suivante indique la transformation relative à l'univers du modèle Boîte, suivi 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)`

global

Syntaxe

`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 Shockwave lues sur Internet ne peuvent pas accéder à des variables globales dans d'autres animations, même si les animations sont lues 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 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

globals

Syntaxe

```
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 la commande showGlobals pour afficher the globals dans la fenêtre Messages.

Voir aussi

showGlobals, clearGlobals

glossMap

Syntaxe

```
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

Syntaxe

`member(`*`quelActeur`*`).modelResource(`*`quelleRessDeMod`*`).gravity`

Description

Propriété 3D de ressource de modèle de système de particules ; lorsque 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`

go

Syntaxe

```
go {to} {frame} quelleImage
go {to} movie quelleAnimation
go {to} {frame} quelleImage of movie quelleAnimation
```

Description

Commande ; place la tête de lecture sur l'image spécifiée par *`quelleImage`* dans l'animation spécifiée par *`quelleAnimation`*. L'expression *`quelleImage`* peut être un libellé de repère ou un numéro d'image entier. L'expression *`quelleAnimation`* doit spécifier un fichier d'animation. Si l'animation se trouve dans un autre dossier, *`quelleAnimation`* doit spécifier son chemin d'accès.

La phrase `go loop` indique à la tête de lecture d'effectuer une boucle vers le repère précédent et est un moyen pratique de garder la tête de lecture dans la même partie de l'animation pendant que Lingo reste actif tout en évitant l'utilisation de `go to the frame` dans une image comportant une transition, ce qui ralentirait l'animation et surchargerait le processeur.

Il est préférable de faire référence aux libellés de repères plutôt qu'aux numéros d'images, la modification d'une animation pouvant entraîner le changement des numéros d'images. L'utilisation du libellé des repères facilite également la lecture des scripts.

La commande `go to movie` charge l'image 1 de l'animation. Si cette commande 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 commande `play` , qui peut être suivie de `play done` 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, afin d'empêcher un 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 commande `downloadNetThing`, afin de télécharger le fichier de l'animation sur un disque local, puis utilisez la commande `go to movie` pour passer à cette animation en utilisant le fichier sur le disque local.

Les paramètres suivants sont réinitialisés au chargement d'une animation : propriétés `beepOn`et `constraint` ; propriétés d'images-objets `keyDownScript`, `mouseDownScript` et `mouseUpScript` ; `cursor` et `immediate` ; commandes `cursor` et `puppetSprite` et menus personnalisés. Cependant, `the timeoutScript` n'est pas réinitialisé au chargement de l'animation.

Exemples

L'instruction suivante envoie la tête de lecture au point de repère intitulé Début :

```
go to "Début"
```

L'instruction suivante envoie la tête de lecture au point de repère Mémoire de l'animation intitulée Mon animation :

```
go frame("Mémoire") of movie("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 attende dans une image tout en répondant aux événements.

```
on exitFrame
  go the frame
end
```

Voir aussi

`downloadNetThing`, `gotoNetMovie`, `label()`, `marker()`, `pathName (propriété d'animation)`, `play`, `play done`

go loop

Syntaxe

```
go loop
```

Description

Commande ; 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.

Remarque Cette commande est équivalente à `marker(0)` dans les versions antérieures à Director 7.

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.

La commande `go loop` est équivalente à l'instruction `go to the marker(0)` utilisée dans les versions précédentes de Lingo.

Exemple

L'instruction suivante fait faire une boucle à l'animation entre l'image courante et le repère précédent :

```
go loop
```

Voir aussi

`go`, `go next`, `go previous`

go next

Syntaxe

```
go next
```

Description

Commande ; 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.

La commande `go next` est équivalente à l'instruction `go marker(1)` utilisée dans les versions précédentes de Lingo.

Exemple

L'instruction suivante envoie la tête de lecture sur le repère suivant de l'animation :

```
go next
```

Voir aussi

`go`, `go loop`, `go previous`

go previous

Syntaxe

```
go previous
```

Description

Commande ; place la tête de lecture sur le point de repère précédent dans l'animation. Ce repère se trouve à deux repères en arrière de l'image courante si celle-ci ne possède pas de repère ou à un repère en arrière si elle en possède un.

Remarque Cette commande est équivalente à `marker(-1)` dans les versions précédentes de Director.

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.

Exemple

L'instruction suivante envoie la tête de lecture sur le repère précédent dans l'animation :

```
go previous
```

Voir aussi

`go`, `go next`, `go loop`

goToFrame

Syntaxe

```
sprite(quelleImageObjetFlash).goToFrame(numéroDimage)
goToFrame(sprite quelleImageObjetFlash, numéroDimage)
sprite(quelleImageObjetFlash).goToFrame(chaîneDeLibellé)
goToFrame(sprite quelleImageObjetFlash, chaîneDeLibellé)
```

Description

Commande ; lit une image-objet d'animation Flash à partir de l'image identifiée par le paramètre *numéroDimage*. 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 indiquant l'image à laquelle passer.

```
on naviguer Cible
  sprite(5).goToFrame(Cible)
end
```

gotoNetMovie

Syntaxe

```
gotoNetMovie URL
gotoNetMovie (URL)
```

Description

Commande ; récupère et lit une nouvelle animation 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.

Exemples

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

Syntaxe

```
gotoNetPage "URL", {"nomDeCible"}
```

Description

Commande ; ouvre une animation Shockwave ou un autre fichier MIME dans le navigateur web.

Seules les URL sont supportées comme paramètres valides. Les URL relatives fonctionnent si l'animation se trouve sur un serveur HTTP ou FTP.

L'argument *nomDeCible* est un paramètre HTML facultatif qui identifie le cadre ou la fenêtre dans laquelle 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* n'est pas inclus, `gotoNetPage` remplace la page courante, où qu'elle se trouve.

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.

Exemples

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()`

gradientType

Syntaxe

`member(`*`quelActeur`*`).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 valide 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.

```
on mouseUp me
  if member("Fond").gradientType = #radial then
    member("Fond").gradientType = #linear
  else
    member("fond").gradientType = #radial
  end if
end
```

Voir aussi

`fillMode`

group

Syntaxe

```
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("univers")`.

Exemples

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`, `parent`

halt

Syntaxe

```
halt
```

Description

Commande ; 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.

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 :

```
if the freeBytes < 50*1024 then halt
```

Voir aussi

`abort`, `exit`, `pass`, `quit`

handler()

Syntaxe

```
objetScript.handler(#symboleDeGestionnaire)
```

Description

Cette fonction renvoie `TRUE` si l'*objetScript* donné contient un gestionnaire dont le nom est #*symboleDeGestionnaire* et `FALSE` dans le cas contraire. L'objet script doit être un script parent, un objet enfant ou un comportement.

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()

Syntaxe

`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.

Notez que 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.

Exemples

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`

height

Syntaxe

```
member(quelActeur).height
the height of member quelActeur
objetImage.height
sprite(quelleImageObjet).height
the height of sprite quelleImageObjet
```

Description

Propriété d'acteur, d'objet image et d'image-objet ; pour les acteurs forme vectorielle, Flash, GIF animé, bitmap et forme, détermine la hauteur, en pixels, de l'acteur affiché sur la scène.

- Pour les acteurs, ne fonctionne qu'avec les acteurs bitmap et forme vectorielle. Cette propriété peut être testée, mais pas définie.
- Dans le cas d'un objet image, cette propriété peut être testée, mais pas définie.

- Pour les images-objets, la définition de la hauteur de l'image-objet affecte automatiquement à la propriété `stretch` de l'image-objet la valeur `TRUE`. Pour que la valeur définie par Lingo dure au-delà de l'image-objet courante, celle-ci doit être un esclave. Cette propriété peut être testée et définie.

Exemples

L'instruction suivante affecte la hauteur de l'acteur Titre à la variable *vHauteur* :

```
vHauteur = member("Titre").height
```

L'instruction suivante définit la hauteur de l'image-objet 10 à 26 pixels :

```
sprite(10).height = 26
```

L'instruction suivante affecte la hauteur de l'image-objet (i + 1) à la variable *vHauteur* :

```
vHauteur = sprite(i + 1).height
```

Voir aussi

`height`, `rect (image-objet)`, `width`

height (3D)

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).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.

Exemples

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

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).\
  heightVertices
```

Description

Propriété 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

Syntaxe

```
member(quelActeur).model(quelModèle).toon.highlightPercentage
member(quelActeur).model(quelModèle).shader.highlightPercentage
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

Syntaxe

```
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 (commande)

Syntaxe

```
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 le Macintosh, la couleur de sélection est définie dans le tableau de bord Couleur.

Exemples

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`

hilite (propriété d'acteur)

Syntaxe

```
member(quelActeur).hilite
the hilite of member quelActeur
```

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).

Si *`quelActeur`* est une chaîne, elle spécifie le nom de l'acteur. S'il s'agit d'un nombre entier, *`quelActeur`* spécifie le numéro de l'acteur.

Cette propriété peut être testée et définie.

Exemples

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 :

```
if member("Son Activé").hilite = TRUE then sound(1).volume = 255
```

L'instruction suivante utilise Lingo pour sélectionner l'acteur bouton Interrupteur en donnant à la propriété d'acteur `hilite` la valeur `TRUE` :

```
member("Interrupteur").hilite = TRUE
```

Voir aussi

`checkBoxAccess`, `checkBoxType`

hitTest()

Syntaxe

```
sprite(quelleImageObjetFlash).hitTest(point)
hitTest(sprite quelleImageObjetFlash, 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.

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 :

```
on exitFrame
  if sprite(5).hitTest(the mouseLoc) = #button then
    member("Ligne de message").text = "Cliquez ici pour lire l'animation."
    updateStage
  else
    member("Ligne de message").text = " "
  end if
  go the frame
end
```

hither

Syntaxe

```
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`

HMStoFrames()

Syntaxe

```
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).

- *hms* – Expression de chaîne spécifiant 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 d'une espace si *composé* a la valeur FALSE.

- *cadence* – Spécifie la cadence en images par seconde.
- *compensé* – Expression logique déterminant 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* – Expression logique déterminant 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).

Exemples

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

Syntaxe

```
sprite(quelleImageObjetFlash).hitTest(point)
hold sprite quelleImageObjetFlash
```

Description

Commande Flash ; arrête une image-objet animation Flash lue dans l'image courante, sans interrompre la lecture audio.

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 :

```
on enterFrame
  repeat with i = 5 to 10
    sprite(i).hold()
  end repeat
end
```

Voir aussi

movieRate, pause (lecture d'animation)

hotSpot

Syntaxe

```
member(quelActeurCurseur).hotspot
the hotspot of member quelActeurCurseur
```

Description

Propriété d'acteur curseur ; spécifie 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 produit 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 :

```
on startMovie
  member(numéroDeCurseur).hotSpot = point(16,16)
end
```

hotSpotEnterCallback

Syntaxe

```
sprite(quelleImageObjetQTVR).hotSpotEnterCallback
the hotSpotEnterCallback of sprite quelleImageObjetQTVR
```

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

Syntaxe

```
sprite(quelleImageObjetQTVR).hotSpotExitCallback
the hotSpotExitCallback of sprite quelleImageObjetQTVR
```

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

Syntaxe

```
member(quelActeur).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 :

```
put member("Page d'accueil").HTML
```

Voir aussi

importFileInto, RTF

hyperlink

Syntaxe

```
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.

```
on startMovie
  member("LienMacromedia").word[1].hyperlink = \
  "http://www.macromedia.com"
end
```

Voir aussi

`hyperlinkRange`, `hyperlinkState`

on hyperlinkClicked

Syntaxe

```
on hyperlinkClicked me, données, plage
  instruction(s)
end
```

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 :

```
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
```

hyperlinkRange

Syntaxe

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

Syntaxe

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 :

```
put member("Glossaire").hyperlinks
-- [[3, 8], [10, 16], [41, 54]]
```

hyperlinkState

Syntaxe

sousChaîneTexte.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.

```
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
```

Voir aussi

`hyperlink`, `hyperlinkRange`

identity()

Syntaxe

```
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.

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()`

on idle

Syntaxe

```
on idle
  instruction(s)
end
```

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 Lingo 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 :

```
on idle
  member("Heure").text = the short time
end idle
```

Voir aussi

`idleHandlerPeriod`

idleHandlerPeriod

Syntaxe

```
the idleHandlerPeriod
```

Description

Propriété d'animation ; détermine le nombre de battements maximum avant l'envoi d'un message `idle` par l'animation. 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 Lingo 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 résulte du fait que 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, ce qui diffère des versions précédentes dans lesquelles elle était de 0.

Exemple

L'instruction suivante entraîne l'animation à envoyer un message `idle` une fois par seconde, au maximum :

```
the idleHandlerPeriod = 60
```

Voir aussi

`idleLoadDone()`, `idleLoadMode`, `idleLoadTag`, `idleReadChunkSize`

idleLoadDone()

Syntaxe

`idleLoadDone(`*`baliseDeChargement`*`)`

Description

Fonction ; indique si tous les acteurs ayant la balise spécifiée ont été chargés (`TRUE`) ou s'ils doivent encore être chargés (`FALSE`).

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 :

```
if idleLoadDone(20) then play movie("Kiosque")
```

Voir aussi

`idleHandlerPeriod`, `idleLoadMode`, `idleLoadPeriod`, `idleLoadTag`, `idleReadChunkSize`

idleLoadMode

Syntaxe

```
the idleLoadMode
```

Description

Propriété système ; détermine le moment auquel les commandes `preLoad` et `preLoadMember` essayent de charger les acteurs pendant les périodes d'inactivité en fonction 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 système `idleLoadMode` n'effectue aucune fonction et n'est exécutée qu'avec les commandes `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` :

```
the idleLoadMode = 3
```

Voir aussi

`idleHandlerPeriod`, `idleLoadDone()`, `idleLoadPeriod`, `idleLoadTag`, `idleReadChunkSize`

idleLoadPeriod

Syntaxe

```
the idleLoadPeriod
```

Description

Propriété système ; détermine le nombre de battements pendant lequel Director attend avant d'essayer de charger les acteurs en attente de chargement. 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 :

```
set the idleLoadPeriod = 30
```

Voir aussi

`idleLoadDone()`, `idleLoadMode`, `idleLoadTag`, `idleReadChunkSize`

idleLoadTag

Syntaxe

```
the idleLoadTag
```

Description

Propriété système ; identifie ou affecte un numéro aux acteurs en attente de chargement pendant les périodes d'inactivité de l'ordinateur. La propriété `idleLoadTag` est pratique pour identifier les acteurs d'un groupe que vous souhaitez précharger.

Cette propriété peut être testée et définie avec n'importe quel numéro.

Exemple

L'instruction suivante fait du numéro 10 la balise de chargement en période d'inactivité :

```
the idleLoadTag = 10
```

Voir aussi

`idleHandlerPeriod`, `idleLoadDone()`, `idleLoadMode`, `idleLoadPeriod`, `idleReadChunkSize`

idleReadChunkSize

Syntaxe

```
the idleReadChunkSize
```

Description

Propriété système ; détermine le nombre maximum d'octets que Director peut charger lorsqu'il essaie de charger les acteurs à partir de la file de chargement. La valeur par défaut est 32 ko.

Cette propriété peut être testée et définie.

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 :

```
the idleReadChunkSize = 500 * 1024
```

Voir aussi

`idleHandlerPeriod`, `idleLoadDone()`, `idleLoadMode`, `idleLoadPeriod`, `idleLoadTag`

if

Syntaxe

```
if expressionLogique 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* 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.

Exemples

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 (the commandDown) and (the key = "q") then
    nettoyage
    quit
  end if
end keyDown
```

Comparez les deux constructions suivantes et les résultats de performance. La première construction évalue les deux conditions et doit déterminer la mesure du temps, ce qui prend un certain temps. 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
  alert "Vous avez trouvé le trésor !"
end if
```

La construction plus rapide serait :

```
imageObjetSousLeCurseur = rollOver()
if (imageObjetSousLeCurseur > 25) then
  if MesureDuTempsDepuisLeDépart() then
    alert "Vous avez trouvé le trésor !"
  end if
end if
```

Voir aussi

`case`

ignoreWhiteSpace()

Syntaxe

`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.

Exemples

Les instructions Lingo suivantes laissent à `ignoreWhiteSpace()` sa valeur par défaut de `TRUE` et convertissent le document XML donné en une liste. Vous remarquerez que 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. Vous remarquerez que 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. Vous remarquerez qu'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. Vous remarquerez qu'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()

Syntaxe

```
ilk(objet)
ilk(objet, type)
```

Description

Fonction ; indique le type d'un objet.

- La syntaxe `ilk(objet)` renvoie une valeur indiquant le type d'un objet. Si l'objet se trouve dans la liste, `ilk(objet)` renvoie `#list` ; si l'objet est une liste de propriétés, `ilk(objet)` renvoie `#propList`.

- La syntaxe `ilk(objet, type)` compare l'objet représenté par la valeur *objet* au type spécifié. 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`.

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"))`
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)`
média	`#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)`

Exemples

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)

Syntaxe

`ilk(`*`objet`*`)`
`ilk(`*`objet`*`,`*`type`*`)`
`objet``.ilk`
`objet``.ilk(`*`type`*`)`

Description

Fonction Lingo ; indique le type d'un objet.

- La syntaxe `ilk(`*`objet`*`)` et *`objet`*`.ilk` renvoie une valeur indiquant le type de l'objet. Si l'objet est un modèle, `ilk(`*`objet`*`)` renvoie `#model` ; si l'objet est un mouvement, `ilk(`*`objet`*`)` renvoie `#motion`. Vous trouverez une liste complète des valeurs renvoyées par les objets 3D dans le tableau suivant.
- La syntaxe `ilk(`*`objet`*`, `*`type`*`)` et *`objet`*`.ilk(`*`type`*`)` compare l'objet représenté par l'objet au type spécifié. 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`.

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

Type d'objet	ilk(objet) renvoie	ilk(objet, type) uniquement si type =
texture	#texture	#texture ou #list
groupe	#group	#group
caméra	#camera	#camera
données de collision	#collisiondata	#collisiondata
vecteurs	#vector	#vector
transformation	#transform	#transform

Exemples

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

Syntaxe

```
quelActeur.image
(the stage).image
window(nomDeFenêtre).image
```

Description

Cette propriété fait référence à l'objet image d'un acteur bitmap ou texte, de la scène ou d'une fenêtre. Vous pouvez obtenir ou définir l'image d'un acteur, mais ne pouvez qu'obtenir l'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 à l'image, 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 fonction `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 fonction `duplicate()`.

Exemples

L'instruction suivante place l'image de l'acteur fleurDorigine dans l'acteur nouvelleFleur :

```
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.

```
monImage = (the stage).image
member("fleur").image = monImage
```

Voir aussi

setPixel(), draw(), image(), fill(), duplicate() (fonction d'image), copyPixels()

image()

Syntaxe

image(*largeur*, *hauteur*, *codage* {, *codageAlpha*} {, *symboleOuActeurPalette*})

Description

Fonction ; crée et renvoie un objet image des dimensions spécifiées par *largeur*, *hauteur* et *codage*, avec les valeurs facultatives de *codageAlpha* et *symboleOuActeurPalette*.

Le codage peut être 1, 2, 4, 8, 16 ou 32. Le codageAlpha, si donné, n'est utilisé que pour les images 32 bits et doit être 0 ou 8. L'objetPalette, si donné, n'est utilisé que pour les images 2, 4 et 8 bits images et peut être un symbole de palette, tel que #grayscale ou un acteur palette. Si aucune palette n'est spécifiée, c'est la palette par défaut de l'animation qui est utilisée.

Les objets image créés avec image() sont indépendants et ne font référence ni à un acteur, ni à une fenêtre.

Vous pourrez voir un exemple de image() dans une animation en consultant l'animation Imaging du dossier Learning/Lingo_Examples, lui-même dans le dossier de Director.

Exemple

Les instructions suivantes créent un objet image de 200 x 200 pixels, 8 bits et le remplissent de rouge.

```
carréRouge = image(200, 200, 8)
carréRouge.fill(0, 0, 200, 200, rgb(255, 0, 0))
```

Voir aussi

palette, image, duplicate() (fonction d'image), fill()

image (RealMedia)

Syntaxe

sprite(*quelleImageObjet*).image
member(*quelActeur*).image

Description

Propriété d'acteur ou d'image-objet RealMedia ; renvoie un objet image Lingo contenant l'image courante du train 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 :

```
member("Insta").image = member("Real").image
```

Ce gestionnaire est appelé une fois par un script d'image, pour chaque image Director. Le gestionnaire crée une nouvelle texture à partir de l'image de l'acteur RealMedia appelé Real, s'il est en cours de lecture ou en pause. La nouvelle texture est ensuite utilisée par le matériau du modèle appelé mSphère. La texture utilisée dans l'image précédente est supprimée. Enfin, la sphère pivote de 1° autour de l'axe des *y*. Le résultat est une vidéo `#realMedia` lue dans une sphère en rotation.

```
on majMatériau
  if member("Real").state = 4 then
    matSphère = member("3d").model("mSphère").shader
    tex = member("3d").newTexture("texture" & gNumDeTexture, \
          #fromImageObject, member("Real").image)
    matSphère.texture = tex
    if gNumDeTexture > 1 then
      member("3d").deleteTexture("texture" & (gNumDeTexture - 1))
    end if
    gNumDeTexture = gNumDeTexture + 1
  end if
  member("3d").model[1].transform.rotate(0, 1, 0)
end
```

imageCompression

Syntaxe

```
member(quelActeur).imageCompression
the imageCompression of member quelActeur
```

Description

Cette propriété d'acteur bitmap indique le type de compression que Director applique à l'acteur lors de l'enregistrement de l'animation au format Shockwave. Cette propriété peut être testée et définie et n'a aucun effet à l'exécution. Sa valeur peut être constituée de l'un de ces symboles :

Valeur	Signification
#movieSetting	Utiliser les paramètres de compression de l'animation stockés dans la propriété movieImageCompression. Il s'agit de la valeur par défaut pour les formats d'image non limités à la compression standard (voir ci-après).
#standard	Utilise le format de compression interne standard de Director.
#jpeg	La compression JPEG est utilisée. Pour plus d'informations, consultez `imageQuality`.

Cette propriété est normalement définie dans le volet Bitmap de l'inspecteur des propriétés. Toutefois, si vous souhaitez définir cette propriété pour un nombre important d'images en une seule opération, vous pouvez la définir à l'aide d'une routine Lingo.

Si un acteur ne supporte pas la compression JPEG (s'il s'agit d'un acteur 8 bits ou une valeur inférieure) ou si l'image est liée à un fichier externe, seule la compression `#standard` pourra être utilisée. Les formats d'image qui ne supportent pas la compression JPEG comprennent le format GIF et les images 8 bits (ou une valeur inférieure).

Exemple

L'instruction suivante affiche la valeur de compression de l'acteur Lever de soleil dans la fenêtre Messages :

```
put member("lever de soleil").imageCompression
-- #movieSetting
```

Voir aussi

`imageQuality`, `movieImageCompression`, `movieImageQuality`

imageEnabled

Syntaxe

```
sprite(quelleImageObjetFlashOuFormeVectorielle).imageEnabled
the imageEnabled of sprite quelleImageObjetFlashOuFormeVectorielle
member(quelleImageObjetFlashOuFormeVectorielle).imageEnabled
the imageEnabled of member quelleImageObjetFlashOuFormeVectorielle
```

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 :

```
global gImageObjetFluxContinu
on beginSprite me
    gImageObjetFluxContinu = me.spriteNum
    sprite(gImageObjetFluxContinu).imageEnabled = FALSE
end
```

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.

```
global gImageObjetFluxContinu
on exitFrame me
  if sprite(gImageObjetFluxContinu).member.percentStreamed < 100 then
    go to frame
  else
    sprite(gImageObjetFluxContinu).imageEnabled = TRUE
    updatestage
  end if
end
```

imageQuality

Syntaxe

```
member(quelActeur).imageQuality
the imageQuality of member quelActeur
```

Description

Cette propriété d'acteur bitmap indique le niveau de compression à utiliser lorsque la propriété `imageCompression` de l'acteur est définie sur `#jpeg`. 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.

Vous ne pouvez définir cette propriété que durant la programmation et elle n'affecte les acteurs que lors de l'enregistrement d'une animation au format Shockwave. Vous pouvez afficher un aperçu de l'image compressée en cliquant sur le bouton Optimiser sous Fireworks de l'onglet Bitmap de l'inspecteur des propriétés ou en choisissant la commande Aperçu dans le navigateur web du menu Fichier.

Si la propriété `imageCompression` d'un acteur image est définie sur `#MovieSetting`, c'est la propriété d'animation `movieImageQuality` qui sera utilisée, plutôt que `imageQuality`.

Voir aussi

`imageCompression`, `movieImageCompression`, `movieImageQuality`

immovable

Syntaxe

```
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)`

importFileInto

Syntaxe

```
importFileInto member quelActeur, nomDuFichier
importFileInto member quelActeur of castLib quelleDistribution, nomDuFichier
importFileInto member quelActeur, URL
```

Description

Commande ; remplace le contenu de l'acteur spécifié par *quelActeur* par le fichier indiqué par *nomDuFichier*.

La commande `importFileInto` est pratique dans les quatre situations suivantes :

- A la fin de la création d'une animation, elle permet d'intégrer des médias externes liés de façon à pouvoir les modifier pendant le projet.
- Lors de la création du scénario à partir de Lingo pendant la création de l'animation, elle permet d'affecter un contenu aux nouveaux acteurs créés.
- 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. Cependant, pour utiliser un fichier d'une URL et minimiser la durée de téléchargement, utilisez la commande `downloadNetThing` ou `preloadNetThing` pour télécharger le fichier sur un disque local avant d'importer le fichier depuis le disque local.
- 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 commande `importFileInto` dans les projections peut rapidement consommer la mémoire disponible et il est donc plus judicieux de réutiliser les mêmes acteurs pour les données importées.

La commande `importFileInto` ne fonctionne pas avec le lecteur Director pour Java. Pour changer le contenu d'un acteur bitmap ou audio dans une animation lue comme applet, liez les acteurs et changez la propriété `fileName` de l'acteur.

Remarque Dans Shockwave, vous devez émettre une commande `preloadNetThing` et attendre la fin du téléchargement avant d'utiliser `importFileInto` avec le fichier. Dans Director et les projections, `importFileInto` télécharge automatiquement le fichier.

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 :

```
on exitFrame
   URLtemporaire = "http://www.uneUrl.fr/couronne.gif"
  importFileInto new(#bitmap), URLtemporaire
end
```

L'instruction suivante remplace le contenu de l'acteur son Mémoire par le fichier audio Vent :

```
importFileInto member "Mémoire", "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 :

```
downLoadNetThing http://www.cbDeMille.com/FilmParlant.AIF, the \
  applicationPath&"FilmParlant.AIF" \
importFileInto(member "Norma Desmond parle", the
  applicationPath&"FilmParlant.AIF")
```

Voir aussi

downloadNetThing, fileName (propriété d'acteur), preloadNetThing()

in

Voir aussi

number (caractères), number (éléments), number (lignes), number (mots)

INF

Description

Valeur de renvoi de Lingo ; indique qu'une expression Lingo spécifiée est évaluée comme un nombre infini.

Voir aussi

NAN

inflate

Syntaxe

```
rectangle.Inflate(changementDeLargeur , changementDeHauteur)
inflate (rectangle, changementDeLargeur, changementDeHauteur)
```

Description

Commande ; change les dimensions du rectangle spécifié par *rectangle* par rapport au centre du rectangle, soit horizontalement (*changementDeLargeur*), soit verticalement (*changementDeHauteur*).

Le changement dans chaque direction est le double du nombre spécifié. Par exemple, le remplacement de *changementDeLargeur* par 15 augmente la largeur du rectangle de 30 pixels. Une valeur inférieure à 0 pour la dimension horizontale ou verticale réduit la taille du rectangle.

Exemples

L'instruction suivante augmente la largeur du rectangle de 4 pixels et sa hauteur de 2 pixels :

```
rect(10, 10, 20, 20).inflate(2, 1)
--rect (8, 9, 22, 21)
```

L'instruction suivante diminue la hauteur et largeur du rectangle de 20 pixels :

```
inflate (rect(0, 0, 100, 100), -10, -10)
--rect (10, 10, 90, 90)
```

ink

Syntaxe

```
sprite(quelleImageObjet).ink
the ink of sprite quelleImageObjet
```

Description

Propriété d'image-objet ; détermine l'effet d'encre appliqué à l'image-objet spécifiée par *quelleImageObjet*, de la façon suivante :

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

Pour une animation lue comme applet, les valeurs valides de la propriété d'image-objet `ink` varient en fonction des images-objets, comme suit :

- Pour les images-objets bitmap, la propriété d'image-objet `ink` peut être 0 (Copie), 8 (Dessin seul), 32 (Opacité) ou 36 (Fond transparent).
- Pour les images-objets forme vectorielle, Flash et forme, la propriété d'image-objet `ink` peut être 0, 8 ou 36.
- Pour les images-objets champ, la propriété d'image-objet `ink` peut être 0 ou 36. Le lecteur traite les encres Opacité et Dessin seul comme Fond transparent.

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 commande `updateStage` pour redessiner la scène. Si vous changez plusieurs propriétés d'images-objets – ou plusieurs images-objets – utilisez une seule commande `updateStage` à la fin de tous les changements.

Cette propriété peut être testée et définie.

Exemples

L'instruction suivante donne à la variable *encreCourante* la valeur de l'effet d'encre de l'image-objet 3 :

```
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 :

```
sprite(i + 1).ink = 8
```

Voir aussi

backColor, foreColor

inker (modificateur)

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).\
  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 au 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 au 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

Syntaxe

```
the inlineImeEnabled
```

Description

Propriété globale ; détermine si la fonction IME en ligne de Director est activée. Lorsqu'elle est définie sur `TRUE`, cette propriété permet à l'utilisateur d'entrer directement des caractères double octets dans les fenêtres Texte, Champ, Script et Messages de Director, sur les systèmes japonais.

Cette propriété peut être testée et définie. La valeur par défaut est définie par le paramètre Activer la fonction IME en ligne dans les préférences générales de Director.

insertBackdrop

Syntaxe

```
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 la position indiquée par le paramètre *index*. Ce fond est affiché dans l'image-objet 3D à l'*emplacementDansLimageObjet* avec la rotation indiquée. Le paramètre *emplacementDansLimageObjet* est l'emplacement 2D mesuré à partir du coin supérieur gauche de l'image-objet.

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("Séquence").texture("Cèdre")
sprite(5).camera.insertBackdrop(1, t1, point(300, 120), 45)
```

Voir aussi

`removeBackdrop`, `bevelDepth`, `overlay`

insertFrame

Syntaxe

```
insertFrame
```

Description

Commande ; 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 commande ne peut être utilisée que pendant une session d'enregistrement du scénario et remplit la même fonction que la commande `duplicateFrame`.

Exemple

Le gestionnaire suivant crée une image dans laquelle l'acteur transition Brouillard est affecté à la piste des transitions, suivi d'un groupe d'images vides. L'argument *nombreDimages* définit ce nombre d'images.

```
on animBalle nombreDimages
  beginRecording
    the frameTransition = member ("Brouillard").number
    go the frame + 1
    repeat with i = 0 to nombreDimages
      insertFrame
    end repeat
  endRecording
end
```

insertOverlay

Syntaxe

```
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 la position indiquée par le paramètre *index*. Ce recouvrement est affiché dans l'image-objet 3D à l'*emplacementDansLimageObjet* avec la *rotation* indiquée. Le paramètre *emplacementDansLimageObjet* est l'emplacement 2D mesuré à partir du coin supérieur gauche de l'image-objet.

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("Séquence").texture("Cèdre")
sprite(5).camera.insertOverlay(1, t1, point(300, 120), 45)
```

Voir aussi

removeOverlay, overlay, bevelDepth

inside()

Syntaxe

```
point.inside(rectangle)
inside(point, rectangle)
```

Description

Fonction ; indique si le point spécifié par *point* se trouve à l'intérieur du rectangle spécifié par *rectangle* (TRUE) ou à l'extérieur de celui-ci (FALSE).

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

Syntaxe

`installMenu` *`quelActeur`*

Description

Commande ; installe le menu défini dans l'acteur champ spécifié par *`quelActeur`*. 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 le Macintosh, il est possible de faire défiler les menus qui ne tiennent pas à l'écran.

Remarque Les menus ne sont pas disponibles dans Shockwave.

Exemples

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()

Syntaxe

(*`expressionNumérique`*).integer
integer(*`expressionNumérique`*)

Description

Fonction ; arrondit la valeur de *`expressionNumérique`* au nombre entier le plus proche.

Vous pouvez forcer un nombre entier en une chaîne avec la fonction `string()`.

Exemples

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()

Syntaxe

```
expression.integerP
(expressionNumérique).integerP
integerP(expression)
```

Description

Fonction ; indique si l'expression spécifiée par *`expression`* est un entier (`1` ou `TRUE`) ou non (`0` ou `FALSE`). Le *P* de la fonction `integerP` signifie *prédicat*.

Exemples

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()

Syntaxe

```
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`.

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()

Syntaxe

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()

Syntaxe

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 de *transformation2*, en fonction du 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.

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()

Syntaxe

```
rectangle1.intersect(rectangle2)
intersect(rectangle1, rectangle2)
```

Description

Fonction ; détermine le rectangle formé par l'intersection de *`rectangle1`* et *`rectangle2`*.

Exemple

L'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()`

interval

Syntaxe

```
member(quelActeurCurseur).interval
the interval of member quelActeurCurseur
```

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.

```
on mouseEnter
  member("Papillon").interval = 50
end

on mouseLeave
  member("Papillon").interval = 100
end
```

into

Ce fragment de code se retrouve dans un certain nombre d'expressions Lingo, telles que `put...into`.

inverse()

Syntaxe

```
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()`.

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()

Syntaxe

```
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()`.

Exemple

L'instruction suivante inverse la transformation du modèle Boîte.

```
member("Univers 3D").model("Boîte").transform.invert()
```

Voir aussi

`inverse()`

invertMask

Syntaxe

```
member(quelActeurQuickTime).invertMask
the invertMask of member quelActeurQuickTime
```

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 :

```
on basculeDuMasque
  member("Etoile").invertMask = not member("Etoile").invertMask
end
```

Voir aussi

mask

isBusy()

Syntaxe

```
sound(numéroDePiste).isBusy()
```

Description

Fonction ; renvoie TRUE si la piste audio *numéroDePiste* est en train de lire un son ou a mis celui-ci en pause, et FALSE si elle n'a pas entamé la lecture de l'un des sons placés en file d'attente, ou si cette lecture a été interrompue.

Vous pourrez voir un exemple de isBusy() dans une animation en consultant l'animation Sound Control du dossier Learning/Lingo_Examples, lui-même dans le dossier de Director.

Exemple

L'instruction Lingo suivante vérifie si un son est en cours de lecture sur la piste audio 1. Le cas échéant, le texte de l'acteur champ devient Vous devriez pouvoir entendre de la musique. Dans le cas contraire, le texte devient Le morceau est terminé.

```
if sound(1).isBusy() then
  member("champ").text = "Vous devriez pouvoir entendre de la musique."
else
  member("champ").text = "Le morceau est terminé."
end if
```

Voir aussi

pause() (lecture audio), playNext(), queue(), status, stop() (audio)

isInWorld()

Syntaxe

```
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.

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`

on isOKToAttach

Syntaxe

```
on isOKToAttach me, typeDimageObjet, numéroDimageObjet
```

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 :

```
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
```

isPastCuePoint()

Syntaxe

```
sprite(numéroDimageObjet).isPastCuePoint(IDduRepère)
isPastCuePoint(sprite(numéroDimageObjet), IDduRepère)
sound(numéroDePiste).isPastCuePoint(IDduRepère)
isPastCuePoint(sound(numéroDePiste), 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 *`numéroDimageObjet`* ou *`numéroDePiste`* 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.

Exemples

L'instruction suivante lit un son jusqu'au troisième passage sur le point de repère Fin du choeur :

```
if (isPastCuePoint(sound 1, "Fin du choeur")=3) then
  puppetSound 0
end if
```

L'exemple suivant permet d'afficher des informations dans l'acteur Champ 2 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.

```
if not sound(1).isPastCuePoint("Apogée") then
    member("champ 2").text = "Début du morceau."
  else
    member("champ 2").text = "Fin du morceau."
end if
```

isVRMovie

Syntaxe

```
member(quelActeur).isVRMovie
isVRMovie of member quelActeur
sprite(quelleImageObjet).isVRMovie
isVRMovie of sprite quelleImageObjet
```

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.

```
on vérifDeVR uneImageObjet
  if sprite(uneImageObjet).member.type = #quickTimeMedia then
    if sprite(uneImageObjet).isVRMovie then
      alert "Ceci est un élément QTVR."
    else
      alert "Ceci n'est pas un élément QTVR."
    end if
  else
    alert "Ceci n'est pas un élément QuickTime."
  end if
end
```

item...of

Syntaxe

```
expressionActeurTexte.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.

Exemples

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 (éléments)`, `word...of`

itemDelimiter

Syntaxe

```
the itemDelimiter
```

Description

Propriété système ; spécifie 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 le 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
```

kerning

Syntaxe

```
member(quelActeurTexte).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

Syntaxe

```
member(quelActeurTexte).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()

Syntaxe

```
the key
```

Description

Fonction ; indique la dernière touche sur laquelle l'utilisateur a appuyé. Cette valeur correspond à la valeur ANSI de la touche et non à sa valeur numérique.

Vous pouvez utiliser la fonction `the 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 `the key` n'est pas mise à jour si l'utilisateur appuie sur une touche lorsque Lingo exécute une boucle de répétition.

Utilisez l'animation Clavier et Lingo pour tester la correspondance des caractères des différentes touches sur différents claviers.

Exemples

Les instructions suivantes font revenir l'animation au repère du menu principal lorsque l'utilisateur appuie sur la touche Retour. 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 Retour a été enfoncée et, le cas échéant, revient au repère du menu principal.

```
on prepareMovie
  the keyDownScript = "vérifierLaTouche"
end prepareMovie
on vérifierLaTouche
  if the key = RETURN then go to frame "Menu principal"
end
```

Le gestionnaire `on keyDown` suivant vérifie si la dernière touche enfoncée est la touche Entrée et, le cas échéant, appelle le gestionnaire *`additionnerLesNombres`* :

```
on keyDown
  if the key = RETURN then additionnerLesNombres
end keyDown
```

Voir aussi

`commandDown`, `controlDown`, `keyCode()`, `optionDown`

keyboardFocusSprite

Syntaxe

```
set the keyboardFocusSprite = numéroDimageObjetTexte
```

Description

Propriété système ; 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. 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

`autoTab`, `editable`

keyCode()

Syntaxe

```
the keyCode
```

Description

Fonction ; indique le code numérique de la dernière touche sur laquelle l'utilisateur a appuyé. Ce code de clavier correspond à la valeur numérique de la touche et non à sa valeur ANSI.

Remarque Lorsque l'animation est lue sous forme d'applet, cette fonction renvoie uniquement les valeurs des touches de fonction et des touches fléchées.

Vous pouvez utiliser la fonction `keyCode` pour détecter si l'utilisateur a appuyé sur une touche fléchée ou une touche de fonction, qui ne peuvent pas être spécifiées à l'aide de la fonction `key`.

Utilisez l'animation Clavier et Lingo pour tester la correspondance des caractères des différentes touches sur différents claviers.

Cette fonction peut être testée, mais pas définie.

Exemples

Le gestionnaire suivant utilise la fenêtre Messages pour afficher le code approprié chaque fois que l'utilisateur appuie sur une touche :

```
on enterFrame
  the keyDownScript = "put the keyCode"
end
```

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 :

```
if the keyCode = 126 then go to marker(-1)
```

Le gestionnaire suivant vérifie si l'une des touches fléchées a été enfoncée et, le cas échéant, répond en conséquence :

```
on keyDown
  case (the keyCode) of
    123: verslaGauche
    126: enAvant
    125: enArrière
    124: verslaDroite
  end case
end
```

Voir aussi

`commandDown`, `controlDown`, `key()`, `optionDown`

on keyDown

Syntaxe

```
on keyDown
  instruction(s)
end
```

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 Lingo 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.

Le lecteur Director pour Java ne répond aux messages `keyDown` que si l'animation est activée dans le navigateur. L'utilisateur doit cliquer dans l'applet pour que celle-ci puisse recevoir les touches sur lesquelles l'utilisateur appuie.

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 :

```
on keyDown
  if the key = RETURN then go to frame "additionnerLaSomme"
end keyDown
```

Voir aussi

`charToNum()`, `keyDownScript`, `keyUpScript`, `key()`, `keyCode()`, `keyPressed()`

keyDownScript

Syntaxe

```
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 encadrée de guillemets droits et peut être une instruction simple ou un 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`.

Exemples

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`

keyPressed()

Syntaxe

```
the keyPressed
keyPressed (codeDeLaTouche)
keyPressed (chaîneDeCaractèresAscii)
```

Description

Fonction ; renvoie le caractère de la touche sur laquelle l'utilisateur a appuyé en dernier si aucun argument n'est utilisé. Le résultat est renvoyé sous forme de chaîne. Si l'utilisateur n'a appuyé sur aucune touche, `the keyPressed` est une chaîne vide.

Si un argument est utilisé, entrez le code de la touche ou la chaîne ASCII qui lui correspond. Dans les deux cas, la valeur renvoyée est `TRUE` si la touche en question est enfoncée ou `FALSE` dans le cas contraire.

Le lecteur Director pour Java ne supporte pas cette propriété. Par conséquent, une animation lue sous forme d'applet ne peut pas détecter la touche sur laquelle l'utilisateur a appuyé lorsque Lingo exécute une boucle de répétition.

La propriété `keyPressed` est mise à jour chaque fois que l'utilisateur utilise le clavier alors que Lingo est dans une boucle de répétition. Ceci constitue un avantage par rapport à la fonction `key`, qui n'est pas mise à jour lorsque Lingo est dans une boucle de répétition.

Utilisez l'animation Clavier et Lingo pour tester la correspondance des caractères des différentes touches sur différents claviers.

Cette propriété peut être testée, mais pas définie.

Exemples

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`* :

```
if the keyPressed = RETURN then 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 :

```
if keyPressed(0) then 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 :

```
if keyPressed("a") and keyPressed ("b") then put "Les touches sont enfoncées"
```

Voir aussi

`keyCode()`, `key()`

on keyUp

Syntaxe

```
on keyUp
  instruction(s)
end
```

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 Lingo 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.

Le lecteur Director pour Java ne répond aux messages `keyUp` que si l'animation est activée dans le navigateur. L'utilisateur doit cliquer dans l'applet pour que celle-ci puisse recevoir les touches sur lesquelles l'utilisateur appuie.

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 :

```
on keyUp
  if the key = RETURN then go to frame "additionnerLaSomme"
end keyUp
```

Voir aussi

`on keyDown`, `keyDownScript`, `keyUpScript`

keyUpScript

Syntaxe

```
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 encadrée de guillemets droits et peut être une instruction simple ou un 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`.

Exemples

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`

label()

Syntaxe

```
label(expression)
```

Description

Fonction ; indique l'image associée au repère spécifié par le terme *expression*. Le terme *expression* doit être le nom d'un repère de l'animation courante. Si ce n'est pas le cas, cette fonction renvoie 0.

Exemples

L'instruction suivante place la tête de lecture sur la dixième image après le repère intitulé Départ :

```
go to label("Départ") + 10
```

L'instruction suivante affecte le numéro d'image du quatrième élément de la liste des repères à la variable *quelleImage* :

```
quelleImage = label(the labelList.line[4])
```

Voir aussi

go, frameLabel, labelList, marker(), play

labelList

Syntaxe

```
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.

Exemples

L'instruction suivante fait de la liste des libellés d'images le contenu de l'acteur champ Images-clés :

```
member("Images-clés").text = the labelList
```

Le gestionnaire suivant détermine le libellé de l'image démarrant la scène en cours :

```
on chercherDernierLibellé
  aa = label(0)
  repeat with i = 1 to (the labelList.line.count - 1)
    if aa = label(the labelList.line[i]) then
      return the labelList.line[i]
    end if
  end repeat
end
```

Voir aussi

frameLabel, label(), marker()

last()

Syntaxe

```
the last sousChaîne of ( expressionSousChaîne )
the last sousChaîne in (expressionSousChaîne)
```

Description

Fonction ; identifie la dernière sous-chaîne de caractères spécifiée par *sousChaîne* dans l'expression spécifiée par *expressionSousChaîne*.

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.

Exemples

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`

lastChannel

Syntaxe

```
the 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.

Cette propriété peut être testée, mais pas définie.

Vous pourrez voir un exemple de `lastChannel` dans une animation en consultant l'animation QT and Flash du dossier Learning/Lingo_Examples, 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 :

```
put the lastChannel
```

lastClick()

Syntaxe

```
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.

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`, `startTimer`

lastError

Syntaxe

```
sprite(quelleImageObjet).lastError
member(quelActeur).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.

Exemples

Les exemples suivants indiquent que la dernière erreur renvoyée par RealPlayer pour l'image-objet 2 et l'acteur Real était `#PNR_OUTOFMEMORY` :

```
put sprite(2).lastError
-- #PNR_OUTOFMEMORY

put member("Real").lastError
-- #PNR_OUTOFMEMORY
```

lastEvent()

Syntaxe

```
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.

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`, `startTimer`

lastFrame

Syntaxe

```
the lastFrame
```

Description

Propriété d'animation ; affiche le numéro de la dernière image de l'animation.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante affiche le numéro de la dernière image de l'animation dans la fenêtre Messages :

```
put the lastFrame
```

lastKey

Syntaxe

```
the lastKey
```

Description

Propriété système ; indique le nombre de battements (1 battement = 1/60ème de seconde) écoulés depuis que l'utilisateur a appuyé sur une touche.

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 :

```
if the lastKey > 10 * 60 then go to "Pas de touche"
```

Voir aussi

`lastClick()`, `lastEvent()`, `lastRoll`, `startTimer`

lastRoll

Syntaxe

```
the lastRoll
```

Description

Propriété système ; indique le nombre de battements (1 battement = 1/60ème de seconde) écoulés depuis que l'utilisateur a déplacé la souris.

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 :

```
if the lastRoll > 45 * 60 then go to "Boucle"
```

Voir aussi

`lastClick()`, `lastEvent()`, `lastKey`, `startTimer`

left

Syntaxe

```
sprite(quelleImageObjet).left
the left of sprite quelleImageObjet
```

Description

Propriété d'image-objet ; identifie la coordonnée horizontale gauche du rectangle de délimitation de l'image-objet spécifiée par *quelleImageObjet*.

Les coordonnées d'images-objets sont exprimées en pixels, (0,0) correspond au coin supérieur gauche de la scène.

Lorsqu'une animation est lue sous forme d'applet, la valeur de cette propriété est calculée à partir du côté gauche de l'applet.

Cette propriété peut être testée et définie.

Exemples

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* :

```
if sprite(3).left < 0 then 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* :

```
set vPlusbas = sprite (i + 1).left
```

Voir aussi

`bottom`, `height`, `locH`, `locV`, `right`, `top`, `width`

left (3D)

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).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

Syntaxe

`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()

Syntaxe

```
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.

Exemples

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 :

```
if member("Nom de fichier").text.length > 31 then
  alert "Ce nom de fichier est trop long."
end if
```

Voir aussi

`chars()`, `offset()` (fonction de chaîne)

length (3D)

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).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

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).\
  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

Syntaxe

```
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

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).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

Syntaxe

```
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.

Vous trouverez une liste complète des propriétés et commandes de lumières dans Chapitre 2, Lingo 3D par fonction, page 33.

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

line...of

Syntaxe

```
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.

Exemples

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 (mots)

lineColor

Syntaxe

```
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

Syntaxe

```
member(quelActeur).lineCount
the lineCount of member quelActeur
```

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* :

```
NombreDeLignes = member("Nouvelles du jour").lineCount
```

lineDirection

Syntaxe

```
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() (fonction)

Syntaxe

```
member(quelActeur).lineHeight(numéroDeLigne)
lineHeight(member quelActeur, numéroDeLigne)
```

Description

Fonction ; renvoie la hauteur, en pixels, d'une ligne spécifique de l'acteur champ spécifié.

Exemple

L'instruction suivante détermine la hauteur, en pixels, de la première ligne de l'acteur champ Nouvelles du jour et affecte le résultat à la variable *titre* :

```
titre = member("Nouvelles du jour").lineHeight(1)
```

lineHeight (propriété d'acteur)

Syntaxe

```
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

Syntaxe

```
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`

linePosToLocV()

Syntaxe

```
member(quelActeur).linePosToLocV(numéroDeLigne )
linePosToLocV(member quelActeur, numéroDeLigne)
```

Description

Fonction ; renvoie la distance, en pixels, d'une ligne spécifique à partir du bord supérieur de l'acteur champ spécifié.

Exemple

L'instruction suivante 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* :

```
débutDeChaîne = member("Nouvelles du jour").linePosToLocV(2)
```

lineSize

Syntaxe

```
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, celle-ci doit être asservie.

Cette propriété peut être testée et définie.

Exemples

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
```

linkAs()

Syntaxe

```
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, animation Shockwave ou animation Java. Ceci diffère des autres médias liés, qui restent externes à l'animation, à moins que vous ne les importiez explicitement.

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
```

linked

Syntaxe

```
member(quelActeur).linked
the linked of member quelActeur
```

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 distribution Director (`FALSE`). Lorsque les données sont stockées dans un fichier externe, la propriété `pathName` de l'acteur doit indiquer l'emplacement du fichier de l'animation.

Cette propriété peut être testée et définie pour les acteurs script, Flash et GIF. Elle peut être testée pour tous les types d'acteurs.

Exemple

L'instruction suivante convertit l'acteur Flash Amis provenant d'un acteur lié, en un acteur stocké dans un fichier interne :

```
member("amis").linked = 0
```

Voir aussi

`fileName (propriété d'acteur)`, `pathName (propriété d'acteur)`

list()

Syntaxe

`list(`*`valeur1`*`, `*`valeur2`*`, `*`valeur3`*`...)`

Description

Fonction et type de données ; définit une liste linéaire composée des valeurs spécifiées par *`valeur1`*, *`valeur2`*, *`valeur3`*, etc. Permet de créer une liste sans utiliser de crochets d'accès (`[ ]`).

La longueur maximale d'une ligne de code Lingo exécutable est de 256 caractères. Cette commande ne permet pas de listes très longues. Si vous avez une quantité importante de données à inclure dans une liste, encadrez-les de crochets et placez-les dans un champ. Vous pourrez ensuite affecter ce champ à une variable. Le contenu de cette variable deviendra une liste de données.

Exemple

L'instruction suivante associe la variable *`Designers`* à une liste linéaire contenant les noms Jean, Pierre et Marc :

```
Designers = list("Jean", "Pierre", "Marc")
```

Le résultat est la liste ["Jean", "Pierre", "Marc"].

Voir aussi

`integer()`, `integerP()`, `value()`

listP()

Syntaxe

`listP(`*`élément`*`)`

Description

Fonction ; indique si l'élément spécifié par *`élément`* est une liste, un rectangle ou un point (`1` ou `TRUE`) ou non (`0` ou `FALSE`).

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()`

loaded

Syntaxe

```
member(quelActeur).loaded
the loaded of member quelActeur
```

Description

Propriété d'acteur ; indique si l'acteur spécifié par *quelActeur* est chargé en mémoire (TRUE) ou non (FALSE).

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.

Cette propriété peut être testée, mais pas définie.

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 :

```
if member("Démonstration").loaded = FALSE then go to movie("Attente")"
```

Voir aussi

preLoad (commande), ramNeeded(), size, unLoad

loadFile()

Syntaxe

```
member(quelActeur).loadFile(nomDeFichier {, écraser?, \
  générerDesNomsUniques?})
```

Description

Commande d'acteur 3D ; importe les actifs du fichier *.w3d, *nomDeFichier*, dans l'acteur.

Le paramètre facultatif *écraser?* 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.

Si le paramètre facultatif *générerDesNomsUniques?* a pour valeur TRUE, tout élément du fichier *.w3d portant le même nom qu'un élément correspondant de l'acteur sera renommé. Si *générerDesNomsUniques?* a pour valeur FALSE, les éléments de l'acteur seront remplacés par les éléments correspondants du même nom dans le fichier *.w3d. La valeur par défaut de *générerDesNomsUniques?* est TRUE.

La propriété state de l'acteur doit avoir pour valeur –1 (erreur) ou 4 (chargé) avant l'utilisation de la commande loadFile.

Exemples

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)`

loc

Syntaxe

```
sprite quelleImageObjet.loc
the loc of sprite quelleImageObjet
```

Description

Propriété d'image-objet ; détermine les coordonnées sur la scène du point d'alignement de l'image-objet spécifiée. La valeur est exprimée sous forme d'un point.

Cette propriété peut être testée et définie.

Vous pourrez voir un exemple de `loc` dans une animation en consultant l'animation Imaging du dossier Learning/Lingo_Examples, lui-même dans le dossier de Director.

Exemple

L'instruction suivante vérifie les coordonnées sur la scène de l'image-objet 6. Le résultat est le point (50, 100) :

```
put sprite(6).loc
-- point(50, 100)
```

Voir aussi

`bottom`, `height`, `left`, `locH`, `locV`, `right`, `top`, `width`

loc (fond et recouvrement)

Syntaxe

```
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

Syntaxe

```
sprite(quelleImageObjet).locH
the locH of sprite quelleImageObjet
```

Description

Propriété d'image-objet ; indique la position horizontale du point d'alignement de l'image-objet spécifiée. Les coordonnées des images-objets sont calculées par rapport au coin supérieur gauche de la scène.

Cette propriété peut être testée et définie. Pour que la valeur définie par Lingo dure au-delà de l'image-objet courante, celle-ci doit être asservie.

Exemples

L'instruction suivante vérifie si la position horizontale du point d'alignement de l'image-objet 9 se trouve à droite du bord droit de la scène et, le cas échéant, place le bord droit de l'image-objet sur le bord de la scène :

```
if sprite(9).locH > (the stageRight - the stageLeft) then
  sprite(9).locH = 0
end if
```

L'instruction suivante place l'image-objet 15 à l'emplacement horizontal auquel l'utilisateur a cliqué :

```
sprite(15).locH = the mouseH
```

Voir aussi

bottom, height, left, loc, locV, point(), right, top, updateStage, width

locToCharPos()

Syntaxe

```
member(quelActeur). locToCharPos(emplacement )
locToCharPos(member quelActeur, emplacement)
```

Description

Fonction ; renvoie le numéro du caractère de l'acteur champ spécifié qui est le plus proche du point spécifié par *emplacement*. La valeur de *emplacement* est un point calculé par rapport au coin supérieur gauche de l'acteur champ.

La valeur 1 correspond au premier caractère de la chaîne, la valeur 2 au second caractère de cette chaîne, et ainsi de suite.

Exemples

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*.

```
miseEnPage = member("Nouvelles du jour").locToCharPos(point(100, 100))
```

Le gestionnaire suivant indique le caractère qui se trouve sous le curseur lorsque l'utilisateur clique sur l'image-objet champ Informations :

```
on mouseDown
  put member("Informations").locToCharPos(the clickLoc - \
  (sprite(the clickOn).loc))
end
```

lockTranslation

Syntaxe

```
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 suite à 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

Syntaxe

```
sprite(quelleImageObjet).locV
the locV of sprite quelleImageObjet
```

Description

Propriété d'image-objet ; indique la position verticale du point d'alignement de l'image-objet spécifiée. Les coordonnées des images-objets sont calculées par rapport au coin supérieur gauche de la scène.

Cette propriété peut être testée et définie. Pour que la valeur définie par Lingo dure au-delà de l'image-objet courante, celle-ci doit être asservie.

Exemple

L'instruction suivante vérifie si la position verticale du point d'alignement de l'image-objet 9 se trouve en dessous du bas de la scène et, le cas échéant, déplace l'image-objet vers le haut de la scène :

```
if sprite(9).locV > (the stageBottom - the stageTop) then
  sprite(9).locV = 0
end if
```

Exemple

L'instruction suivante place l'image-objet 15 à la position verticale où l'utilisateur a cliqué :

```
sprite(15).locV = the mouseV
```

Voir aussi

`bottom`, `height`, `left`, `loc`, `locH`, `point()`, `right`, `top`, `updateStage`, `width`

locVToLinePos()

Syntaxe

```
member(quelActeur).locVToLinePos(emplacementV )
locVToLinePos(member quelActeur, emplacementV)
```

Description

Fonction; renvoie le numéro de la ligne de caractères apparaissant à la position verticale spécifiée par *`emplacementV`*. La valeur de *`emplacementV`* représente le nombre de pixels depuis le haut de l'acteur champ et non depuis la partie de l'acteur champ affichée sur la scène.

Exemple

L'instruction suivante détermine la ligne de caractères apparaissant à 150 pixels du haut de l'acteur Nouvelles du jour et affecte le résultat à la variable *`sautDePage`* :

```
sautDePage = member("Nouvelles du jour").locVToLinePos(150)
```

locZ of sprite

Syntaxe

```
sprite(quelleImageObjet).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.

Cette propriété peut être testée et définie.

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 dans 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`.

```
on mouseUp me
  global gImageObjetPlusElevée
  sprite(me.spriteNum).locZ = gImageObjetPlusElevée + 1
  gImageObjetPlusElevée = gImageObjetPlusElevée + 1
end
```

Voir aussi

`locH`, `locV`

lod (modificateur)

Syntaxe

`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`

log()

Syntaxe

`log(`*`nombre`*`)`

Description

Fonction mathématique ; calcule le logarithme naturel du nombre spécifié par *`nombre`*, qui doit être un nombre décimal 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 en affecte le résultat à la variable *`Réponse`*.

```
Réponse = log(Nombre.sqrt)
```

long

Consultez

`date() (horloge du système)`, `time()`

loop (propriété d'acteur)

Syntaxe

`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)

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).\
  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 (mot-clé)

Syntaxe

```
loop
```

Description

Mot-clé ; fait référence au repère. Le mot-clé `loop`, lorsque utilisé avec la commande `go to`, équivaut à l'instruction `go to the marker`.

Exemple

Le gestionnaire suivant fait boucler l'animation entre le repère précédent et l'image courante :

```
on exitFrame
  go loop
end exitFrame
```

loop (propriété d'acteur)

Syntaxe

```
member(quelActeur).loop
the loop of member quelActeur
```

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 :

```
member("Démo").loop = 1
```

loop (propriété Flash)

Syntaxe

```
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
```

loopBounds

Syntaxe

```
sprite(quelleImageObjetQuickTime).loopBounds
the loopBounds of sprite quelleImageObjetQuickTime
```

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 : [*positionDeDépart*, *positionDeFin*].

Les paramètres *positionDeDépart* et *positionDeFin* 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).

```
on beginSprite me
  sprite(me.spriteNum).loopBounds = [(16 * 60),(32 * 60)]
end
```

loopCount

Syntaxe

```
sound(numéroDePiste).loopCount
the loopCount of sound numéroDePiste
```

Description

Propriété d'acteur ; définit le nombre d'occurrences de lecture en boucle du son courant de la piste audio *numéroDePiste*. La valeur par défaut 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 commande `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 de `boucle` 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`.

```
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:8000, #loopCount:3])
  sound(2).play()
end
```

Exemple

Le gestionnaire suivant affiche un message d'alerte indiquant le nombre de lectures en boucle de l'acteur son 2. Si aucune boucle n'a été définie dans le son courant de la piste audio 2, la valeur `sound(2).loopCount` repasse à 1.

```
on afficherLeNombreDeBoucles
  alert "La boucle de ce son est lue" && sound(2).loopCount && "fois."
end
```

Voir aussi

`breakLoop()`, `setPlaylist()`, `loopEndTime`, `loopsRemaining`, `loopStartTime`, `queue()`

loopEndTime

Syntaxe

```
sound(numéroDePiste).loopEndTime
the loopEndTime of sound numéroDePiste
```

Description

Propriété audio ; position temporelle de fin, en millisecondes, de la boucle définie dans le son courant de la piste audio *numéroDePiste*. Il s'agit 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.

```
on lireLaMusique
  sound(2).play([#member:member("Intro"), #startTime:3000,\
  #loopCount:5,#loopStartTime:8000, #loopEndTime:8900])
end
```

Exemple

Le gestionnaire suivant entraîne l'affichage de Qu'est-ce que c'est que ce truc ? dans le champ texte Au secours lorsque la valeur `currentTime` de la piste audio 2 est comprise entre les valeurs `loopStartTime` et `loopEndTime`.

```
on idle
  if sound(2).currentTime > sound(2).loopStartTime and \
  sound(2).currentTime < sound(2).loopEndTime then
    member("Cris").text = "Au secours!"
  else
    member("Tim").text = "Qu'est-ce que c'est que ce truc ?"
  end if
end
```

Voir aussi

`breakLoop()`, `getPlaylist()`, `loopCount`, `loopsRemaining`, `loopStartTime`, `queue()`

loopsRemaining

Syntaxe

```
sound(numéroDePiste).loopsRemaining
the loopsRemaining of sound(numéroDePiste)
```

Description

Propriété en lecture seule ; nombre d'occurrences restantes de lecture en boucle du son courant sur la piste audio *numéroDePiste*. 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 commande `queue()` ou `setPlayList()`.

Voir aussi

`breakLoop()`, `loopCount`, `loopEndTime`, `loopStartTime`, `queue()`

loopStartTime

Syntaxe

```
sound(numéroDePiste).loopStartTime
the loopStartTime of sound(numéroDePiste)
```

Description

Propriété d'acteur ; position temporelle de début, en millisecondes, de la boucle définie pour le son courant lu par l'*objetAudio*. 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 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.

```
on lireLaMusique
  sound(2).play([#member:member("Intro"), #startTime:3000,\
  #loopCount:5,#loopStartTime:8000, #loopEndTime:8900])
end
```

Exemple

Le gestionnaire suivant entraîne l'affichage de Qu'est-ce que c'est que ce truc ? dans le champ texte Au secours lorsque la valeur `currentTime` de la piste audio 2 est comprise entre les valeurs `loopStartTime` et `loopEndTime` :

```
on idle
  if sound(2).currentTime > sound(2).loopStartTime and \
  sound(2).currentTime < sound(2).loopEndTime then
    member("Cris").text = "Au secours!"
  else
    member("Tim").text = "Qu'est-ce que c'est que ce truc ?"
  end if
end
```

Voir aussi

`breakLoop()`, `setPlaylist()`, `loopCount`, `loopEndTime`, `loopsRemaining`, `queue()`

magnitude

Syntaxe

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()

makeList()

Syntaxe

objetDanalyse.makeList()

Description

Fonction ; renvoie une liste de propriétés basée sur le document XML analysé à l'aide de `parseString()` ou `parseURL()`.

Exemple

Le gestionnaire suivant analyse un document XML et renvoie la liste résultante :

```
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 " && errorString
    exit
  end if
  return listeAnalysée
end
```

Voir aussi

makeSubList()

makeSubList()

Syntaxe

```
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.

Exemple

Avec le code XML suivant :

```
<?xml version="1.0"?>
  <e1>
    <nomDeBalise attr1="val1" attr2="val2"/>
    <e2>élément 2</e2>
    <e3>élément 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()

Syntaxe

```
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`.

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)

Syntaxe

```
member(quelActeur).motion(quelMouvement).\
  map(quelAutreMouvement{, nomDeSegment})
```

Description

Commande 3D de mouvement ; mappe le mouvement spécifié par *quelAutreMouvement* au mouvement courant (*quelMouvement*) et l'applique au segment spécifié par *nomDeSegment* et à tous ses enfants. 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.

Le paramètre *nomDeSegment* prend par défaut la valeur du segment racine si aucun segment n'est spécifié.

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()

Syntaxe

```
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.

Voir aussi

`map()`, `mapStageToMember()`

mapStageToMember()

Syntaxe

```
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, la valeur `VOID` est renvoyée.

Voir aussi

`map()`, `mapMemberToStage()`

margin

Syntaxe

```
member(quelActeur).margin
the margin of member quelActeur
```

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 :

```
member("Nouvelles du Jour").margin = 15
```

marker()

Syntaxe

```
marker(expressionNombreEntier)
marker("chaîne")
```

Description

Fonction ; renvoie le numéro des repères situés avant et après l'image courante. Elle est utile pour implémenter un bouton Suivant ou Précédent ou pour définir une boucle d'animation.

L'argument *expressionNombreEntier* peut être un nombre entier positif ou négatif, ou 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 l'argument de `marker` est une chaîne, `marker` renvoie le numéro de la première image dont le repère correspond à la chaîne.

Exemples

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 :

```
go to marker(0)
```

L'instruction suivante donne à la variable *repèreSuivant* la même valeur que celle du repère suivant de la distribution :

```
repèreSuivant = marker(1)
```

Voir aussi

`go`, `frame() (fonction)`, `frameLabel`, `label()`, `labelList`

the markerList

Syntaxe

```
the markerList
```

Description

Propriété globale ; contient une liste de propriétés Lingo des repères du scénario. La liste est au format :

```
numéroDimage: "numéroDeRepère"
```

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante affiche la liste des repères dans la fenêtre Messages :

```
put the markerList
-- [1: "Ouverture", 15: "Menu principal", 26: "Fermeture"]
marker()
```

mask

Syntaxe

```
member(quelActeurQuickTime).mask
the mask of member quelActeurQuickTime
```

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 :

```
on prepareFrame
  member("Voyeur").mask = member("Serrure")
end
```

Voir aussi

`invertMask`

max()

Syntaxe

```
liste.max()
max(liste)
max(valeur1, valeur2, valeur3, ...)
```

Description

Fonction ; renvoie la valeur la plus élevée de la liste spécifiée ou de la 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.

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.

```
on rechercheDuGagnant Devis
  Gagnant = Devis.max()
  member("Félicitations").text = \
  "Vous avez gagné, avec un devis de " & Gagnant &"euros !"
end
```

maxInteger

Syntaxe

```
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

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).\
  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`

mci

Syntaxe

```
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.

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"
```

me

Syntaxe

```
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 dans 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_Examples, lui-même dans le dossier de Director.

Exemples

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

media

Syntaxe

```
member(quelActeur).media
the media of member quelActeur
```

Description

Propriété d'acteur ; identifie l'acteur spécifié comme un jeu de numéros. La définition de la propriété d'acteur `media` utilisant une grande quantité de mémoire, il est préférable de ne l'utiliser que pendant la programmation.

Vous pouvez utiliser la propriété d'acteur `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 d'acteur `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 :

```
member("Aube").media = member("leverDeSoleil").media
```

Voir aussi

type (propriété d'acteur), media

mediaReady

Syntaxe

```
member(quelActeur).mediaReady
the mediaReady of member quelActeur
sprite(quelNuméroDimageObjet).mediaReady
the mediaReady of sprite quelNuméroDimageObjet
```

Description

Propriété d'acteur ou d'image-objet ; détermine si le contenu d'une image-objet, d'un acteur, d'un fichier d'animation ou de distribution ou d'un acteur lié a été complètement téléchargé depuis Internet et est disponible sur le disque dur local (`TRUE`) ou non (`FALSE`).

Cette propriété n'est utile que lors du transfert en flux continu d'un fichier d'animation ou 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 fonction `mediaReady`, consultez l'animation Shockwave en flux continu dans l'Aide de Director.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante change les acteurs lorsque l'acteur souhaité est récupéré et disponible localement :

```
if member("arrière-plan").mediaReady = TRUE then
   sprite(2).memberNum = 10
  -- 10 est le numéro de l'acteur "arrière-plan"
end if
```

Voir aussi

`frameReady()`

mediaStatus

Syntaxe

```
sprite(quelleImageObjet).mediaStatus
member(quelActeur).mediaStatus
```

Description

Propriété d'image-objet ou d'acteur RealMedia ; permet d'obtenir un symbole représentant l'état du train RealMedia. Cette propriété peut être testée, mais pas définie, et est dynamique en cours de lecture.

Cette propriété peut avoir les valeurs suivantes :

- `#closed` indique que l'acteur RealMedia n'est pas actif. La valeur de `mediaStatus` reste `#closed` jusqu'au début de la lecture.
- `#connecting` indique qu'une connexion au train RealMedia est en cours d'établissement.
- `#opened` indique qu'une connexion au train RealMedia a été établie et est ouverte. Il s'agit d'un état transitoire, rapidement suivi par `#buffering`.
- `#buffering` indique que le train RealMedia 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 train RealMedia démarre si la propriété `pausedAtStart` a pour valeur `FALSE`. Pour plus d'informations, consultez `percentBuffered`.

- `#playing` indique que le train RealMedia 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 par l'invocation de la méthode `pause` dans Lingo.
- `#error` indique que le train 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`. Pour plus d'informations, consultez l'entrée `state (RealMedia)`.

Exemples

Les exemples suivants indiquent que l'élément RealMedia de l'image-objet 2 et l'acteur Real sont en cours de lecture.

```
put sprite(2).mediaStatus
-- #playing

put member("Real").mediaStatus
-- #playing
```

Voir aussi

`state (RealMedia)`, `percentBuffered`, `lastError`

member

Syntaxe

```
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 (mot-clé)

Syntaxe

```
member quelActeur
member quelActeur of castLib quelleDistribution
member(quelActeur, quelleDistribution)
```

Description

Mot-clé ; indique que l'objet spécifié par *quelActeur* est un acteur. Lorsque *quelActeur* est une chaîne, elle est utilisée comme nom d'acteur. Lorsque *quelActeur* est un nombre entier, il est utilisé comme numéro d'acteur.

Lors de la lecture d'une animation sous forme d'applet, il est préférable de faire référence aux acteurs par leurs numéros plutôt que par leurs noms, cela améliorant les performances de l'applet.

Lorsque utilisé seul, le mot-clé `member` est une référence spécifique à une distribution et à un de ses acteurs :

```
put sprite(12).member
-- (member 3 of castLib 2)
```

Cette propriété est différente de la propriété d'image-objet `memberNum`, qui est toujours un nombre entier désignant la position d'un acteur dans une distribution, mais qui ne spécifie pas la distribution :

```
put sprite(12).memberNum
-- 3
```

Le numéro d'un acteur est également une référence absolue à un acteur spécifique d'une distribution particulière :

```
put sprite(12).member.number
-- 131075
```

Exemples

L'instruction suivante donne à la propriété `hilite` de l'acteur bouton Appel d'offres la valeur `TRUE` :

```
member("Appel d'offres").hilite = TRUE
```

L'instruction suivante place le nom de l'acteur son 132 dans la variable *nomDuSon* :

```
put member(132, "Viva Las Vegas").name
```

L'instruction suivante vérifie le type de l'acteur Portrait de Napoléon dans la distribution Empereurs :

```
typeDacteur = member("Portrait de Napoléon", "Empereurs").type
```

L'instruction suivante détermine si l'acteur 9 possède un nom :

```
if member(9).name = EMPTY then exit
```

Vous pouvez vérifier l'existence d'un acteur en testant son numéro :

```
vérifDeLacteur = member("Epiphanie").number
if vérifDeLacteur = -1 then alert "Désolé, cet acteur n'existe pas."
```

Vous pouvez également vérifier l'existence d'un acteur en testant son type :

```
vérifDeLacteur = member("Epiphanie").type
if vérifDeLacteur = #empty then alert "Désolé, cet acteur n'existe pas."
```

Voir aussi

memberNum

member (propriété audio)

Syntaxe

```
sound(numéroDePiste).member
the member of sound(numéroDePiste)
```

Description

Cette propriété en lecture seule est l'acteur son en cours de lecture dans la piste audio *numéroDePiste*. Cette propriété renvoie une valeur nulle si aucun son n'est en cours de lecture.

Exemple

L'instruction suivante affiche le nom de l'acteur correspondant au son lu dans la piste 2 dans la fenêtre Messages :

```
put sound(2).member
-- (member 4 of castLib 1)
```

Voir aussi

getPlaylist(), queue()

member (propriété d'image-objet)

Syntaxe

```
sprite(quelleImageObjet).member
the member of sprite quelleImageObjet
```

Description

Propriété d'image-objet ; spécifie l'acteur et la distribution d'une image-objet.

La propriété d'image-objet `member` est différente de la propriété d'image-objet `memberNum`, qui ne spécifie que le numéro de l'image-objet pour identifier sa position dans la distribution, mais ne spécifie pas la distribution elle-même. La propriété d'image-objet `member` est également différente des propriétés `mouseMember` et `castNum` (obsolète), qui ne spécifient pas non plus la distribution de l'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 distribution (`sprite(x).member = member(A, B)`).
- Spécifiez le nom de l'acteur (`sprite(x).member = member ("MELODIE ")`).
- Spécifiez le nombre entier unique qui inclut toutes les bibliothèques de distributions et correspond à la fonction `mouseMember` (`sprite(x).member = 132`).

Si vous n'utilisez que le nom de l'acteur, Director trouve le premier acteur portant ce nom dans toutes les distributions courantes. Si ce nom se répète dans deux 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 distribution sans affecter la distribution de l'image-objet (`set the memberNum of sprite x to 132`).

Vous pouvez déterminer la propriété d'image-objet `memberNum` à partir de la propriété d'image-objet `member` à l'aide de l'instruction `the number of the member of sprite x`. Vous pouvez également récupérer d'autres propriétés d'acteurs à l'aide d'instructions telles que `the name of the member of sprite x` ou `the rect of the member of sprite x`.

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 d'image-objet `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. Bien que vous puissiez changer toutes les propriétés d'images-objets avec la propriété d'image-objet `type`, 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.

Cette propriété peut être testée et définie.

Exemples

L'instruction suivante affecte l'acteur 3 de la distribution 4 à l'image-objet 15 :

```
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 :

```
on exitFrame
  MM = the mouseMember
  target = sprite(1).member
  if target = MM then put "au-dessus de la zone référencée"
  go the frame
end
```

Voir aussi

`castLibNum`, `memberNum`

memberNum

Syntaxe

```
sprite(quelleImageObjet).memberNum
the memberNum of sprite quelleImageObjet
```

Description

Propriété d'image-objet ; identifie la position de l'acteur (mais n'identifie pas sa distribution) associé à l'image-objet spécifiée par *`quelleImageObjet`*. Sa valeur est le numéro de l'acteur uniquement, car la distribution de l'acteur n'est pas spécifiée.

La propriété `memberNum` est pratique pour permuter les acteurs affectés à une image-objet lorsqu'ils appartiennent tous à la même distribution. Pour permuter des acteurs appartenant à des distributions différentes, utilisez la propriété d'image-objet `member`. Pour que la valeur définie par Lingo dure au-delà de l'image-objet courante, celle-ci doit être un esclave.

Cette propriété est également pratique pour permuter les acteurs lorsque l'utilisateur clique sur une image-objet et simuler ainsi l'image inversée qui apparaît lorsque l'utilisateur clique sur un bouton standard. Vous pouvez également décider de ce qui se passe dans l'animation en fonction de l'acteur associé à une image-objet spécifique.

Si vous définissez cette propriété dans un script alors que la tête de lecture ne bouge pas, utilisez la commande `updateStage` pour redessiner la scène.

Cette propriété peut être testée et définie.

Exemples

L'instruction suivante remplace l'acteur affecté à l'image-objet 3 par l'acteur 35 de la même distribution :

```
sprite(3).memberNum = 35
```

L'instruction suivante affecte l'acteur Narrateur à l'image-objet 10 en donnant le numéro de l'acteur Narrateur à la propriété d'image-objet `memberNum`. L'acteur Narrateur est dans la même distribution que l'acteur actuel de l'image-objet.

```
sprite(10).memberNum = member("Narrateur").number
```

Le gestionnaire suivant permute les bitmaps lorsque l'utilisateur clique sur un bouton ou le survole. Le dessin du bouton enfoncé doit suivre immédiatement celui du bouton relâché dans la même distribution.

```
on mouseDown
  boutonHaut = sprite(the clickOn).memberNum
  boutonBas = boutonHaut + 1
  repeat while the stillDown
    if rollover(the clickOn) then
      sprite(the clickOn).memberNum = boutonBas
    else
      sprite(the clickOn).memberNum = boutonHaut
    end if
    updateStage
  end repeat
  if rollover (the clickOn) then put "Le bouton a été activé"
end
```

Voir aussi

`castLib`, `member (propriété d'image-objet)`, `number (propriété d'acteur)`, `member (mot-clé)`

members

Consultez

`number of members`

memorySize

Syntaxe

```
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 le 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`

menu

Syntaxe

```
menu: nomDeMenu
nomDélément | script
nomDélément | script
...
```

– ou –

```
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.

Sur le Macintosh, vous pouvez utiliser des caractères spéciaux pour définir des menus personnalisés. Ces caractères spéciaux différencient les majuscules des minuscules. 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 le Macintosh, crée le symbole Pomme et active les éléments de la barre des menus lorsque vous définissez un menu Pomme.
!Ã	!ÃSélection rapide	*Sur le Macintosh, place une coche (Option+v) près du menu.
<B	Gras<B	*Sur le Macintosh, donne à l'élément de menu le style Gras.
<I	Italique<I	*Sur le Macintosh, définit le style Italique.
<U	Souligné<U	*Sur le Macintosh, définit le style Souligné.
<O	Relief<O	*Sur le Macintosh, définit le style Relief.
<S	Ombré<S	*Sur le Macintosh, définit le style Ombré.
|	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 le 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
Ouvrir/O | go to frame "Ouvrir"
Fermer/W | go to frame "Fermer"
Convertir/C | convertirCeci
  (-
Quitter/Q | go to frame "Quitter"
```

Voir aussi

`installMenu`, `name (propriété de menu)`, `name (propriété d'élément de menu)`, `number (éléments de menu)`, `checkMark`, `enabled`, `script`

mesh (propriété)

Syntaxe

```
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).

Exemples

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("univers").model("Avion").meshDeform.mesh.count
-- 4
```

Voir aussi

`meshDeform (modificateur)`, `colorList`, `textureCoordinateList`, `textureLayer`, `normalList`, `vertexList (déformation de maille)`, `face`

meshDeform (modificateur)

Syntaxe

`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.

Exemples

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

Syntaxe

```
the milliseconds
```

Description

Propriété système ; renvoie l'heure courante en millisecondes (1/1000ème de seconde). Le compte commence à partir du démarrage de l'ordinateur.

Exemples

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 :

```
secondesCourantes = milliseconds/1000
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.

```
on idle
  if the milliseconds > 1000 * 60 * 60 * 4 then
    alert "Faites une pause"
  end if
end
```

Voir aussi

`ticks`, `time()`, `timer`

min

Syntaxe

```
liste.min
min(liste)
min (a1, a2, a3...)
```

Description

Fonction ; indique la valeur minimale de la liste spécifiée par *liste*.

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()

minSpeed

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).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("Feux").modelResource("systèmeThermique").emitter.\
  minSpeed = 5
```

Voir aussi

maxSpeed, emitter

missingFonts

Syntaxe

```
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`

mod

Syntaxe

```
expressionEntière1 mod expressionEntière2
```

Description

Opérateur mathématique ; calcule le reste de la division de deux entiers. Dans l'opération suivante, *`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 (positif ou négatif) de *`expressionEntière1`*.

Cet opérateur arithmétique a un niveau de priorité de 4.

Exemples

L'instruction suivante divise 7 par 4, puis affiche le reste dans la fenêtre Messages :

```
put 7 mod 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`.

```
on définirLencre
  repeat with monImageObjet = 1 to the lastChannel
    if (monImageObjet mod 2) = 1 then
      sprite(monImageObjet).ink = 0
    else
      sprite(monImageObjet).ink = 8
    end if
  end repeat
end définirLencre
```

Le gestionnaire suivant change régulièrement l'acteur d'une image-objet en choisissant un autre acteur parmi un certain nombre de bitmaps :

```
on exitFrame
  global gCompteur
  -- exemples de valeurs des numéros d'acteurs bitmaps
  lesBitmaps = [2,3,4,5,6,7]
  -- spécifier 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]
  go the frame
end
```

modal

Syntaxe

```
window "fenêtre".modal
the modal of window "fenêtre"
```

Description

Propriété de fenêtre ; spécifie si les animations peuvent répondre aux événements se produisant à l'extérieur de la fenêtre spécifiée par *fenêtre*.

- Lorsque la propriété de fenêtre `modal` a pour valeur `TRUE`, les animations ne peuvent pas répondre aux événements se produisant à l'extérieur de la fenêtre.
- Lorsque la propriété de fenêtre `modal` a pour valeur `FALSE`, les animations peuvent répondre aux événements se produisant à l'extérieur de la fenêtre.

Lorsque la propriété de fenêtre `modal` a pour valeur `TRUE`, vous pouvez faire d'une animation spécifique lue dans une fenêtre la seule animation avec laquelle une interaction utilisateur est permise.

Cette propriété fonctionne également dans l'environnement auteur. Si vous donnez à la propriété de fenêtre `modal` la valeur `TRUE`, vous ne pourrez pas non plus utiliser les autres fenêtres de Director de manière interactive.

Vous pouvez toujours fermer une fenêtre modale en utilisant le raccourci clavier Ctrl+Alt+point (Windows) ou Cmd+point (Macintosh).

Exemple

L'instruction suivante permet aux animations de répondre aux événements extérieurs à la fenêtre Palette des outils :

```
window("Palette des outils").modal = FALSE
```

mode (émetteur)

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).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)

Syntaxe

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

Syntaxe

```
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`.

Exemples

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

Syntaxe

```
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

Syntaxe

```
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

Syntaxe

```
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.

Exemples

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

Syntaxe

```
member(quelActeur).camera(quelleCaméra).modelsUnderLoc\
  (pointDansLimageObjet {, nombreMaximumDeModèles, précision})
```

Description

Commande 3D ; renvoie une liste des modèles situés sous le point spécifié par *pointDansLimageObjet* à l'intérieur du rect d'une image-objet utilisant la caméra référencée. L'emplacement de *pointDansLimageObjet* est calculé en fonction du coin supérieur gauche de l'image-objet, en pixels.

Le paramètre facultatif *nombreMaximumDeModèles* vous permet de limiter la longueur 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.

Le paramètre facultatif *précision* vous permet de spécifier la précision des informations renvoyées. Le paramètre *précision* peut avoir les valeurs suivantes :

`#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.

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.

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.

```
on mouseUp
  pt = the mouseLoc - point(sprite(5).left, sprite(5).top)
  m = sprite(5).camera.modelsUnderLoc(pt, 3, #detailed)
  put m
end
```

Voir aussi

`modelsUnderRay`, `modelUnderLoc`

modelsUnderRay

Syntaxe

```
member(quelActeur).modelsUnderRay(vecteurDemplacement, vecteurDeDirection {,
  nombreMaxDeModèles, précision})
```

Description

Commande 3D ; renvoie une liste des modèles situés sous un rayon tracé à partir de la position spécifiée par *`vecteurDemplacement`*, en direction de *`vecteurDeDirection`* ; les deux vecteurs étant spécifiés en coordonnées relatives à l'univers.

Le paramètre facultatif *`nombreMaximumDeModèles`* vous permet de limiter la longueur 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.

Le paramètre facultatif *`précision`* vous permet de spécifier la précision des informations renvoyées. Le paramètre *`précision`* peut avoir les valeurs suivantes :

`#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.

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.

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

Syntaxe

```
member(quelActeur).camera(quelleCaméra).\
  modelUnderLoc(pointDansLimageObjet)
```

Description

Commande 3D ; renvoie une référence au premier modèle situé sous le point spécifié par *`pointDansLimageObjet`* à l'intérieur du rect d'une image-objet utilisant la caméra référencée. L'emplacement de *`pointDansLimageObjet`* est calculé en fonction du coin supérieur gauche de l'image-objet, en pixels.

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`.

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.

```
on mouseUp
  pt = the mouseLoc - point(sprite(5).left, sprite(5).top)
  m = sprite(5).camera.modelUnderLoc(pt)
  put m
end
```

Voir aussi

`modelsUnderLoc`, `modelsUnderRay`

modified

Syntaxe

```
member(quelActeur).modified
the modified of member quelActeur
```

Description

Propriété d'acteur ; indique si l'acteur spécifié par *quelActeur* a été modifié depuis sa lecture depuis le fichier de l'animation.

- Lorsque la propriété d'acteur `modified` a pour valeur `TRUE`, l'acteur a été modifié depuis sa lecture depuis le fichier de l'animation.
- Lorsque la propriété d'acteur `modified` a pour valeur `FALSE`, l'acteur n'a pas été modifié depuis sa lecture depuis le fichier de l'animation.

La fonction d'acteur `modified` renvoie toujours `FALSE` dans le cas des acteurs d'une animation lue sous forme d'applet.

Exemple

L'instruction suivante vérifie si l'acteur Introduction a été modifié depuis sa lecture à partir du fichier de l'animation :

```
return member("Introduction").modified
```

modifiedBy

Syntaxe

```
quelActeur.modifiedBy
the modifiedBy of quelActeur
```

Description

Propriété d'acteur ; enregistre le nom de l'utilisateur qui a effectué la dernière modification de l'acteur. Cette information 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 peut être testée, mais pas définie. Elle 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 :

```
put member(1).modifiedBy
  -- "Jean Laforêt"
```

Voir aussi

`comments`, `creationDate`, `modifiedDate`

modifiedDate

Syntaxe

```
Acteur.modifiedDate
the modifiedDate of Acteur
```

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. Cette propriété s'avère utile pour assurer le suivi et la coordination des projets Director.

Cette propriété peut être testée, mais pas définie. 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 :

```
put member(1).modifiedDate
  -- date( 2002, 12, 8 )
```

Voir aussi

`comments`, `creationDate`, `modifiedBy`

modifier

Syntaxe

```
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.

Veuillez noter que, 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 retirer.

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[]

Syntaxe

```
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

Syntaxe

```
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

Syntaxe

```
sprite(quelleImageObjet).mostRecentCuePoint
the mostRecentCuePoint of sprite quelleImageObjet
sound(numéroDePiste).mostRecentCuePoint
the mostRecentCuePoint of sound numéroDePiste
```

Description

Propriété d'acteur, de piste audio et d'image-objet ; pour les acteurs son, 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.

Pour les pistes audio, la valeur renvoyée concerne l'acteur son en cours de lecture sur la 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.

Exemples

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 :

```
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 :

```
put sound(2).mostRecentCuePoint
```

Voir aussi

`cuePointNames`, `isPastCuePoint()`, `cuePointTimes`, `on cuePassed`

motion

Syntaxe

```
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`.

Exemples

```
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

Syntaxe

```
sprite(quelleImageObjetQTVR).motionQuality
motionQuality of sprite quelleImageObjetQTVR
```

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`.

```
sprite(1).motionQuality = #minQuality
```

mouseChar

Syntaxe

```
the mouseChar
```

Description

Propriété système ; s'utilise avec les images-objets champ. Elle contient le numéro du caractère se trouvant sous le curseur lorsqu'elle est appelée. 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 la propriété `mouseChar` peut changer dans un gestionnaire ou une boucle de répétition. 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.

Exemples

L'instruction suivante détermine si le curseur est placé sur une image-objet champ et, si ce n'est pas le cas, donne à l'acteur champ Instructions le contenu Veuillez pointer sur un caractère:

```
if the mouseChar = -1 then
  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 :

```
caractèreActuel = member(the mouseMember).char[the mouseChar]
```

Le gestionnaire suivant met en surbrillance le caractère qui se trouve sous le curseur lorsque l'image-objet contient un acteur texte :

```
property spriteNum

on mouseWithin me
  if sprite(spriteNum).member.type = #field then
    MC = the mousechar
    if MC < 1 then exit    -- si sur une bordure, ligne de fin, etc.
    hilite char MC of field sprite(spriteNum).member
    else alert "hilite et mouseChar : uniquement pour les champs."
end
```

Voir aussi

`mouseItem`, `mouseLine`, `mouseWord`, `char...of`, `number (caractères)`

on mouseDown (gestionnaire d'événement)

Syntaxe

```
on mouseDown
  instruction(s)
end
```

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 Lingo 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.

Lorsque utilisé dans un comportement, cet événement reçoit en paramètre la référence au script de l'image-objet ou au script d'image `me`.

Exemples

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 :

```
on mouseDown
  if the clickOn = 0 then go to frame "additionnerLaSomme"
end
```

Le gestionnaire suivant, affecté à un script d'image-objet, lit un son lorsque l'utilisateur clique sur l'image-objet :

```
on mouseDown
  puppetSound "Grillons"
end
```

Voir aussi

clickOn, mouseDownScript, mouseUpScript

the mouseDown (propriété système)

Syntaxe

```
the mouseDown
```

Description

Propriété système ; indique si l'utilisateur est en train d'appuyer sur le bouton de la souris (TRUE) ou non (FALSE).

Le lecteur Director pour Java ne met pas à jour la propriété mouseDown lorsque Lingo exécute une boucle de répétition. Si vous tentez de tester mouseDown dans une boucle de répétition d'une applet, celle-ci s'arrête.

Exemple

Le gestionnaire suivant émet un bip sonore jusqu'à ce que l'utilisateur clique sur la souris :

```
on enterFrame
  repeat while the mouseDown = FALSE
    beep
  end repeat
end
```

L'instruction suivante fait sortir Lingo de la boucle de répétition ou du gestionnaire dans lequel il se trouve lorsque l'utilisateur appuie sur le bouton de la souris :

```
if the mouseDown then exit
```

Voir aussi

mouseH, the mouseUp (propriété système), mouseV, on mouseDown (gestionnaire d'événement), on mouseUp (gestionnaire d'événement)

mouseDownScript

Syntaxe

```
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 = "suivant"
```

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)`

on mouseEnter

Syntaxe

```
on mouseEnter
   instruction(s)
end
```

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.

Lorsque 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 :

```
property spriteNum
on mouseEnter me
  -- détermine l'acteur courant, passe à l'acteur suivant de la distr.
  acteurCourant = sprite(spritenum).member.number
  sprite(spritenum).member = acteurCourant +1
end
on mouseLeave me
  -- détermine l'acteur courant, passe à l'acteur précédent de la distr.
  acteurCourant = sprite(spritenum).member.number
  sprite(spritenum).member = acteurCourant - 1
end
```

Voir aussi

`on mouseLeave`, `on mouseWithin`

mouseH

Syntaxe

```
the mouseH
mouseH()
```

Description

Propriété et fonction système ; indique la position horizontale du pointeur de la souris. La valeur de `mouseH` correspond au nombre de pixels entre le curseur et le bord gauche de la scène.

La fonction `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 combinée des fonctions `mouseH` et `mouseV` vous permet de déterminer la position exacte du curseur.

Le lecteur Director pour Java ne met pas à jour la fonction `mouseH` lorsque Lingo exécute une boucle de répétition.

Cette fonction peut être testée, mais pas définie.

Exemples

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 :

```
on mouseDown
  sprite(10).locH = the mouseH
  sprite(10).locV = the mouseV
  updateStage
end
```

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` :

```
if abs(mouseH() - startH) > 10 then
  glissementSuffisant = TRUE
```

Voir aussi

`locH`, `locV`, `mouseV`, `mouseLoc`

mouseItem

Syntaxe

```
the mouseItem
```

Description

Propriété système ; 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. 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 la propriété une seule fois et affectez sa valeur à une variable locale.

Exemples

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 :

```
if the mouseItem = -1 then
  member("Instructions") = "Veuillez pointer sur un élément"
end if
```

L'instruction suivante affecte à la variable *`élémentCourant`* l'élément situé en-dessous du curseur dans le champ spécifié :

```
élémentCourant = member(mouseMember).item(mouseItem)
```

Le gestionnaire suivant met en surbrillance l'élément placé sous le pointeur à chaque fois que l'utilisateur appuie sur le bouton de la souris :

```
on mouseDown
  ceChamp = sprite(the clickOn).member
  if the mouseItem < 1 then exit
  dernierElément = 0
  repeat while the stillDown
    MI = the mouseItem
    if MI < 1 then next repeat
    if MI <> dernierElément then
      ceChamp.item[MI].hilite()
      dernierElément = MI
    end if
  end repeat
end
```

Voir aussi

`item...of`, `mouseChar`, `mouseLine`, `mouseWord`, `number (éléments)`, `itemDelimiter`

on mouseLeave

Syntaxe

```
on mouseLeave
  instruction(s)
end
```

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.

Lorsque 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 :

```
property spriteNum
on mouseEnter me
  -- détermine l'acteur courant, passe à l'acteur suivant de la distr.
  acteurCourant = sprite(spritenum).member.number
  sprite(spritenum).member = acteurCourant +1
end
on mouseLeave me
  -- détermine l'acteur courant, passe à l'acteur précédent de la distr.
  acteurCourant = sprite(spritenum).member.number
  sprite(spritenum).member = acteurCourant - 1
end
```

Voir aussi

`on mouseEnter`, `on mouseWithin`

mouseLevel

Syntaxe

```
sprite(quelleImageObjetQuickTime).mouseLevel
the mouseLevel of sprite quelleImageObjetQuickTime
```

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 Lingo.

- `#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 Lingo. 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`.

```
on prepareFrame
  if sprite(5).member.name contains "QTVR" then
    sprite(5).mouseLevel = #all
  else
    sprite(5).mouseLevel = #none
  end if
end
```

mouseLine

Syntaxe

```
the mouseLine
```

Description

Propriété système ; lorsque appelée et que le curseur se trouve au-dessus d'une image-objet champ, renvoie le numéro de la ligne sur laquelle se trouve le pointeur. Le compte commence au début du champ. Une ligne est définie par un retour de 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 de répétition. 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.

Exemples

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 :

```
if the mouseLine = -1 then
  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 :

```
ligneCourante = member(the mouseMember).line[the mouseLine]
```

Le gestionnaire suivant met en surbrillance la ligne placée sous le pointeur à chaque fois que l'utilisateur appuie sur le bouton de la souris :

```
on mouseDown
  ceChamp = sprite(the clickOn).member
  if the mouseLine < 1 then exit
  dernièreLigne = 0
  repeat while the stillDown
    ML = the mouseLine
    if ML < 1 then next repeat
    if ML <> dernièreLigne then
      ceChamp.line[ML].hilite()
      dernièreLigne = ML
    end if
  end repeat
end
```

Voir aussi

`mouseChar`, `mouseItem`, `mouseWord`, `line...of`, `number (lignes)`

mouseLoc

Syntaxe

```
the mouseLoc
```

Description

Fonction ; renvoie la position actuelle de la souris sous forme d'un point().

L'emplacement du point est indiqué sous forme de deux coordonnées, d'abord l'emplacement horizontal, puis l'emplacement vertical.

Voir aussi

`mouseH`, `mouseV`

mouseMember

Syntaxe

```
the mouseMember
```

Description

Propriété système ; lorsque appelée, renvoie l'acteur associé à l'image-objet qui se trouve sous le pointeur. Si le pointeur ne se trouve pas au-dessus d'une image-objet, elle renvoie le résultat `VOID`.

La propriété `mouseMember` remplace la propriété `mouseCast` des versions précédentes de Director.

Vous pouvez utiliser cette fonction 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.

Pour les distributions autres que la distribution 1, `mouseMember` renvoie une valeur ne distinguant pas le numéro de l'acteur de celui de la distribution. Pour faire la différence entre le numéro de l'acteur et celui de la distribution, utilisez l'expression `member (the mouseMember)`. Toutefois, si l'utilisateur ne clique pas sur une image-objet, cette expression produit une erreur de script.

Exemples

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.

```
if the mouseMember = member "Hors limites" then alert "Gardez vos distances !"
```

L'instruction suivante affecte l'acteur de l'image-objet qui se trouve sous le curseur à la variable *dernierActeur* :

```
dernierActeur = the mouseMember
```

Voir aussi

`member (propriété d'image-objet)`, `memberNum`, `clickLoc`, `mouseChar`, `mouseItem`, `mouseLine`, `mouseWord`, `rollOver()`, `number (propriété d'acteur)`

mouseOverButton

Syntaxe

```
sprite quelleImageObjetFlash.mouseOverButton
the mouseOverButton of sprite quelleImageObjetFlash
```

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.

```
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
  updatestage
end
```

on mouseUp (gestionnaire d'événement)

Syntaxe

```
on mouseUp
  instruction(s)
end
```

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.

Lorsque 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 :

```
on mouseUp
  sprite(10).member = member "Gris"
end
```

Voir aussi

`on mouseDown (gestionnaire d'événement)`

the mouseUp (propriété système)

Syntaxe

```
the mouseUp
```

Description

Propriété système ; indique si le bouton de la souris est relâché (`TRUE`) ou enfoncé (`FALSE`).

Le lecteur Director pour Java ne met pas à jour la propriété `mouseUp` lorsque Lingo exécute une boucle de répétition.

Exemples

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.

```
on exitFrame me
  if the mouseUp then
    go the frame
  end if
end
```

L'instruction suivante fait sortir Lingo de la boucle de répétition ou du gestionnaire qu'il est en train d'exécuter lorsque l'utilisateur relâche le bouton de la souris :

```
if the mouseUp then exit
```

Voir aussi

`the mouseDown (propriété système)`, `mouseH`, `mouseV`, `the mouseUp (propriété système)`

on mouseUpOutside

Syntaxe

```
on mouseUpOutside me
  instruction(s)
end
```

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 :

```
on mouseUpOutside me
  puppetSound "Professeur Machin"
end
```

Voir aussi

`on mouseEnter`, `on mouseLeave`, `on mouseWithin`

mouseUpScript

Syntaxe

```
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`.

Exemples

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

Syntaxe

```
the mouseV
mouseV()
```

Description

Propriété système ; indique la position verticale du curseur de la souris, en pixels, calculée à partir du haut de la scène. Cette valeur 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.

Le lecteur Director pour Java ne met pas à jour la propriété `mouseV` lorsque Lingo exécute une boucle de répétition.

Cette propriété peut être testée, mais pas définie.

Exemples

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 :

```
on mouseDown
  sprite(1).locH = the mouseH
  sprite(1).locV = the mouseV
  updateStage
end
```

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 :

```
if abs(the mouseV - startV) > 10 then glissementSuffisant = TRUE
```

Voir aussi

mouseH, locH, locV, mouseLoc

on mouseWithin

Syntaxe

```
on mouseWithin
  instruction(s)
end
```

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.

Lorsque 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 :

```
on mouseWithin
  member("Affichage").text = string(the mouseH)
end mouseWithin
```

Voir aussi

on mouseEnter, on mouseLeave

mouseWord

Syntaxe

```
the mouseWord
```

Description

Propriété système ; 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. 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 de répétition. Lorsqu'un gestionnaire ou une boucle utilisent cette propriété plusieurs fois, il est d'usage d'appeler cette dernière une seule fois et d'affecter sa valeur à une variable locale.

Exemples

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 :

```
if the mouseWord = -1 then
  member("Instructions").text = "Veuillez pointer sur un mot"
else
  member("Instructions").text = "Merci."
end if
```

L'instruction suivante affecte à la variable *motCourant* le numéro du mot qui se trouve sous le pointeur dans le champ spécifié.

```
motCourant = member(the mouseMember).word[the mouseWord]
```

Le gestionnaire suivant met en surbrillance le mot placé sous le pointeur à chaque fois que l'utilisateur appuie sur le bouton de la souris :

```
on mouseDown
  ceChamp = sprite(the clickOn).member
  if the mouseWord < 1 then exit
  dernierMot = 0
  repeat while the stillDown
    MW = the mouseWord
    if MW < 1 then next repeat
    if MW <> dernierMot then
      ceChamp.word[MW].hilite()
        dernierMot = MW
    end if
  end repeat
end
```

Voir aussi

mouseChar, mouseItem, mouseLine, number (mots), word...of

moveableSprite

Syntaxe

```
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.

Exemples

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`

move member

Syntaxe

```
member(quelActeur).move()
member(quelActeur).move(member quelEmplacement)
move member quelActeur {,member quelEmplacement}
```

Description

Commande ; place l'acteur spécifié par *`quelActeur`* dans la première position vide de la fenêtre Distribution (si aucun paramètre n'est défini) ou dans la position spécifiée par *`quelEmplacement`*.

Pour obtenir les meilleurs résultats, il est préférable d'utiliser cette commande pendant la programmation et non pendant l'exécution d'une animation, car le déplacement est en général enregistré avec le fichier de l'animation. 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.

Exemples

L'instruction suivante déplace l'acteur Temple vers la première position vide de la fenêtre Distribution :

```
member("Temple").move()
```

L'instruction suivante déplace l'acteur Temple vers l'emplacement 20 de la fenêtre Distribution Bitmaps :

```
member("Temple").move(20, "Bitmaps")
```

moveToBack

Syntaxe

```
window("quelleFenêtre ").MoveToBack()
moveToBack window quelleFenêtre
```

Description

Commande ; place la fenêtre spécifiée par *`quelleFenêtre`* derrière toutes les autres fenêtres.

Exemples

Les instructions suivantes placent la première fenêtre de la liste `windowList` derrière toutes les autres fenêtres :

```
maFenêtre = the windowList[1]
moveToBack maFenêtre
```

Si vous connaissez le nom de la fenêtre à déplacer, utilisez la syntaxe suivante :

```
window("Fenêtre Démo").moveToBack()
```

moveToFront

Syntaxe

```
window("quelleFenêtre ").MoveToFront()
moveToFront window quelleFenêtre
```

Description

Commande ; place la fenêtre spécifiée par *quelleFenêtre* devant toutes les autres fenêtres.

Exemples

Les instructions suivantes placent la première fenêtre de la liste `windowList` devant toutes les autres fenêtres :

```
maFenêtre = the windowList[1]
moveToFront maFenêtre
```

Si vous connaissez le nom de la fenêtre à placer au premier plan, utilisez la syntaxe suivante :

```
window("Fenêtre Démo").moveToFront()
```

moveVertex()

Syntaxe

```
member(RéfDacteur). MoveVertex(indiceDeSommet, changementX, changementY)
moveVertex(member RéfDacteur, indiceDeSommet, 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.

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 :

```
member("Archie").moveVertex(1, 25, 10)
```

Voir aussi

`addVertex`, `deleteVertex()`, `moveVertexHandle()`, `originMode`, `vertexList`

moveVertexHandle()

Syntaxe

```
moveVertexHandle(member réfDacteur, IndiceDeSommet, IndiceDePoignée,
  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.

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 :

```
MoveVertexHandle(member "Archie", 2, 1, 15, -5)
```

Voir aussi

`addVertex`, `deleteVertex()`, `originMode`, `vertexList`

on moveWindow

Syntaxe

```
on moveWindow
  instruction(s)
end
```

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 :

```
on moveWindow
  put "Déplacement de la fenêtre contenant"&&the movieName
end
```

Voir aussi

`activeWindow`, `movieName`, `windowList`

movie

Cette propriété est obsolète. Utilisez `movieName`.

movieAboutInfo

Syntaxe

```
the movieAboutInfo
```

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.

Cette propriété peut être définie, mais pas testée.

Voir aussi

`movieCopyrightInfo`

movieCopyrightInfo

Syntaxe

```
the movieCopyrightInfo
```

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.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante affiche les informations sur le copyright dans un acteur texte :

```
member("Affichage").text = "Copyright"&&the movieCopyrightInfo
```

Voir aussi

`movieAboutInfo`

movieFileFreeSize

Syntaxe

```
the movieFileFreeSize
```

Description

Propriété d'animation ; renvoie le nombre d'octets inutilisés dans l'animation courante par suite de modifications apportées aux acteurs et distributions de cette animation. 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, la fonction `movieFileFreeSize` renvoie 0.

Exemple

L'instruction suivante affiche le nombre d'octets inutilisés dans le fichier de l'animation courante :

```
put the movieFileFreeSize
```

movieFileSize

Syntaxe

```
the movieFileSize
```

Description

Propriété d'animation ; renvoie le nombre d'octets du fichier de l'animation courante enregistré sur le disque dur. 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 :

```
put the movieFileSize
```

movieFileVersion

Syntaxe

```
the movieFileVersion
```

Description

Propriété d'animation ; indique la version de Director dans laquelle l'animation a été le plus récemment enregistrée. Le résultat est une chaîne.

Exemple

L'instruction suivante affiche la version de Director dans laquelle l'animation a été enregistrée pour la dernière fois :

```
put the movieFileVersion
-- "800"
```

movieImageCompression

Syntaxe

```
the movieImageCompression
```

Description

Propriété 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. 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. Sa valeur peut être constituée de l'un de ces symboles :

Valeur	Signification
#standard	Utilise le format de compression interne standard de Director.
#jpeg	Utilise la compression JPEG (voir `imageCompression`).

Vous devez normalement définir cette propriété dans la boîte de dialogue Paramètres de publication de Director.

Vous pouvez supplanter ce paramètre pour certains acteurs bitmap en définissant leurs propriétés d'acteur `imageCompression` et `imageQuality`.

Voir aussi

`imageCompression`, `imageQuality`, `movieImageQuality`

movieImageQuality

Syntaxe

```
the movieImageQuality
```

Description

Propriété d'animation ; indique le niveau de compression à utiliser lorsque la propriété `movieImageCompression` est définie sur `#jpeg`. 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.

Les différents acteurs peuvent supplanter cette propriété d'animation en utilisant la propriété d'acteur `imageCompression`.

Voir aussi

`imageCompression`, `imageQuality`, `movieImageCompression`

movieName

Syntaxe

```
the movieName
```

Description

Propriété d'animation ; indique le nom simple de l'animation courante.

Dans l'environnement auteur de Director, cette propriété affecte une chaîne vide à toute nouvelle animation non encore enregistrée.

Exemple

L'instruction suivante affiche le nom de l'animation courante dans la fenêtre Messages :

```
put the movieName
```

Voir aussi

`moviePath`

moviePath

Syntaxe

```
the moviePath
```

Description

Propriété d'animation ; indique le chemin d'accès du dossier de l'animation courante.

Pour les noms de fichiers fonctionnant sur Windows et Macintosh, utilisez l'opérateur de chemin d'accès *@*.

Vous pourrez voir un exemple de `moviePath` dans une animation en consultant l'animation Read and Write Text du dossier Learning/Lingo_Examples, lui-même dans le dossier de Director.

Exemples

L'instruction suivante affiche le chemin d'accès du dossier de l'animation courante :

```
put the moviePath
```

L'instruction suivante lit le fichier son Violon.aif stocké dans le sous-dossier Sons du dossier de l'animation courante. Notez le délimiteur de chemin, qui indique l'environnement Windows :

```
sound playFile 1, the moviePath&"Sons/Violon.aif"
```

Voir aussi

`@ (chemin d'accès)`, `movieName`

movieRate

Syntaxe

```
sprite(quelleImageObjet).movieRate
the movieRate of sprite quelleImageObjet
```

Description

Propriété d'image-objet vidéo numérique ; contrôle la vitesse de lecture de la vidéo numérique dans une piste spécifique. `movieRate` 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 `movieRate` 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.

Vous pourrez voir un exemple de `movieRate` dans une animation en consultant l'animation QT et Flash du dossier Learning/Lingo_Examples, lui-même dans le dossier de Director.

Exemples

L'instruction suivante définit une vitesse de lecture normale de la vidéo numérique contenue dans l'image-objet vidéo numérique de la piste 9 :

```
sprite(9).movieRate = 1
```

L'instruction suivante provoque la lecture en sens inverse de la vidéo numérique de l'image-objet de la piste 9 :

```
sprite(9).movieRate = -1
```

Voir aussi

`duration`, `movieTime`

movieTime

Syntaxe

```
sprite(quelleImageObjet).movieTime
the movieTime of sprite quelleImageObjet
```

Description

Propriété d'image-objet vidéo numérique ; détermine la position temporelle courante de l'animation vidéo numérique exécutée dans la piste spécifiée par *`quelleImageObjet`*. La valeur de `movieTime` est mesurée en battements.

Cette propriété peut être testée et définie.

Vous pourrez voir un exemple de movieTime dans une animation en consultant l'animation QT et Flash du dossier Learning/Lingo_Examples, lui-même dans le dossier de Director.

Exemples

L'instruction suivante affiche dans la fenêtre Messages la durée actuelle de l'animation QuickTime de la piste 9 :

```
put sprite(9).movieTime
```

L'instruction suivante affecte à la variable *Poster* la valeur de position temporelle actuelle de l'animation QuickTime de la piste 9 :

```
sprite(9).movieTime = Poster
```

Voir aussi

duration

movieXtraList

Syntaxe

```
the movieXtraList
```

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.

- #name – Spécifie le nom de fichier de l'Xtra sur la plate-forme courante. Il est possible que cette liste ne contienne aucune entrée #name, si l'Xtra n'existe que sur une seule plate-forme.
- #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.

Deux propriétés différentes peuvent figurer dans la propriété movieXtraList :

Exemple

L'instruction suivante affiche les données de movieXtraList dans la fenêtre Messages :

```
put the movieXtraList
-- [[#name: "Mix Services"], [#name: "Sound Import Export"], [#name: "SWA
   Streaming  \ PPC Xtra"], [#name: "TextXtra PPC"], [#name: "Font Xtra
   PPC"],[#name: "Flash Asset  \ Options PPC"], [#name: "Font Asset PPC"],
   [#name: "Flash Asset PPC",  \
   #packagefiles: [[#fileName: "Flash Asset PPC", #version: "1.0.3"]]]]
```

Voir aussi

xtraList

multiply()

Syntaxe

transformation.multiply(*transformation2*)

Description

Commande 3D ; applique les effets de position, de rotation et de redimensionnement de *transformation2* après la transformation d'origine.

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("Séquence").model("Pluton").transform.multiply(member("Séquence")\
   .model("Mars").transform)
```

multiSound

Syntaxe

```
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 dans la piste audio 2 si l'ordinateur supporte plus d'une piste audio :

```
if the multiSound then sound playFile 2, "Musique.wav"
```

name

Syntaxe

```
member(quelActeur).texture(quelleTexture).name
member(quelActeur).shader(quelMatériau).name
member(quelActeur).motion(quelMouvement).name
member(quelActeur).modelResource(quelleRessDeMod).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 distribution)

Syntaxe

```
castLib (quelleDistribution).name
the name of castLib quelleDistribution
```

Description

Propriété de distribution ; renvoie le nom de la distribution spécifiée.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante passe en revue toutes les distributions de l'animation et affiche leurs noms dans la fenêtre Messages :

```
totalDesDistributions = the number of castLibs
  repeat with distrCourante = 1 to totalDesDistributions
    put "La
  distribution"&&distrCourante&&"s'appelle"&&castLib(distrCourante).name
  end repeat
```

Voir aussi

&& (opérateur de concaténation)

name (propriété d'acteur)

Syntaxe

```
member(quelActeur).name
the name of member quelActeur
```

Description

Propriété d'acteur ; détermine le nom de l'acteur spécifié. L'argument *quelActeur* est une chaîne lorsqu'il sert à désigner le nom d'un acteur ou un nombre entier lorsqu'il sert à désigner le numéro d'un acteur.

Ce nom est une chaîne descriptive créée par le développeur. L'utilisation de cette propriété équivaut à saisir le nom de l'acteur dans la boîte de dialogue Propriétés de l'acteur.

Cette propriété peut être testée et définie.

Exemples

L'instruction suivante donne à l'acteur Activé le nom Désactivé :

```
member("Activé").name = "Désactivé"
```

L'instruction suivante donne à l'acteur 15 le nom Fond Sonore :

```
member(15).name = "Fond Sonore"
```

L'instruction suivante donne à la variable *sonNom* le nom de l'acteur qui suit l'acteur dont le numéro est identique à la variable i:

```
sonNom = member(i + 1).name
```

Voir aussi

number (propriété d'acteur)

name (propriété de menu)

Syntaxe

```
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.

Exemples

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.

Exemples

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 (propriété de fenêtre)

Syntaxe

```
window (quelleFenêtre).name
the name of window quelleFenêtre
```

Description

Propriété de fenêtre ; détermine le nom de la fenêtre spécifiée dans `windowList`. La propriété `title` détermine le titre qui apparaît dans la barre de titre de la fenêtre.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante donne à la fenêtre Hier le nom Aujourd'hui :

```
window("Hier").name = "Aujourd'hui"
```

name (propriété système)

Syntaxe

```
xtra (quelXtra).name
the name of xtra quelXtra
```

Description

Propriété système ; indique le nom de l'Xtra Lingo spécifié. Les Xtras offrant des services d'assistance ou d'autres fonctions qui ne sont pas disponibles pour Lingo ne supportent pas cette propriété.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante affiche le nom du premier Xtra dans la fenêtre Messages :

```
put xtra(1).name
```

name (propriété de temporisation)

Syntaxe

```
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`, `new()`, `period`, `persistent`, `target`, `time (propriété d'objet de temporisation)`, `timeout()`, `timeoutHandler`, `timeoutList`

name (propriété XML)

Syntaxe

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>élément 2</e2>
    <e3>élément 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

NAN

Description

Valeur de renvoi Lingo ; indique qu'une 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 :

```
put (-1).sqrt
-- NAN
```

Voir aussi

INF

near (brouillard)

Syntaxe

member(*quelActeur*).camera(*quelleCaméra*).fog.near
référenceDeCaméra.fog.near
member(*quelActeur*).camera(*quelleCaméra*).fog.far
référenceDeCaméra.fog.far

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

Syntaxe

```
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
```

neighbor

Syntaxe

```
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.

Voir aussi

`meshDeform (modificateur)`

netAbort

Syntaxe

```
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.

Exemple

L'instruction suivante passe un identifiant de réseau à `netAbort` pour annuler une opération réseau particulière :

```
on mouseUp
  netAbort(monIDRéseau)
end
```

Voir aussi

`getNetText()`, `postNetText`

netDone()

Syntaxe

```
netDone()
netDone(IDRéseau)
```

Description

Fonction ; indique si une opération de chargement en tâche de fond (telle que `getNetText`, `preloadNetThing`, `gotoNetMovie`, `gotoNetPage` ou `putNetText`) 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.

Exemples

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.

```
on exitFrame
  if netDone() = 1 then
    member("Affichage du texte").text = netTextResult()
  end if
end
```

Le gestionnaire suivant utilise un identifiant de réseau spécifique comme argument pour que `netDone` vérifie l'état d'une opération réseau particulière :

```
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
```

Voir aussi

`getNetText()`, `netTextResult()`, `gotoNetMovie`, `preloadNetThing()`

netError()

Syntaxe

```
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.

Lorsque l'animation est lue sous forme d'applet, cette fonction renvoie toujours une chaîne. Cette chaîne est une chaîne vide ou contient le texte décrivant l'erreur. Le contenu de la chaîne provient de Java et peut varier selon le système d'exploitation ou le navigateur utilisé. Il ne peut pas être traduit dans la langue locale.

Exemple

L'instruction suivante passe un identifiant de réseau à `netError` pour vérifier l'état d'erreur d'une opération réseau spécifique :

```
on exitFrame
  global monIDRéseau
  if netError(monIDRéseau)<>"OK" then beep
end
```

netLastModDate()

Syntaxe

```
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`.

Lorsque l'animation est lue sous forme d'applet, le format de date de cette fonction peut être différent de celui que Shockwave utilise ; la date est toujours dans le format que la fonction Java `Date.asString` renvoie. Ce format peut également varier sur les systèmes utilisant des langues différentes.

Exemple

Les instructions suivantes vérifient la date d'un fichier téléchargé depuis Internet :

```
if netDone() then
  laDate = netLastModDate()
  if laDate.char[6..11] <> "30 Jan" then
    alert "Ce fichier est périmé."
  end if
end if
```

Voir aussi

`netDone()`, `netError()`

netMIME()

Syntaxe

`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.

Exemple

Le gestionnaire suivant vérifie le type MIME d'un fichier téléchargé depuis Internet et répond en conséquence :

```
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
```

Voir aussi

`netDone()`, `netError()`, `getNetText()`, `postNetText`, `preloadNetThing()`

netPresent

Syntaxe

```
netPresent()
the netPresent
```

Description

Propriété système ; détermine si les Xtras requis pour accéder à Internet sont disponibles, mais n'indique pas si une connexion à Internet est actuellement active.

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 :

```
if not (the netPresent) then
  alert "Désolé, les Xtras réseau sont introuvables."
end if
```

netStatus

Syntaxe

```
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.

Exemple

L'instruction suivante place la chaîne Test dans la zone d'état du navigateur web dans lequel l'animation est lue :

```
on exitFrame
  netStatus "Test"
end
```

netTextResult()

Syntaxe

```
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. Lorsque l'animation est lue sous forme d'animation Shockwave, cette fonction ne renvoie que les résultats valides de l'opération `getNetText()` la plus récente.

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.

```
global gIdRéseau

on exitFrame
  if (netDone(gIdRéseau) = TRUE) and (netError(gIdRéseau) = "OK") then
    member("Affichage du texte").text = netTextResult()
  end if
end
```

Voir aussi

`netDone()`, `netError()`, `postNetText`

netThrottleTicks

Syntaxe

```
the netThrottleTicks
```

Description

Propriété système ; dans un environnement auteur Macintosh, permet de contrôler la fréquence de service d'une opération réseau.

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 le Macintosh. Elle n'a aucun effet sous Windows ou dans Shockwave sur le Macintosh.

new()

Syntaxe

```
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.

Le lecteur Director pour Java ne supporte cette fonction que pour la création d'objets enfants. Lorsqu'une animation est lue sous forme d'applet, vous ne pouvez pas utiliser la fonction `new` pour créer des acteurs.

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 instantanément 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, et renvoyer 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.

Vous pourrez voir un exemple de new() dans une animation en consultant les animations Parent Scripts et Read and Write Text du dossier Learning/Lingo_Examples, lui-même dans le dossier de Director.

Exemples

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
```

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 appellera le 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 (propriété d'acteur), timeout()

newCamera

Syntaxe

```
member(quelActeur).newCamera(nomDeNouvelleCaméra)
```

Description

Commande 3D ; crée une caméra, *nomDeNouvelleCaméra*, dans l'acteur. Le nom spécifié pour *nomDeNouvelleCaméra* doit être unique dans l'acteur.

Exemple

L'instruction suivante crée une nouvelle caméra appelée Caméra_embarquée.

```
member("Univers 3D").newCamera("Caméra_embarquée")
```

newCurve()

Syntaxe

```
acteurVecteur.newCurve(positionDansLaListeDesSommets)
newCurve(acteurVecteur, positionDansLaListeDesSommets)
```

Description

Fonction ; ajoute un symbole #newCurve à la liste vertexList de *acteurVecteur*, qui ajoute une nouvelle forme à la forme vectorielle. Le symbole #newCurve est ajouté à *positionDansLaListeDesSommets*. Vous pouvez fractionner une forme existante en appelant newCurve() positionné au milieu d'une série de sommets.

Exemple

L'instruction Lingo suivante ajoute 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.

```
member(2).newCurve(3)
member(2).curve[2]=member(2).curve[3]
curve, vertexList
```

newGroup

Syntaxe

```
member(quelActeur).newGroup(nomDeNouveauGroupe)
```

Description

Commande 3D ; crée un groupe, *nomDeNouveauGroupe*, et l'ajoute à la palette des groupes. Deux groupes d'une même palette ne peuvent pas avoir le même nom.

Exemple

L'instruction suivante crée le groupe gbGroupe2 dans l'acteur Séquence et une référence à ce groupe dans la variable *ng*.

```
ng = member("Séquence").newGroup("gbGroupe2")
```

newLight

Syntaxe

```
member(quelActeur).newLight(nomDeNouvelleLumière, #indicateurDeType)
```

Description

Commande 3D ; crée une lumière, *nomDeNouvelleLumière*, de type *#indicateurDeType*, et l'ajoute à la palette des lumières. Deux lumières d'une même palette ne peuvent pas avoir le même nom.

Les valeurs possibles du paramètre *#indicateurDeType* sont les suivantes :

- `#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)
```

newMesh

Syntaxe

```
member(quelActeur).newMesh(nom, nombreDeFaces, nombreDeSommets,
nombreDeNormales,nombreDeCouleurs,nombreDeCoordonnéesDeTexture)
```

Description

Commande 3D ; crée une ressource de modèle de maille avec les arguments fournis. Veuillez noter qu'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.

Les paramètres de `newMesh` sont les suivants :

- *nombreDeFaces* est le nombre total de triangles devant apparaître dans la maille.
- *nombreDeSommets* est le nombre total, facultatif, des sommets utilisés par toutes les faces (triangulaires). Un sommet peut être partagé par plus d'une face.
- *nombreDeNormales* est le nombre total, facultatif, 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 cet argument si vous prévoyez d'utiliser la commande `generateNormals()` de la maille pour générer les normales.

- *nombreDeCouleurs* est le nombre total, facultatif, des 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. Entez 0 ou ignorez cet argument pour obtenir la couleur blanche par défaut pour chaque coin de face.
- *nombreDeCoordonnéesDeTexture* est le nombre total, facultatif, des coordonnées de texture spécifié par l'utilisateur utilisées par toutes les faces. Entrez 0 ou ignorez cet argument 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 de coordonnées de textures n'est pas défini. 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 = [ 4,1,2 ]
nm.face[2].vertices = [ 4,2,3 ]
nm.face[3].vertices = [ 5,2,1 ]
nm.face[4].vertices = [ 5,3,2 ]
nm.face[5].vertices = [ 5,4,3 ]
nm.face[6].vertices = [ 5,1,4 ]
nm.face[1].colors = [3,2,3]
nm.face[2].colors = [3,3,2]
nm.face[3].colors = [1,3,2]
nm.face[4].colors = [1,2,3]
nm.face[5].colors = [1,3,2]
nm.face[6].colors = [1,2,3]
nm.generateNormals(#flat)
nm.build()
nm = member("Formes").newModel("Pyramide1", nm)
```

Voir aussi

newModelResource

newModel

Syntaxe

```
member( quelActeur ).newModel( nomDeNouveauModèle \
  {, quelleRessDeMod } )
```

Description

Commande 3D ; crée un modèle dans l'acteur référencé. Le *nomDeNouveauModèle* doit être unique, car la commande échoue si un modèle de ce nom existe déjà. La propriété de ressource de tous les nouveaux modèles est nulle par défaut. Vous pouvez utiliser le second paramètre facultatif pour spécifier une ressource de modèle à partir de laquelle créer le modèle.

Exemples

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", \
  Univers 3D").modelResource("grandeBoîte"))
```

newModelResource

Syntaxe

```
member(quelActeur).newModelResource(nomDeNouvRessDeModèle\
  { ,#type, #faisantFaceA })
```

Description

Commande 3D ; crée une ressource de modèle, du `#type` et `#faisantFaceA` (si spécifié), et l'ajoute à la palette de ressources de modèle.

Le paramètre *#type* peut être une des primitives suivantes :

```
#plane
#box
#sphere
#cylinder
#particle
```

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 seront générées. Les ressources de modèle de type `#plane` ont deux mailles générées (une pour chaque côté), et par conséquent, deux matériaux dans `shaderList`.

Le paramètre *#faisantFaceA* peut avoir l'une des valeurs suivantes :

- `#front`
- `#back`
- `#both`

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).

Exemples

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("boîteAchâpeaux4", #box)
```

Voir aussi

`primitives`

newMotion()

Syntaxe

`member(`*`quelActeur`*`).newMotion(`*`nom`*`)`

Description

Commande 3D ; crée un mouvement dans l'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()`. Tous les mouvements d'un acteur référencé doivent avoir un nom unique.

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()

Syntaxe

```
réfDimageObjetFlash.newObject("typeDobjet" {, arg1, arg2 ....})
newObject("typeDobjet" {, arg1, arg2 ....})
```

Description

Commande d'image-objet Flash ; crée un objet ActionScript du type spécifié. Tous les paramètres d'initialisation requis par l'objet peuvent être définis après le type d'objet. Chaque argument doit être séparé des autres par une virgule. La commande renvoie une référence au nouvel objet.

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 à la liste des Xtras de votre animation pour permettre aux objets Flash globaux de fonctionner correctement dans Shockwave et dans les projections. Les Xtras sont ajoutés à la liste des Xtras via Modification › Animation › Xtras. Pour plus d'informations, consultez la section Gestion des Xtras des animations distribuées dans l'aide de Director (Aide › Utilisation de Director › Options de distribution des animations).

Exemples

Cette instruction Lingo définit la variable *tObjetDeConnexionLocale* sur une référence à un nouvel objet `LocalConnection` dans l'animation Flash, au niveau de l'image-objet 3 :

```
tObjetDeConnexionLocale = sprite(3).newObject("ConnexionLocale")
```

L'instruction Lingo 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.

```
tObjetTableau = sprite(3).newObject("Array",23,34,19)
```

Voir aussi

`setCallback()`, `clearAsObjects()`

newShader

Syntaxe

`member(`*quelActeur*`).newShader(`*nomDeNouveauMatériau, #typeDeMatériau*`)`

Description

Commande 3D ; crée un matériau du *#typeDeMatériau* spécifié dans la liste des matériaux de l'acteur référencé et renvoie une référence au nouveau matériau. Tous les matériaux de la liste de matériaux doivent avoir un nom unique. L'argument *#typeDeMatériau* détermine le style d'application du matériau et ses valeurs possibles sont les suivantes :

- 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`.

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`.

Exemple

L'instruction suivante crée le matériau `#painter` nouveauPeintre.

```
nouveauPeintre = member("Univers 3D").newShader("nouveauPeintre",#painter)
```

Voir aussi

`shadowPercentage`

newTexture

Syntaxe

```
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. Toutes les textures de la palette de textures de l'acteur doivent avoir un nom unique. Les paramètres *`#indicateurDeType`* et *`référenceDobjetSource`* sont facultatifs et, s'ils ne sont pas spécifiés, une nouvelle texture sans type ni source sera créée. Les textures d'acteur ne fonctionnent que lorsque vous spécifiez l'acteur dans le constructeur `newTexture`.

Le paramètre *`#indicateurDeType`* peut avoir deux valeurs, `#fromCastMember` (un acteur) ou `#fromImageObject` (un objet image Lingo). Le paramètre `référenceDobjetSource` doit être une référence à un acteur si vous spécifiez `#fromCastMember` ou un objet image Lingo si vous spécifiez `#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")
```

next

Syntaxe

```
next
```

Description

Mot-clé ; fait référence au repère suivant de l'animation et revient à utiliser la phrase `the marker (+ 1)`.

Exemples

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 the keyCode = 124 then go next
  if the keyCode = 123 then go previous
end keyUp
```

Voir aussi

`loop (mot-clé)`, `go previous`

next repeat

Syntaxe

```
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
```

node

Syntaxe

```
sprite(quelleImageObjetQTVR).node
the node of sprite quelleImageObjetQTVR
```

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

Syntaxe

```
sprite(quelleImageObjetQTVR).nodeEnterCallback
the nodeEnterCallback of sprite quelleImageObjetQTVR
```

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

Syntaxe

```
sprite(quelleImageObjetQTVR).nodeExitCallback
the nodeExitCallback of sprite quelleImageObjetQTVR
```

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

Syntaxe

```
sprite(quelleImageObjetQTVR).nodeType
nodeType of sprite quelleImageObjetQTVR
```

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.

normalize

Syntaxe

```
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.

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, 0.0000, 1.0000 )
```

Voir aussi

`getNormalized`, `randomVector`, `magnitude`

normalList

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).\
  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`.

Exemples

```
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

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).\
  face[index].normals
```

Syntaxe

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

not

Syntaxe

```
not 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.

Exemples

L'instruction suivante détermine si 1 n'est pas inférieur à 2 :

```
put not (1 < 2)
```

Puisque 1 est inférieur à 2, le résultat est 0, ce qui indique que la valeur de l'expression est FALSE.

L'instruction suivante détermine si 1 n'est pas supérieur à 2 :

```
put not (1 > 2)
```

Puisque 1 n'est pas supérieur à 2, le résultat est 1, ce qui indique que la valeur de 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 :

```
on razDeLélémentDeMenu
  the checkMark of menuItem("Gras") of menu("Style") = \
  not (the checkMark of menuItem("Gras") of menu("Style"))
end razDeLélémentDeMenu
```

Voir aussi

`and`, `or`

nothing

Syntaxe

```
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.

Exemples

L'instruction imbriquée `if...then...else` du gestionnaire suivant utilise la commande `nothing` à la suite de la clause `else` de l'instruction :

```
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
```

Avec le gestionnaire suivant, l'animation n'évolue pas tant que l'utilisateur appuie sur le bouton de la souris :

```
on mouseDown
  repeat while the stillDown
    nothing
  end repeat
end mouseDown
```

Voir aussi

`if`

nudge

Syntaxe

```
sprite(quelleImageObjetQTVR).nudge(#direction )
nudge(sprite quelleImageObjetQTVR, #direction)
```

Description

Commande QuickTime VR ; déplace la perspective de l'image-objet QuickTime VR spécifiée dans la direction spécifiée par la valeur *#direction*. Les valeurs possibles de `#direction` sont `#down`, `#downLeft`, `#downRight`, `#left`, `#right`, `#up`, `#upLeft` et `#upRight`. 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.

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.

```
on mouseDown me
  repeat while the stillDown
    sprite(1).nudge(#left)
  end repeat
end
```

number (propriété de distribution)

Syntaxe

```
the number of castLib quelleDistribution
```

Description

Propriété de distribution ; indique le numéro de la distribution spécifiée. Par exemple, 2 est le numéro de castLib de la Distribution 2.

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 (propriété d'acteur)

Syntaxe

```
member(quelActeur).number
the number of member quelActeur
```

Description

Propriété d'acteur ; indique le numéro de l'acteur spécifié par *quelActeur* : ce peut être un nom, si *quelActeur* est une chaîne, ou un nombre, si *quelActeur* est un nombre entier.

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 distribution.

Cette propriété peut être testée, mais pas définie.

Remarque Si vous utilisez la première syntaxe, `member(quelActeur).number`, une erreur se produit si l'acteur n'existe pas. Si vous n'êtes pas certain de l'existence d'un acteur, utilisez l'autre syntaxe pour éviter cette erreur.

Exemples

L'instruction suivante donne le numéro de l'acteur Interrupteur à la variable *quelActeur* :

```
quelActeur = member("Interrupteur").number
```

L'instruction suivante affecte l'acteur Ballon rouge à l'image-objet 1 :

```
sprite(1).member = member("Ballon rouge").number
```

L'instruction suivante vérifie l'existence d'un acteur avant de changer l'acteur associé à l'image-objet :

```
property spriteNum

on mouseUp me
  if (member("Visage").number > 0) then
    sprite(spriteNum).member = "Visage"
  end if
end
```

Voir aussi

`member (propriété d'image-objet)`, `memberNum`, `number of members`

number (caractères)

Syntaxe

`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.

Exemples

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)

Syntaxe

```
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 de caractères d'une expression de sous-chaîne.

Exemples

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)

Syntaxe

```
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.

Exemples

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 donne à la variable *compteurDeLignes* le nombre de lignes contenues dans le 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 (menus)

Syntaxe

```
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.

Exemples

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)

Syntaxe

```
the number of menuItems of menu quelMenu
```

Description

Propriété de menu ; indique le nombre d'articles 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.

Exemples

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 (propriété système)

Syntaxe

```
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)

Syntaxe

```
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.

Exemples

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

Syntaxe

```
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

Syntaxe

```
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 standard des 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

Syntaxe

```
member(quelActeur).numChannels
the numChannels of member quelActeur
```

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 :

```
maVariable = member("Duke Ellington").numChannels
if maVariable = 1 then
  member("Affichage des pistes").text = "Mono"
else
  member("Affichage des pistes").text = "Stereo"
end if
```

numParticles

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).\
  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("Feux").modelResource("systèmeThermique").emitter.\
  numParticles = 50000
```

Voir aussi

`emitter`

numSegments

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).\
  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
```

numToChar()

Syntaxe

`numToChar(`*`expressionEntière`*`)`

Description

Fonction ; affiche une chaîne contenant le caractère dont le code ASCII est la valeur de *`expressionEntière`*. 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.

Exemples

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 :

```
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
```

Voir aussi

`charToNum()`

obeyScoreRotation

Syntaxe

```
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
```

objectP()

Syntaxe

```
objectP(expression)
```

Description

Fonction ; indique si l'expression spécifiée par *expression* 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.

Vous pourrez voir un exemple de objectP() dans une animation en consultant l'animation Read and Write Text du dossier Learning/Lingo_Examples, lui-même dans le dossier de Director.

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.

```
if objectP(gBaseDeDonnées) then
  nothing
else
  gBaseDeDonnées = script("Contrôleur de base de données").new()
end if
```

Voir aussi

floatP(), ilk(), integerP(), stringP(), symbolP()

of

Le mot of fait partie de nombreuses propriétés Lingo, telles que foreColor, number, name, etc.

offset() (fonction de chaîne)

Syntaxe

offset(*expressionChaîne1, expressionChaîne2*)

Description

Fonction ; renvoie un nombre entier indiquant la position du premier caractère de *expressionChaîne1* dans *expressionChaîne2*. Cette fonction renvoie 0 si *expressionChaîne1* ne se trouve pas dans *expressionChaîne2*. Lingo considère les espaces comme des caractères dans les deux chaînes.

Sur le Macintosh, la comparaison des chaînes ne tient pas compte de la casse. Ainsi, Lingo considère *a* et *Å* comme identiques sur le Macintosh.

Exemples

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 *sortie*, puis les remplace par la chaîne représentée par *chaîneAinsérer*.

```
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
```

Voir aussi

chars(), length(), contains, starts

offset() (fonction de rectangle)

Syntaxe

```
rectangle.offset(changementHorizontal, changementVertical)
offset (rectangle, changementHorizontal, changementVertical)
```

Description

Fonction ; produit un rectangle décalé par rapport au rectangle spécifié par *rectangle*. Le décalage horizontal est la valeur spécifiée par *changementHorizontal* ; le décalage vertical est la valeur spécifiée par *changementVertical*.

- 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.
- 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.

Les valeurs de *changementVertical* et de *changementHorizontal* sont exprimées en pixels.

Exemple

Le gestionnaire suivant déplace l'image-objet 1 de cinq pixels vers la droite et de cinq pixels vers le bas.

```
on mouvementDiagonal
  nouveauRect=sprite(1).rect.offset(5, 5)
  sprite(1).rect=nouveauRect
end
```

on

Syntaxe

```
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.

open

Syntaxe

```
open {quelDocument with} quelleApplication
```

Description

Commande ; lance l'application spécifiée par la chaîne *quelleApplication*. Utilisez *quelDocument* pour spécifier un document que l'application ouvre lorsqu'elle est lancée. Si l'application ou le document à ouvrir se trouvent dans un autre dossier que l'animation courante, vous devez spécifier leur chemin d'accès complet.

L'ordinateur doit avoir assez de mémoire pour exécuter simultanément Director et d'autres applications.

Cette commande est une commande très simple d'ouverture d'une application ou d'un document au sein d'une application. Pour d'autres contrôles, consultez les options disponibles dans les Xtras fournis par d'autres développeurs.

Exemples

L'instruction suivante vérifie si l'ordinateur est un Macintosh et, le cas échéant, ouvre l'application TextEdit :

```
if the platform contains "Mac" then open "TextEdit"
```

L'instruction suivante ouvre l'application TextEdit, qui se trouve dans le dossier Applications sur le disque dur, ainsi que le document Synopsis :

```
open "Synopsis" with "HD:Applications:TextEdit"
```

Voir aussi

`openXlib`

openResFile

Obsolète. Utilisez `recordFont`.

open window

Syntaxe

```
window(quelleFenêtre).open()
open window quelleFenêtre
```

Description

Commande de fenêtre ; ouvre l'objet fenêtre ou le fichier d'animation spécifié par *quelleFenêtre* et l'amène à l'avant de la scène. Si aucune animation n'est affectée à la fenêtre, la boîte de dialogue d'ouverture de fichier s'affiche.

- Si vous remplacez *quelleFenêtre* par le nom d'un fichier d'animation, la fenêtre utilise ce nom de fichier.
- Si vous remplacez *quelleFenêtre* par le nom d'une fenêtre, la fenêtre prend ce nom. Vous devez cependant affecter une animation à la fenêtre en utilisant `set the fileName of window`.

Pour ouvrir une fenêtre qui utilise une animation provenant d'une adresse URL, il est judicieux de télécharger le fichier sur votre disque local à l'aide de la commande `downloadNetThing`, puis d'utiliser ce fichier depuis votre disque local. Ceci minimise les problèmes d'attente nécessaire au téléchargement.

Pour les médias locaux, l'animation n'est chargée en mémoire qu'à l'exécution de la commande `open movie`. Ceci peut entraîner un retard important si vous n'utilisez pas `preloadMovie` pour charger au moins la première image de l'animation avant de lancer la commande `open window`.

Remarque L'ouverture d'une animation dans une fenêtre n'est pas encore supportée pour la lecture dans un navigateur web.

Exemple

L'instruction suivante ouvre la fenêtre Tableau de commande et la place au premier plan :

```
window("Tableau de commande").open()"
```

Voir aussi

`close window`, `downloadNetThing`, `preLoadMovie`

on openWindow

Syntaxe

```
on openWindow
  instruction(s)
end
```

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 Lingo 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 :

```
on openWindow
  puppetSound 2, "Hourra"
end
```

openXlib

Syntaxe

```
openXlib quelFichier
```

Description

Commande ; ouvre le fichier Xlibrary spécifié par la chaîne *quelFichier*. Si le fichier n'est pas dans le dossier de l'animation courante, *quelFichier* doit inclure le chemin.

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.

Sous Windows, l'extension .dll est facultative.

Remarque Cette commande n'est pas supportée dans Shockwave.

Exemples

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()`, `showXlib`

optionDown

Syntaxe

```
the optionDown
```

Description

Propriété système ; détermine si l'utilisateur a appuyé sur la touche Alt (Windows) ou Option (Macintosh) est enfoncée (`TRUE`) ou non (`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.

Si l'animation est lue avec le lecteur Director pour Java, cette fonction ne renvoie `TRUE` que si une deuxième touche est enfoncée en même temps que la touche Alt ou Option. Si seule la touche Alt ou Option est enfoncée, `optionDown` renvoie `FALSE`.

Le lecteur Director pour Java supporte les combinaisons de touches avec Alt ou Option. Le navigateur web reçoit cependant les signaux des touches avant la lecture de l'animation. Il y répond et intercepte toute combinaison servant également de raccourci clavier pour le navigateur.

Exemple

Le gestionnaire suivant vérifie si la touche Alt ou Option est enfoncée et, le cas échéant, appelle le gestionnaire `toucheDoption` :

```
on keyDown
  if (the optionDown) then toucheDoption(key)
end keyDown
```

Voir aussi

`controlDown`, `commandDown`, `key()`, `keyCode()`, `shiftDown`

or

Syntaxe

`expressionLogique1` `or` *`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.

Exemples

L'instruction suivante indique dans la fenêtre Messages si l'une au moins des expressions 1 < 2 et 1 > 2 est `TRUE` :

```
put (1 < 2) or (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 :

```
if member("Département").text = "Ariège" or member("Etat").text = "Gironde"
  then
  alert "Vous vous êtes perdu !"
end if
```

Voir aussi

`and`, `not`

organizationName

Syntaxe

```
the organizationName
```

Description

Propriété d'animation ; contient le nom de société entré lors de l'installation de Director.

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 :

```
on prepareMovie
  chaîneAffichée = the userName
  put RETURN & the organizationName after chaîneAffichée
  put RETURN & the serialNumber after chaîneAffichée
  member("Infos utilisateur").text = chaîneAffichée
end
```

Voir aussi

`serialNumber`, `userName`, `window`

originalFont

Syntaxe

```
member(quelActeurPolice).originalFont
the originalFont of member quelActeurPolice
```

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 :

```
put member(11).originalFont
-- "Monaco"
```

Voir aussi

`recordFont, bitmapSizes, characterSet`

originH

Syntaxe

```
sprite(quelleImageObjetVecteurOuFlash).originH
the originH of sprite quelleImageObjetVecteurOuFlash
member(quelleImageObjetVecteurOuFlash).originH
the originH of member quelleImageObjetVecteurOuFlash
```

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.

```
on beginSprite me
  sprite(spriteNum of me).originMode = #point
  sprite(spriteNum of me).originH = 100
  sprite(spriteNum of me).originV = 80
end
```

Voir aussi

`originV`, `originMode`, `originPoint`, `scaleMode`

originMode

Syntaxe

```
sprite(quelleImageObjetFlashOuFormeVectorielle ).originMode
the originMode of sprite quelleImageObjetFlashOuFormeVectorielle
member(quelleImageObjetFlashOuFormeVectorielle).originMode
the originMode of member quelleImageObjetFlashOuFormeVectorielle
```

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.

```
on beginSprite me
  sprite(spriteNum of me).originMode = #point
  sprite(spriteNum of me).originH = 100
  sprite(spriteNum of me).originV = 80
end
```

Voir aussi

`originH`, `originV`, `originPoint`, `scaleMode`

originPoint

Syntaxe

```
sprite quelleImageObjetVecteurOuFlash.originPoint
the originPoint of sprite quelleImageObjetVecteurOuFlash
member(quelleImageObjetVecteurOuFlash).originPoint
the originPoint of member quelleImageObjetVecteurOuFlash
```

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` comme 75.

Les valeurs de point Director spécifiées pour la propriété `originPoint` doivent être des nombres 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.

```
on beginSprite me
  sprite(me.spriteNum).scaleMode = #showAll
  sprite(me.spriteNum).originMode = #point
  sprite(me.spriteNum).originPoint = point(100, 80)
end
```

Voir aussi

`originH`, `originV`, `scaleMode`

originV

Syntaxe

```
sprite(quelleImageObjetVecteurOuFlash).originV
the originV of sprite quelleImageObjetVecteurOuFlash
member(quelActeurVecteurOuFlash).originV
the originV of member quelActeurVecteurOuFlash
```

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.

```
on beginSprite me
  sprite(me.spriteNum).scaleMode = #showAll
  sprite(me.spriteNum).originMode = #point
  sprite(me.spriteNum).originH = 100
  sprite(me.spriteNum).originV = 80
end
```

Voir aussi

`originH`, `originPoint`, `scaleMode`

otherwise

Syntaxe

```
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 (the key) of
    "A": go to frame "Pomme"
    "B", "C":
      puppetTransition 99
       go to frame "Oranges"
    otherwise:
      alert "Cette touche n'est pas valide."
  end case
end keyDown
```

orthoHeight

Syntaxe

```
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.

Veuillez noter qu'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

Syntaxe

```
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 rect 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

Syntaxe

```
member(quelActeur).pageHeight
the pageHeight of member quelActeur
```

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 :

```
put member("Nouvelles du jour").pageHeight"
```

palette

Syntaxe

```
member(quelActeur).palette
the palette of member quelActeur
```

Description

Propriété d'acteur ; détermine, pour les acteurs bitmap uniquement, la palette associée à l'acteur spécifié par *quelActeur*.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante affiche la palette affectée à l'acteur Feuilles dans la fenêtre Messages :

```
put member("Feuilles").palette"
```

paletteMapping

Syntaxe

```
the 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. Son effet 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 :

```
set the paletteMapping = TRUE
```

paletteRef

Syntaxe

```
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 (propriété QTVR)

Syntaxe

```
pan of sprite quelleImageObjetQTVR
```

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.

pan (propriété audio)

Syntaxe

```
sound(numéroDePiste).pan
the pan of sound(numéroDePiste)
```

Description

Propriété ; indique la balance gauche/droite du son en cours de lecture sur la piste audio *numéroDePiste*. 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 (propriété audio) dans une animation en consultant l'animation Sound Control du dossier Learning/Lingo_Examples, lui-même dans le dossier de Director.

Exemple

L'instruction suivante équilibre le son de la piste audio 2, de la piste gauche vers la piste droite :

```
repeat with x = -100 to 100
  sound(2).pan = x
end repeat
```

Voir aussi

`fadeIn()`, `fadeOut()`, `fadeTo()`, `volume (piste audio)`

paragraph

Syntaxe

`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 de chariot.

```
put member("Texte Animé").paragraph[3]
```

Voir aussi

`line...of`

param()

Syntaxe

`param(`*`positionDeParamètre`*`)`

Description

Fonction ; renvoie la valeur d'un paramètre transmis à un gestionnaire. L'expression *`positionDeParamètre`* représente la position du paramètre dans les arguments.

Cette fonction peut s'utiliser pour déterminer le type d'un paramètre particulier afin d'éviter les erreurs dans un gestionnaire.

Exemple

Le gestionnaire suivant accepte un nombre quelconque d'arguments, fait le total du nombre transmis en paramètres, puis renvoie la somme :

```
on ajouterLesNombres
  somme = 0
  repeat with numDeParamCourant = 1 to the paramCount
    somme = somme + param(numDeParamCourant)
  end repeat
  return somme
end
```

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()

Syntaxe

```
the paramCount
```

Description

Fonction ; indique le nombre de paramètres transmis au gestionnaire courant.

Exemple

L'instruction suivante définit la variable *compteur* en fonction du nombre de paramètres transmis au gestionnaire courant :

```
set compteur = the paramCount
```

parent

Syntaxe

```
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 Univers (`group("Univers")`) 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`, `addChild`

parseString()

Syntaxe

`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'analyse, le second paramètre étant la chaîne contenant les données XML. 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()`.

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()

Syntaxe

`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 XML Parser, le second étant l'adresse URL à laquelle les données XML résident.

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. Le troisième paramètre facultatif est le nom d'un gestionnaire qui devrait être exécuté une fois l'URL entièrement analysée, le quatrième paramètre facultatif étant le nom de l'objet script contenant ce gestionnaire. Si le quatrième paramètre n'est pas transmis, le nom du gestionnaire du troisième paramètre est considéré comme étant un gestionnaire de l'animation.

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().

Exemples

L'instruction suivante analyse le fichier exemple.xml qui se trouve sur www.monEntreprise.fr. Vous utiliserez doneParsing() pour être averti(e) de la fin de l'opération d'analyse.

```
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

Syntaxe

```
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.

Le lecteur Director pour Java ne supporte cette commande que dans les gestionnaires `on keyDown` et `on keyUp` liés à des images-objets modifiables.

La commande `pass` passe à la position suivante dès son exécution. Aucun élément Lingo 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.

Exemple

Le gestionnaire suivant détermine les touches enfoncées et en permet la transmission à l'image-objet texte modifiable s'il s'agit de caractères valides :

```
on keyDown me
  caractèresAdmis = "1234567890"
  if caractèresAdmis contains the key then
    pass
  else
    beep
  end if
end
```

Voir aussi

```
stopEvent
```

password

Syntaxe

```
sprite(quelleImageObjet).password
member(quelActeur).password
sprite(quelleImageObjet).password = motDePasse
member(quelActeur).password = motDePasse
```

Description

Propriété d'acteur et image-objet RealMedia ; permet de définir le mot de passe nécessaire à l'accès à un train 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'encore été défini, la valeur de cette propriété est une chaîne vide.

Exemples

Les exemples suivants indiquent que le mot de passe a été défini pour le train RealMedia de l'acteur Real ou de l'image-objet 2.

```
put sprite(2).password
-- "********"

put member("Real").password
-- "********"
```

Les exemples suivants indiquent que le mot de passe n'a jamais été défini pour le train RealMedia de l'acteur Real ou de l'image-objet 2.

```
put sprite(2).password
-- ""

put member("Real").password
-- ""
```

Les exemples suivants définissent le mot de passe pour le train RealMedia de l'image-objet 2 et l'acteur Real à *abracadabra*.

```
sprite(2).password = "abracadabra"
member("Real").password = "abracadabra"
```

Voir aussi

`userName (RealMedia)`

pasteClipBoardInto

Syntaxe

```
member(quelActeur). pasteClipBoardInto()
pasteClipBoardInto member quelActeur
```

Description

Commande ; colle le contenu du Presse-papiers dans l'acteur spécifié par *quelActeur* et efface l'acteur existant. Par exemple, le collage d'un bitmap sur un acteur champ transforme le bitmap en acteur et efface l'acteur champ.

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.

La commande `pasteClipBoardInto` 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.

Remarque Lorsque vous utilisez cette commande dans Shockwave ou dans l'environnement auteur et les 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.

Exemple

L'instruction suivante colle le contenu du Presse-papiers dans l'acteur bitmap Temple :

```
member("Temple").pasteClipBoardInto()
```

Voir aussi

`safePlayer`

path

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).\
  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 (propriété d'acteur)

Syntaxe

```
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 (propriété d'acteur), linked

pathName (propriété d'animation)

Obsolète. Utilisez `moviePath`.

pathStrength

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).\
  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("Feux").modelResource("systèmeThermique").emitter.\
  pathStrength = 0.97
```

Voir aussi

`path`, `emitter`

pattern

Syntaxe

```
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.

Le lecteur Director pour Java ne peut affecter que les motifs des puces 1 et 15 de la palette des motifs de Director.

Cette propriété peut être utile dans les animations 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.

Exemples

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
```

pause (lecture d'animation)

Obsolète. Utilisez `go to the frame`.

pause() (3D)

Syntaxe

```
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`.

Exemple

L'instruction suivante met l'animation courante du modèle fourmi3 en pause.

```
member("Picnic").model("fourmi3").bonesplayer.pause()
```

Voir aussi

`play() (3D)`, `playing (3D)`, `playlist`

pause (RealMedia)

Syntaxe

```
sprite(quelleImageObjet).pause()
member(quelActeur).pause()
```

Description

Méthode d'acteur ou d'image-objet RealMedia ; interrompt la lecture du train multimédia. La valeur de `mediaStatus` devient `#paused`.

L'appel de cette méthode, lors de la lecture du train RealMedia, 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 train RealMedia en tampon.

Exemples

Les exemples suivants arrêtent la lecture de l'image-objet 2 ou de l'acteur Real.

```
sprite(2).pause()
member("Real").pause()
```

Voir aussi

`mediaStatus`, `play (RealMedia)`, `seek`, `stop (RealMedia)`

pause() (lecture audio)

Syntaxe

```
sound(numéroDePiste).pause()
pause(sound(numéroDePiste))
```

Description

Cette commande interrompt la lecture du son courant dans la piste audio *`numéroDePiste`*. Une prochaine commande `play()` entraînera la reprise de la lecture.

Exemple

L'instruction suivante met en pause la lecture de l'acteur son lu dans la piste audio 1 :

```
sound(1).pause()
```

Voir aussi

`breakLoop()`, `isBusy()`, `play() (audio)`, `playNext()`, `queue()`, `rewind()`, `status`, `stop() (audio)`

pausedAtStart (Flash, vidéo numérique)

Syntaxe

```
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
```

Voir aussi

play

pausedAtStart (RealMedia)

Syntaxe

```
sprite(quelleImageObjet).pausedAtStart
member(quelActeur).pausedAtStart
```

Description

Propriété d'acteur ou image-objet RealMedia ; permet d'obtenir ou de définir si la lecture d'un train RealMedia présent sur la scène démarre automatiquement à la fin de la mise en tampon (FALSE) ou non (TRUE). 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 sur un bouton que vous avez créé dans ce but dans votre animation), ou appeler la commande Lingo 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.

Exemples

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 train RealMedia sera automatiquement lu à la fin de la mise en tampon.

```
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 train RealMedia ne sera pas lu jusqu'à l'appel de la commande play.

```
sprite(2).pausedAtStart = TRUE

member("Real").pausedAtStart = TRUE
```

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 :

```
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
```

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.

pause member

Syntaxe

```
member(quelActeur). pause()
pause member ("quelActeur")
```

Description

Commande ; met en pause le flux d'un acteur Shockwave Audio (SWA). Lorsque le son est en état de pause, la propriété `state` de l'acteur est égale à 4. La lecture de la portion du son déjà téléchargée et disponible se poursuit jusqu'à épuisement du cache.

Exemple

Le gestionnaire suivant peut servir de bouton lecture/pause. Si le son est en cours de lecture, le gestionnaire le met en pause ; sinon, le gestionnaire lit le son lié à l'acteur son SWA en flux continu.

```
on mouseDown
  quelEtat = member("SWA").state
  if quelEtat = 3 then
    member("SWA").pause()
  else
    member("SWA").play()
  end if
end
```

Voir aussi

`play member`, `stop member`

pause sprite

Syntaxe

```
sprite(quelNuméroDimageObjetGIF).pause()
pause(sprite quelNuméroDimageObjetGIF)
```

Description

Commande ; met en pause la lecture d'une image-objet GIF animée et reste sur l'image courante.

Exemple

```
sprite(1).pause()
```

Voir aussi

resume sprite, rewind sprite

percentBuffered

Syntaxe

```
sprite(quelleImageObjet).percentBuffered
member(quelActeur).percentBuffered
```

Description

Propriété d'acteur ou d'image-objet RealMedia ; renvoie le pourcentage du tampon rempli par le train 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 train 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 train 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 train 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 train. L'utilisation de la commande `pausedAtStart` permet de lire le train en dehors de la scène pendant le processus de mise en mémoire tampon, puis d'afficher le train sur la scène lorsque la lecture démarre. Pour plus d'informations, consultez l'entrée `pausedAtStart (RealMedia)`.

Exemples

Les exemples suivants indiquent que 56 % du train RealMedia de l'image-objet 2 et de l'acteur Real a été mis en tampon.

```
put sprite(2).percentBuffered
-- 56

put member("Real").percentBuffered
-- 56
```

Voir aussi

mediaStatus, pausedAtStart (RealMedia), state (RealMedia)

pauseState

Syntaxe

```
the pauseState
```

Description

Propriété d'animation ; détermine si l'animation est actuellement interrompue (`TRUE`) ou non (`FALSE`) avec la commande `pause`.

La commande `pause` étant obsolète, cette propriété est rarement utilisée.

Exemple

L'instruction suivante vérifie si l'animation est interrompue et, le cas échéant, en continue la lecture :

```
if the pauseState = TRUE then go the frame + 1
```

Voir aussi

`pause (lecture d'animation)`

percentPlayed

Syntaxe

```
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`

percentStreamed

Syntaxe

```
member(quelActeurSWAouFlash).percentStreamed
the percentStreamed of member quelActeurSWAouFlash
sprite(quelleImageObjetQuickTime).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 :

```
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
```

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 :

```
on exitFrame
  if member("Ecran d'accueil").percentStreamed < 60 then
    go to the frame
  end if
 end
```

Voir aussi

`percentPlayed`

percentStreamed (3D)

Syntaxe

`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
```

period

Syntaxe

`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 (propriété de temporisation)`, `persistent`, `target`, `time (propriété d'objet de temporisation)`, `timeout()`, `timeoutHandler`, `timeoutList`

perpendicularTo

Syntaxe

`vecteur1``.perpendicularTo(`*`vecteur2`*`)`

Description

Commande 3D de vecteur ; renvoie un vecteur perpendiculaire au vecteur d'origine et à un second vecteur (*`vecteur2`*). Cette commande équivaut à la commande de vecteur `crossProduct`. Pour plus d'informations, consultez `crossProduct()`.

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`

persistent

Syntaxe

`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 (propriété de temporisation)`, `period`, `target`, `time (propriété d'objet de temporisation)`, `timeout()`, `timeoutHandler`, `timeoutList`

PI

Syntaxe

```
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 :

```
set vSurface = PI*power(vRayon,2)
```

picture (propriété d'acteur)

Syntaxe

```
member(quelActeur).picture
the picture of member quelActeur
```

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 :

```
contenuPict = member("Crépuscule").picture
```

Voir aussi

type (propriété d'image-objet)

picture (propriété de fenêtre)

Syntaxe

```
the stage.picture
the picture of the stage
window quelleFenêtre.picture
the picture of window quelleFenêtre
```

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).

Vous pouvez appliquer les données du bitmap résultant à un bitmap existant ou les utiliser pour en créer un nouveau.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante saisit le contenu courant de la scène et le place dans un acteur bitmap :

```
member("Image de la scène").picture = (the stage).picture
```

Voir aussi

media, picture (propriété d'acteur)

pictureP()

Syntaxe

```
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.

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*.

```
set valeurDimage to the picture of member "Temple"
put pictureP(valeurDimage)
```

Le résultat est 1, équivalent numérique de `TRUE`.

platform

Syntaxe

```
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`.

Lorsque l'animation est lue en tant qu'applet Java convertie, la valeur de cette propriété indique le navigateur web et le système d'exploitation dans lesquels elle est lue. La valeur de la propriété respecte la syntaxe suivante lorsque l'animation est lue en tant qu'applet :

`Java` *`versionJava`*`,` *`navigateurWeb`*`,` *`systèmeDexploitation`*

Les valeurs possibles pour les paramètres de cette propriété sont les suivantes :

- *`versionJava`* : `1.0` ou `1.1`.
- *`navigateurWeb`* : `IE`, `Netscape` ou `UnknownBrowser`
- *`systèmeDexploitation`* : `Macintosh`, `Windows` ou `UnknownOS`

Par exemple, si une applet est lue dans Internet Explorer avec Java 1.1 sous Windows, `platform` a la valeur `Java 1.1, IE, Windows`.

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`

play

Syntaxe

```
sprite(quelleImageObjetFlash).play()
play [frame] quelleImage
play movie quelleAnimation
play frame quelleImage of movie quelleAnimation
play sprite quelleImageObjetFlash
```

Description

Commande ; fait passer la tête de lecture à l'image spécifiée de l'animation sélectionnée ou lance la lecture d'une image-objet animation Flash. Dans le premier cas, l'expression *quelleImage* peut être soit un repère de chaîne, soit un numéro entier d'image. L'expression *quelleAnimation* doit être une chaîne spécifiant un fichier d'animation. Si l'animation se trouve dans un autre dossier, *quelleAnimation* doit spécifier son chemin d'accès.

La commande `play` est similaire à la commande `go to` à la différence que lorsque la séquence en cours termine sa lecture, la commande `play` déplace automatiquement la tête de lecture sur l'image à partir de laquelle la commande `play` a été appelée.

Si la commande `play` est issue d'un script d'image, la tête de lecture revient à l'image suivante ; si elle est issue d'un script d'image-objet ou d'un gestionnaire, la tête de lecture revient à la même image. Une séquence de lecture est terminée lorsque la tête de lecture atteint la fin de l'animation ou lorsque la commande `play done` est émise.

Pour lire une animation depuis une adresse URL, utilisez `downloadNetThing` ou `preloadNetThing()` pour télécharger le fichier sur un disque local. Utilisez ensuite la commande `play` pour lire l'animation sur le disque local. Cette procédure permet de réduire le temps de chargement du fichier.

La commande `play` peut également servir à lire plusieurs animations à partir d'un même gestionnaire. L'exécution du gestionnaire est interrompue pendant la lecture de chaque animation, pour reprendre à la fin de chacune. Par comparaison, une série de commandes `go` appelées par un gestionnaire lisent la première image de chaque animation. L'exécution du gestionnaire n'est pas interrompue pendant la lecture de l'animation, mais se poursuit sans délai.

Lorsque la commande `play` est utilisée pour lire une image-objet animation Flash, cette animation est lue à partir de son image courante si elle est arrêtée ou à partir de sa première image si elle est déjà sur la dernière.

Chaque commande `play` doit être associée à une commande `play done` correspondante afin d'éviter de saturer la mémoire si vous ne retournez pas au script d'appel initial. Pour ce faire, vous pouvez utiliser une variable globale spécifiant où l'animation doit reprendre.

Exemples

L'instruction suivante place la tête de lecture sur le repère Clignotement :

```
play "Clignotement"
```

L'instruction suivante place la tête de lecture sur le repère suivant :

```
play marker(1)
```

L'instruction suivante déplace la tête de lecture jusqu'à une autre animation :

```
play movie "Mon disque:Autres animations:" & nouvelleAnimation
```

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 :

```
on enterFrame
  if not sprite(5).playing then
    sprite(5).play()
  end if
end
```

Voir aussi

downloadNetThing

play() (3D)

Syntaxe

```
member(quelActeur).model(quelModèle).bonesPlayer.play()
member(quelActeur).model(quelModèle).keyframePlayer.play()
member(quelActeur).model(quelModèle).bonesPlayer.\
  play(nomDuMouvement {, enBoucle, posInitiale, posFinale, échelle, décalage})
member(quelActeur).model(quelModèle).keyframePlayer.\
  play(nomDuMouvement {, enBoucle, posInitiale, posFinale, 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().

Lorsque play() est appelée et que seul le paramètre *nomDeMouvement* est spécifié, le mouvement est exécuté par le modèle, une fois du début à la fin, à la cadence définie par la propriété playRate du modificateur.

Les paramètres facultatifs de la commande play sont les suivants :

enBoucle spécifie si le mouvement est lu une seule fois (FALSE) ou continuellement (TRUE).

posInitiale 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 *posFinale*, avec toutes les répétitions suivantes du mouvement démarrant à *posInitiale* et se terminant à *posFinale*.

posFinale 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 *posFinale*. Lorsque *enBoucle* a pour valeur TRUE, la première itération de la boucle commence à *décalage* et se termine à *posFinale*, avec toutes les répétitions suivantes démarrant à *posInitiale* et se terminant à *posFinale*. Donnez à *posFinale* la valeur -1 si le mouvement doit être lu jusqu'à la fin.

échelle est multiplié par la propriété playRate du modificateur #keyframePlayer ou #bonesPlayer du modèle pour déterminer la cadence réelle de la lecture du mouvement.

décalage 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 *posFinale*. Lorsque *enBoucle* a pour valeur TRUE, la première itération de la boucle commence à *décalage* et se termine à *posFinale*, avec toutes les répétitions suivantes démarrant à *posInitiale* et se terminant à *posFinale*. 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.

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.

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, playlist, pause() (3D), removeLast(), playing (3D)

play (RealMedia)

Syntaxe

```
sprite(quelleImageObjet).play()
member(quelActeur).play()
```

Description

Méthode d'acteur ou d'image-objet RealMedia ; démarre le processus de lecture en flux continu pour le train RealMedia si le train est fermé ou reprend la lecture si le train est en pause. Si nécessaire, le train sera mis en mémoire tampon avant le début de la lecture. La valeur de mediaStatus devient #playing.

Exemples

Les exemples suivants démarrent le processus de lecture en flux continu de l'image-objet 2 et de l'acteur Real.

```
sprite(2).play()
member("Real").play()
```

Voir aussi

`mediaStatus`, `pause (RealMedia)`, `seek`, `stop (RealMedia)`

play() (audio)

Syntaxe

```
sound(numéroDePiste).play()
sound(numéroDePiste).play(member (quelActeur))
sound(numéroDePiste).play([#member: member(quelActeur), {#startTime:
  millisecondes, #endTime: millisecondes, #loopCount: nombreDeBoucles,
  #loopStartTime: millisecondes, #loopEndTime: millisecondes, #preloadTime:
  millisecondes}])
```

Description

Cette fonction entame la lecture de tous les sons placés en file d'attente dans *objetAudio* ou place l'acteur donné en file d'attente et en entame la lecture.

Le chargement des acteurs son en mémoire RAM requiert un certain temps avant de pouvoir entamer la lecture. 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 fonction. 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. 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. Pour plus d'informations, consultez `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`.

Vous pourrez voir un exemple de `play() (son)` dans une animation en consultant l'animation Sound Control du dossier Learning/Lingo_Examples, lui-même dans le dossier de Director.

Exemples

L'instruction suivante lit l'acteur son Intro sur la piste audio 1 :

```
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.

```
sound(2).play([#member: member("Crédits"), #startTime: 4000, \
#endTime: 15000, #loopCount: 6, #loopStartTime: 10500, #loopEndTime: 14000])
```

Voir aussi

`setPlaylist()`, `isBusy()`, `pause() (lecture audio)`, `playNext()`, `preLoadTime`, `queue()`, `rewind()`, `stop() (audio)`

playBackMode

Syntaxe

```
sprite(quelleImageObjetFlash).playBackMode
the playBackMode of sprite quelleImageObjetFlash
member(quelActeurAnimationGIF).playBackMode
the playBackMode of member quelActeurAnimationGIF
```

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 :

```
property spriteNum
on beginSprite me
  sprite(spriteNum).playBackMode = #lockStep
end
```

Voir aussi

`fixedRate`

play done

Syntaxe

```
play done
```

Description

Commande ; indique la fin de la séquence lancée par la commande `play` la plus récente. La commande `play done` renvoie la tête de lecture à la position où la séquence initiale avait lancé la commande `play`. Si la commande `play` est issue d'un script d'image, la tête de lecture revient à l'image suivante ; si elle est issue d'un script d'image-objet, la tête de lecture revient à la même image.

Chaque commande `play` doit être associée à une commande `play done` correspondante afin d'éviter de saturer la mémoire si vous ne retournez pas au script d'appel initial. Pour ce faire, vous pouvez utiliser une variable globale spécifiant où l'animation doit reprendre.

La commande `play done` n'a aucun effet sur les animations dans une fenêtre.

Exemple

Le gestionnaire suivant renvoie la tête de lecture sur l'image de l'animation qui était lue avant le début de l'animation courante :

```
on exitFrame
  play done
end
```

Voir aussi

`play`

playing

Syntaxe

```
sprite(quelleImageObjetFlash).playing
the playing of sprite quelleImageObjetFlash
```

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 :

```
on enterFrame
  if not sprite(5).playing then
    sprite(5).play()
  end if
end
```

playing (3D)

Syntaxe

```
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("nouveauMartiens").model("Martien3").keyframePlayer.playing
-- 1
```

Voir aussi

play() (3D), pause() (3D), playlist, queue() (3D)

playlist

Syntaxe

```
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
-- [[#nom: "Marche", #loop: 1, #startTime: 1500, #endTime: 16000, \
  #scale:1.0000, #offset: 0], [#nom: "Saut", #loop: 1, \
  #startTime: 0, #endTime: 1200, #scale: 1.0000, #offset: 0]]
```

Voir aussi

`play() (3D)`, `playNext() (3D)`, `removeLast()`, `queue() (3D)`

play member

Syntaxe

```
member(quelActeur).play()
play member quelActeur
```

Description

Commande ; lance la lecture en flux continu d'un acteur Shockwave Audio (SWA).

Si le son SWA n'a pas été préchargé à l'aide de la commande `preLoadBuffer`, il est préchargé avant le début de sa lecture. Pendant la lecture du son, la propriété d'acteur `state` est égale à 3.

Veuillez noter que, pour supporter cette fonctionnalité, les Xtras doivent être inclus lors de la lecture d'un son en flux continu.

Exemple

Le gestionnaire suivant lance la lecture de l'acteur Orchestre :

```
on mouseDown
  member("Orchestre").play()
end
```

Voir aussi

`pause member`, `stop member`

playNext()

Syntaxe

```
sound(numéroDePiste).playNext()
playNext(sound(numéroDePiste))
```

Description

Cette commande 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.

Exemple

L'instruction suivante exécute la lecture sur la piste audio 2 du son suivant placé en file d'attente.

```
sound(2).playNext()
```

Voir aussi

`pause() (lecture audio)`, `play() (audio)`, `stop() (audio)`

playNext() (3D)

Syntaxe

```
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("séquence").model[1].bonesPlayer.playnext()
```

Voir aussi

blend (3D), playlist

playRate

Syntaxe

```
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("nouveauMartiens").model("MartienVert").keyframePlayer.playRate = 3
```

Voir aussi

play() (3D), queue() (3D), playlist, currentTime (3D)

point()

Syntaxe

`point(horizontal, vertical)`

Description

Fonction et type de données ; fournit un point dont les coordonnées horizontale et verticale sont spécifiées par *horizontal* et *vertical*.

Un point a une propriété `locH` et une propriété `locV`. Les coordonnées d'un point peuvent être changées par des opérations arithmétiques.

Vous pourrez voir un exemple de `point()` dans une animation en consultant les animations Imaging et Vector Shapes du dossier Learning/Lingo_Examples, lui-même dans le dossier de Director.

Exemples

L'instruction suivante définit la variable *dernièrePosition* au point (250, 400) :

```
set dernièrePosition = point(250, 400)
```

L'instruction suivante ajoute 5 pixels à la coordonnée horizontale du point affecté à la variable *monPoint* :

```
monPoint.locH = monPoint.locH + 5
```

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.

```
sprite(the clickOn).loc = point(the mouseH, the mouseV) + point(10, 10)
sprite(the clickOn).loc = the mouseLoc + 10
```

Le gestionnaire suivant place une image-objet nommée à l'endroit sur lequel l'utilisateur clique :

```
end mouseDown

on mouseDown
  -- définissez ces variables selon votre animation
  imageObjet = 1 -- Définissez l'image-objet à déplacer
  étapes = 40 -- Définissez le nombre d'étapes du déplacement
  emplacementInitial = sprite(imageObjet).loc
  delta = (the clickLoc - emplacementInitial) / étapes
  repeat with i = 1 to étapes
    sprite(ImageObjet).loc = emplacementInitial + (i * delta)
    updateStage
  end repeat
end mouseDown
```

Voir aussi

`mouseLoc`, `flashToStage()`, `rect()`, `stageToFlash()`

pointAt

Syntaxe

```
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 la position relative à l'univers spécifiée par *positionDuVecteur*, puis fait pivoter l'objet référencé pour que son vecteur vertical pointe vers la position relative à l'univers spécifiée par *vecteurVertical*. La valeur de *positionDuVecteur* peut également être une référence de nœud.

Le paramètre facultatif *vecteurVertical* est 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, cette commande 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.

Le vecteur vertical et le vecteur horizontal de l'objet sont définis par la propriété `pointAtOrientation` de l'objet.

Exemples

L'exemple suivant dirige trois objets vers le modèle Mars : la caméra camMars, la lumière Spot et le modèle Pistolet.

```
posDansCetUnivers = member("Séquence").model("Mars").worldPosition
member("Séquence").camera("camMars").pointAt(posDansCetUnivers)
member("Séquence").light("Spot").pointAt(posDansCetUnivers)
member("Séquence").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 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 non par défaut
nœud.transform.scale = scale
```

Voir aussi

`pointAtOrientation`

pointAtOrientation

Syntaxe

```
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`

pointInHyperlink()

Syntaxe

```
sprite(quelNuméroDimageObjet).pointInHyperlink(point)
pointInHyperlink(sprite quelNuméroDimageObjet, 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.

Voir aussi

`cursor (commande)`, `mouseLoc`

pointOfContact

Syntaxe

```
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`

pointToChar()

Syntaxe

```
pointToChar(sprite numéroDimageObjet, pointAConvertir)
```

Description

Fonction ; renvoie un nombre entier représentant la position du caractère situé dans l'image-objet texte ou champ *numéroDimageObjet* à la coordonnée d'écran de *pointAConvertir* ou renvoie -1 si le point n'est pas dans le texte.

Cette fonction permet de déterminer le caractère sous le curseur.

Exemple

L'instruction suivante affiche le numéro du caractère cliqué, ainsi que la lettre, dans la fenêtre Messages :

```
property spriteNum

 on mouseDown me
  pointCliqué = the mouseLoc
  acteurCourant = sprite(spriteNum).member
  numDuCaractère = sprite(spriteNum).pointToChar(pointCliqué)
  caractère = acteurCourant.char[numDuCaractère]
  put "Caractère sélectionné" && numDuCaractère & ", lettre" && caractère
end
```

Voir aussi

mouseLoc, pointToWord(), pointToItem(), pointToLine(), pointToParagraph()

pointToItem()

Syntaxe

```
sprite(quelNuméroDimageObjet).pointToItem(pointAconvertir)
pointToItem(sprite numéroDimageObjet, pointAconvertir)
```

Description

Fonction ; renvoie un nombre entier représentant la position de l'élément situé dans l'image-objet texte ou champ *numéroDimageObjet* à la coordonnée d'écran de *pointAConvertir* 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.

Exemple

L'instruction suivante affiche le numéro de l'élément cliqué, ainsi que son texte, dans la fenêtre Messages :

```
property spriteNum

on mouseDown me
  pointCliqué = the mouseLoc
  acteurCourant = sprite(spriteNum).member
  numéroDélément = sprite(numéroDimageObjet).pointToItem(pointCliqué)
  texteDélément = acteurCourant.item[numéroDélément]
  put "Elément sélectionné" && numéroDélément & ", le texte" && texteDélément
end
```

Voir aussi

itemDelimiter, mouseLoc, pointToChar(), pointToWord(), pointToItem(), pointToLine(), pointToParagraph()

pointToLine()

Syntaxe

```
sprite(quelNuméroDimageObjet).pointToLine(pointAconvertir)
pointToLine(sprite numéroDimageObjet, pointAconvertir)
```

Description

Fonction ; renvoie un nombre entier représentant la ligne de l'élément situé dans l'image-objet texte ou champ *numéroDimageObjet* à la coordonnée d'écran de *pointAConvertir* 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.

Exemple

L'instruction suivante affiche le numéro de la ligne cliquée, ainsi que son texte, dans la fenêtre Messages :

```
property spriteNum

 on mouseDown me
  pointCliqué = the mouseLoc
  acteurCourant = sprite(spriteNum).member
  numDeLigne = sprite(numéroDimageObjet).pointToLine(pointCliqué)
  texteDeLigne = acteurCourant.line[numDeLigne]
  put "Ligne sélectionnée" && numDeLigne & ", le texte" && texteDeLigne
end
```

Voir aussi

itemDelimiter, mouseLoc, pointToChar(), pointToWord(), pointToItem(), pointToLine(), pointToParagraph()

pointToParagraph()

Syntaxe

```
sprite(quelNuméroDimageObjet).pointToParagraph(pointAconvertir)
pointToParagraph(sprite numéroDimageObjet, pointAconvertir)
```

Description

Fonction ; renvoie un nombre entier représentant le numéro du paragraphe situé dans l'image-objet texte ou champ *numéroDimageObjet* à la coordonnée d'écran de *pointAConvertir* 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.

Exemple

L'instruction suivante affiche le numéro du paragraphe cliqué, ainsi que son texte, dans la fenêtre Messages :

```
property spriteNum

 on mouseDown me
  pointCliqué = the mouseLoc
  acteurCourant = sprite(spriteNum).member
  numDeParagraphe = sprite(spriteNum).pointToParagraph(pointCliqué)
  texteDuParagraphe = acteurCourant.paragraph[numDeParagraphe]
  put "Paragraphe sélectionné" && numDeParagraphe & ", le texte" &&
  texteDuParagraphe
end
```

Voir aussi

itemDelimiter, mouseLoc, pointToChar(), pointToWord(), pointToItem(), pointToLine()

pointToWord()

Syntaxe

```
sprite(quelNuméroDimageObjet).pointToWord(pointAconvertir)
pointToWord(sprite numéroDimageObjet, pointAconvertir)
```

Description

Fonction ; renvoie un nombre entier représentant le numéro du mot situé dans l'image-objet texte ou champ *numéroDimageObjet* à la coordonnée d'écran de *pointAConvertir* 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.

Exemple

L'instruction suivante affiche le numéro du mot cliqué, ainsi que son texte, dans la fenêtre Messages :

```
property spriteNum

 on mouseDown me
   pointCliqué = the mouseLoc
   acteurCourant = sprite(spriteNum).member
   numDuMot = sprite(numéroDimageObjet).pointToWord(pointCliqué)
   texteDuMot = acteurCourant.word[numDuMot]
   put "Mot sélectionné" && numDuMot & ", le texte" && texteDuMot
end
```

Voir aussi

`itemDelimiter`, `mouseLoc`, `pointToChar()`, `pointToItem()`, `pointToLine()`, `pointToParagraph()`

position (transformation)

Syntaxe

```
member(quelActeur).node(quelNœud).transform.position
member(quelActeur).node(quelNœud).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.

Exemples

L'instruction suivante affiche la position relative au parent du modèle Pneu.

```
put member("Séquence").model("Pneu").transform.position
--vector(-15.000, -2.5000, 20.0000)
```

L'instruction suivante affiche la position relative à l'univers du modèle Pneu.

```
put member("Séquence").model("Pneu").getWorldTransform().position
--vector(5.0000, -2.5000, -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(5.0000, -2.5000, -10.0000)
```

Voir aussi

transform (propriété), getWorldTransform(), rotation (transformation), scale (transformation)

positionReset

Syntaxe

```
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

Syntaxe

```
member(quelActeurFlash).posterFrame
the posterFrame of member quelActeurFlash
```

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 :

```
on redéfinirLaMiniature quelleAnimationFlash, quelleImage
  member(quelleAnimationFlash).posterFrame = quelleImage
end
```

postNetText

Syntaxe

```
postNetText(url, listeDePropriétés {,chaîneOSduServeur}
  {,chaîneJeuCarServeur})
postNetText(url, textePosté {,chaîneOSduServeur} {,chaîneJeuCarServeur})
```

Description

Commande ; envoie une requête POST à *url*, qui est une adresse URL HTTP, avec *textePosté* comme données.

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.

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.

Le paramètre facultatif *chaîneOSduServeur* prend la valeur par défaut UNIX mais peut recevoir Win ou Mac comme valeur et convertit les retours chariot de l'argument *textePosté* 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 *chaîneJeuCarServeur* 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.

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 L'utilisation de postNetText pour afficher des données dans un domaine différent de celui dans lequel l'animation est lue déclenche une alerte de sécurité lors de la lecture sous Shockwave.

Vous pourrez voir un exemple de postNetText dans une animation en consultant l'animation Forms and Post du dossier Learning/Lingo_Examples, lui-même dans le dossier de Director.

Exemples

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()

Syntaxe

`power(`*`base`*`, `*`exposant`*`)`

Description

Fonction mathématique ; calcule la valeur du nombre spécifié par *base* à la puissance spécifiée par *exposant*.

Exemple

L'instruction suivante donne à la variable *vRésultat* la valeur du cube de 4 :

```
set vRésultat = power(4,3)
```

preferred3DRenderer

Syntaxe

```
the preferred3DRenderer
```

Description

Propriété 3D 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. 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 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.

```
the preferred3dRenderer = #auto
```

Voir aussi

`renderer, getRendererServices(), rendererDeviceList`

preLoad (3D)

Syntaxe

```
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, 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 (commande)

Syntaxe

```
preLoad
preLoad numDimageFinale
preLoad imageInitiale, numDimageFinale
```

Description

Commande ; 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, la commande précharge tous les acteurs utilisés depuis l'image courante jusqu'à la dernière image d'une animation.
- Si un seul argument est fourni (*imageFinale*), précharge tous les acteurs utilisés dans la plage d'images depuis l'image courante jusqu'à l'image *numDimageFinale*, comme spécifié par le numéro ou le nom de l'image.
- Si deux arguments sont fournis (*imageInitiale* et *numDimageFinale*, précharge tous les acteurs utilisés dans la plage d'images depuis l'image *imageInitiale* jusqu'à l'image *numDimageFinale*, comme spécifié par le numéro ou le nom de l'image.

La commande `preLoad` renvoie aussi le numéro de la dernière image qu'il a été possible de charger. Utilisez la fonction `result` pour obtenir cette valeur.

Exemples

L'instruction suivante précharge les acteurs utilisés depuis l'image courante jusqu'à l'image contenant le repère suivant :

```
preLoad marker (1)
```

L'instruction suivante précharge les acteurs utilisés de l'image 10 à l'image 50 :

```
preLoad 10, 50
```

Voir aussi

`preLoadMember`

preLoad (propriété d'acteur)

Syntaxe

```
member(quelActeur).preLoad
the preLoad of member quelActeur
```

Description

Propriété d'acteur ; détermine si l'acteur vidéo numérique spécifié par *quelActeur* 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.

Exemples

L'instruction suivante indique dans la fenêtre Messages si l'animation QuickTime Chaise pivotante peut être préchargée en mémoire :

```
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` :

```
on startMovie
  member("Démo Flash").preload = FALSE
  member("Démo Flash").bufferSize = 65536
end
```

Voir aussi

`bufferSize`, `streamMode`

preLoadBuffer member

Syntaxe

```
member(quelActeur).preLoadBuffer()
preLoadBuffer member quelActeur
```

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`.

Exemple

L'instruction suivante charge l'acteur Jacques Brel en mémoire :

```
member("Jacques Brel").preLoadBuffer()
```

Voir aussi

`preLoadTime`

preLoadEventAbort

Syntaxe

```
the 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).

Cette propriété peut être testée et définie. 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 :

```
set the preLoadEventAbort = TRUE
```

Voir aussi

`preLoad (commande)`, `preLoadMember`

preLoadMember

Syntaxe

```
preLoadMember
member(quelActeur).preLoad()
preLoadMember quelActeur
member(premierActeur).preLoad(dernierActeur)
preLoadMember premierActeur, dernierActeur
```

Description

Commande ; précharge les acteurs et arrête le préchargement lorsque la mémoire est saturée ou que tous les acteurs spécifiés ont été préchargés. La commande `preLoadMember` renvoie le numéro du dernier acteur chargé qu'il a été possible de charger. Utilisez la fonction `result` pour obtenir cette valeur.

En l'absence d'arguments, `preLoadMember` précharge tous les acteurs de l'animation.

Si l'argument *`quelActeur`* est fourni, `preLoadMember` ne précharge que cet acteur. Si *`quelActeur`* est un nombre entier, il n'est fait référence qu'à la première distribution. Si *`quelActeur`* est une chaîne, le premier acteur possédant cette chaîne comme nom est utilisé.

Si les deux arguments *`deLacteur`* et *`aLacteur`* sont fournis, la commande `preLoadMember` précharge tous les acteurs dans la plage spécifiée par les noms ou numéros des acteurs.

Exemples

L'instruction suivante précharge l'acteur 20 de la première distribution (interne) :

```
member(20).preLoad()
```

L'instruction suivante précharge dans la fenêtre Distribution l'acteur Temple et les 10 acteurs qui le suivent :

```
member("Temple").preLoad(member("Temple").number + 10)
```

Pour précharger un acteur particulier d'une distribution spécifique, utilisez la syntaxe suivante :

```
member("Jean", "Membres de la famille").preLoad()
```

preLoadMode

Syntaxe

```
castLib(quelleDistribution).preLoadMode
the preLoadMode of castLib quelleDistribution
```

Description

Propriété d'acteur ; détermine le mode de préchargement de la distribution spécifiée. Elle produit le même effet que la commande de chargement de la distribution de la boîte de dialogue Propriétés de la distribution. Les valeurs possibles sont :

- `0` – Charger la distribution au fur et à mesure.
- `1` – Charger la distribution avant l'image 1.
- `2` – Charger la distribution après l'image 1.

La valeur par défaut pour les acteurs est 0, au fur et à mesure.

Un gestionnaire `on prepareMovie` est généralement un bon endroit pour placer un Lingo déterminant quand les acteurs sont chargés.

Cette propriété peut être testée et définie.

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 :

```
CastLib("Boutons").preLoadMode = 1
```

preLoadMovie

Syntaxe

```
preLoadMovie quelleAnimation
```

Description

Commande ; 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 à la suite d'une commande `go to movie` ou `play movie`.

Pour 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.

Exemple

L'instruction suivante précharge l'animation Introduction, qui est située dans le même dossier que l'animation courante :

```
preLoadMovie "Introduction"
```

preloadNetThing()

Syntaxe

```
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. Remplacez *url* par un nom de fichier Internet valide, tel qu'une animation Director, un graphique ou l'adresse d'un serveur FTP. La valeur renvoyée est un identifiant réseau à utiliser pour suivre le déroulement de l'opération.

Le lecteur Director pour Java ne supporte pas cette commande car le modèle de sécurité de Java ne permet pas d'écrire sur le disque local.

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.

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()`

preLoadRAM

Syntaxe

```
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

Syntaxe

```
member(quelActeur).preLoadTime
the preLoadTime of member quelActeur
sound(numéroDePiste).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.

Exemples

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.

```
on mouseDown
  member("Louis Armstrong").stop()
  ember("Louis Armstrong").preLoadTime = 6
end
```

L'instruction suivante renvoie la valeur `preLoadTime` de l'acteur son lu dans la piste 1.

```
put sound(1).preLoadTime
```

Voir aussi

`preLoadBuffer member`

preMultiply

Syntaxe

`transformation1``.preMultiply(`*`transformation2`*`)`

Description

Commande 3D de transformation ; modifie *`transformation1`* en appliquant au préalable les effets de positionnement, de rotation et de redimensionnement de *`transformation2`*.

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.

Exemple

L'instruction suivante exécute un calcul qui applique la transformation du modèle Mars à la transformation du modèle Pluton :

```
member("Séquence").model("Pluton").transform.preMultiply\
  (member("Séquence").model("Mars").transform)
```

on prepareFrame

Syntaxe

```
on prepareFrame
  instruction(s)
end
```

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 :

```
on prepareFrame me
  sprite(me.spriteNum).locH  = the mouseH
end
```

Voir aussi

`on enterFrame`

on prepareMovie

Syntaxe

```
on prepareMovie
  instruction(s)
end
```

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 initialisés 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 où placer un Lingo 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 :

```
on prepareMovie
  global ScoreActuel
  set ScoreActuel = 0
end
```

Voir aussi

`on enterFrame`, `on startMovie`

preRotate

Syntaxe

```
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, 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 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.

Exemple

L'instruction suivante effectue une rotation de 20° sur chaque axe. La propriété `transform` du modèle étant 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("séquence").model("bip01").transform.preRotate(20, 20, 20)
```

Veuillez noter que la ligne précédente équivaut à la suivante :

```
member("Séquence").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("Séquence").camera[1].transform.position
t.preRotate(vector(100, 0, 0), vector(0, 1, 0), 180)
member("Séquence").camera[1].transform = t
```

Voir aussi

`rotate`

previous

Consultez

`go previous`

preScale()

Syntaxe

```
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.

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 sa rotation autour de la planète aurait lieu deux fois plus souvent 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("Séquence").model("Lune1").transform.duplicate()
t.preScale(2,2,2)
t.rotate(180,0,0)
member("Séquence").model("Lune2").transform = t
```

preTranslate()

Syntaxe

```
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. Notez que `model.transform.pretranslate(x, y, z)` est équivalent 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`.

Exemple

```
t = transform()
t.transform.identity()
t.transform.rotate(0, 90, 0)
t.transform.preTranslate(100, 0, 0)
gbModèle = member("séquence").model("Mars")
gbModèle.transform = t
put gbModèle.transform.position
--vector(0.0000, 0.0000, -100.0000)
```

primitives

Syntaxe

```
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`

print()

Syntaxe

```
sprite(quelleImageObjet).print({"nomDeCible", #limitesDimpression})
```

Description

Commande ; appelle la commande `print` ActionScript correspondante, qui est une commande qui a fait son apparition avec Flash 5. Toutes les images de l'animation Flash libellées `#p` sont imprimées. Si aucune image individuelle n'a été libellée, toute l'animation est imprimée.

Les deux arguments de cette fonction sont facultatifs. L'animation cible est l'animation ou le clip d'animation à imprimer. Si vous ne spécifiez pas de cible (ou si vous spécifiez la cible 0), c'est l'animation Flash principale qui est imprimée.

Les deux valeurs possibles des limites d'impression sont `#bframe` et `#bmax`. Avec `#bmax`, les limites d'impression forment un cadre virtuel de dimensions suffisantes pour contenir toutes les images à imprimer. Avec `#bframe`, les limites d'impression de chaque image sont modifiées en fonction de l'image à imprimer. Si aucune limite d'impression n'est spécifiée, les limites de l'animation cible sont utilisées.

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.

printAsBitmap()

Syntaxe

```
sprite(quelleImageObjet).printAsBitmap({"nomDeCible", #limitesDimpression})
```

Description

Commande ; fonctionne de façon semblable à la commande `print`. Cependant, la commande `printAsBitmap` peut être utilisée pour imprimer des objets contenant des informations alpha.

printFrom

Syntaxe

```
printFrom imageInitiale {,imageFinale} {,réduction}
```

Description

Commande ; imprime tout le contenu de chaque image de la scène, que l'image soit sélectionnée ou non, à partir de l'image spécifiée par *imageInitiale*. Vous pouvez, si vous le voulez, indiquer l'*imageFinale* et la réduction (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.

Exemples

L'instruction suivante imprime le contenu de la scène à l'image 1 :

```
printFrom 1
```

L'instruction suivante imprime le contenu de chaque image de la scène, du repère Intro au repère Conte. La valeur de réduction est de 50 %.

```
printFrom label("Intro"), label("Conte"), 50
```

projection

Syntaxe

```
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, projectionAngle

projectionAngle

Ce terme Lingo est obsolète. Utilisez fieldOfView.

Voir aussi

fieldOfView (3D)

property

Syntaxe

property {*propriété1*}{, *propriété2*} {,*propriété3*} {...}

Description

Mot-clé ; déclare que les propriétés indiquées par *propriété1*, *propriété2*, etc. sont des variables de propriétés.

Déclarez les variables de propriétés au début du script parent ou du script de 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_Examples, lui-même dans le dossier de Director.

Exemples

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
  set ancestor = new(script "Ancien", spriteNum)
end
```

Voir aussi

`me`, `ancestor`, `spriteNum`

proxyServer

Syntaxe

```
proxyServer typeDeServeur, "adresseIP", numDePort
proxyServer()
```

Description

Commande ; définit les valeurs d'un serveur proxy FTP ou HTTP comme suit :

- *typeDeServeur* – `#ftp` ou `#http`
- *adresseIP* – Chaîne contenant l'adresse IP
- *numéroDePort* – Entier correspondant au numéro du port

Si vous utilisez la syntaxe `proxyServer()`, cet élément renvoie les valeurs d'un serveur proxy FTP ou HTTP.

Exemples

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()

Syntaxe

```
ptToHotSpotID(quelleImageObjetQTVR, 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`.

puppet

Syntaxe

```
sprite(quelleImageObjet).puppet
the puppet of sprite quelleImageObjet
```

Description

Propriété d'image-objet ; détermine si la piste d'image-objet spécifiée par *quelleImageObjet* est un esclave contrôlé par Lingo (`TRUE`) ou non (`FALSE`, valeur par défaut).

- Si une piste d'image-objet est asservie, toute modification apportée par Lingo 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 Lingo à l'image-objet ne dure que jusqu'à la fin de l'image-objet courante.

Tant que la tête de lecture reste dans la même image-objet, le fait de donner à la propriété `puppet` de la piste d'image-objet la valeur `FALSE` rend à l'image-objet les propriétés définies dans le scénario.

Le fait d'asservir une piste d'image-objet permet de contrôler depuis Lingo de nombreuses propriétés d'image-objet, telles que `member`, `locH` et `width`, une fois la tête de lecture sortie de l'image-objet.

La définition de la propriété d'image-objet `puppet` équivaut à l'utilisation de la commande `puppetSprite`. Par exemple, les instructions suivantes sont équivalentes : `set the puppet of sprite 1 to TRUE` et `puppetSprite 1, TRUE`.

Cette propriété peut être testée et définie.

Exemples

L'instruction suivante asservit l'image-objet dont le numéro est i + 1 :

```
sprite(i + 1).puppet = TRUE
```

L'instruction suivante enregistre si l'image-objet 5 est asservie en affectant la valeur de la propriété d'image-objet `puppet` à la variable. Si l'image-objet 5 est asservie, *`estEsclave`* prend la valeur `TRUE`. Dans le cas contraire, *`estEsclave`* prend la valeur `FALSE`.

```
estEsclave = sprite(5).puppet
```

Voir aussi

`puppetSprite`

puppetPalette

Syntaxe

`puppetPalette` *`quellePalette`* `{,` *`vitesse`*`} {,`*`nImages`*`}`

Description

Commande ; asservit la piste des palettes et permet à Lingo d'avoir priorité sur les paramètres de la piste des palettes du scénario ainsi que d'affecter des palettes à l'animation.

La commande `puppetPalette` définit comme palette courante l'acteur palette spécifié par l'expression *`quellePalette`*. Si *`quellePalette`* est une chaîne, elle spécifie le nom d'acteur de la palette. Si *`quellePalette`* est un nombre entier, elle spécifie le numéro d'acteur de la palette.

Pour optimiser les résultats, utilisez la commande `puppetPalette` avant de passer à l'image sur laquelle l'effet se produira, afin que Director puisse effectuer une conversion dans 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 *`nImages`* par un nombre entier correspondant au nombre d'images.

Une palette asservie reste active jusqu'au moment de sa désactivation par le biais de la commande `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 et le lecteur Director pour Java utilisent toujours la palette du navigateur web.

Pour obtenir des couleurs fiables pour la création d'une animation destinée au lecteur Director pour Java, utilisez la palette par défaut du système auteur.

Exemples

L'instruction suivante fait d'Arc-en-ciel la palette de l'animation :

```
puppetPalette "Arc-en-ciel"
```

L'instruction suivante fait de Niveaux de gris la palette de l'animation. La transition vers la palette Niveaux de gris s'effectue pendant un laps de temps de 15 et entre les images intitulées Gris et Couleur.

```
puppetPalette "Niveaux de gris", 15, label("Gris") - label("Couleur")
```

puppetSound

Syntaxe

```
puppetSound quellePiste, quelActeur
puppetSound quelActeur
puppetSound member quelActeur
puppetSound 0
puppetSound quellePiste, 0
```

Description

Commande ; asservit la piste audio, lit l'acteur son spécifié par *quelActeur* et permet à Lingo d'avoir priorité sur tous les sons affectés dans les pistes audio du scénario.

Spécifiez une piste audio en remplaçant *quellePiste* par un numéro de piste.

La lecture du son démarre dès que la tête de lecture se met en mouvement ou que la commande `updateStage` est exécutée. L'utilisation de 0 comme argument de numéro dans la distribution empêche la lecture du son. Elle rend également le contrôle de la piste audio au scénario.

Les sons esclaves peuvent s'avérer utiles pour lire un son tandis qu'une animation différente est chargée en mémoire.

Le lecteur Director pour Java supporte les versions suivantes de la commande `puppetSound` :

- `puppetSound` *quellePiste, quelActeur* ou `puppetSound` *quelActeur* – Lit un son.
- `puppetSound 0` ou `puppetSound quellePiste, 0` – Arrête un son.

Exemples

L'instruction suivante lit le son Vent sous le contrôle de Lingo :

```
puppetSound "Vent"
```

L'instruction suivante désactive le son lu dans la piste 2 :

```
puppetSound 2, 0
```

Voir aussi

`sound fadeIn`, `sound fadeOut`, `sound playFile`, `sound stop`

puppetSprite

Syntaxe

```
puppetSprite quelleImageObjet, état
```

Description

Commande ; détermine si la piste d'image-objet spécifiée par *quelleImageObjet* est un esclave contrôlé par Lingo (`TRUE`) ou non (`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 commande `puppetSprite` *quelleImageObjet*, `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 commande `puppetSprite`. Vous pouvez utiliser Lingo pour modifier les propriétés d'image-objet comme suit :

- Si une piste d'image-objet est asservie, toute modification apportée par Lingo 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 Lingo 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 commande `puppetSprite`.

Le fait d'asservir la piste d'image-objet vous permet de contrôler depuis Lingo de nombreuses propriétés d'image-objet, telles que `memberNum`, `locH` et `width`, une fois la tête de lecture sortie de l'image-objet.

Utilisez la commande `puppetSprite` *`quelleImageObjet`*, `FALSE` pour rendre le contrôle au scénario lorsque vous avez fini de contrôler une piste d'image-objet depuis Lingo et pour éviter des résultats imprévisibles qui peuvent se produire lorsque la tête de lecture se trouve dans des images qui ne sont pas 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.

Exemples

L'instruction suivante asservit l'image-objet de la piste 15 :

```
puppetSprite 15, TRUE
```

L'instruction suivante libère de son état d'asservissement l'image-objet de la piste numéro i + 1 :

```
puppetSprite i + 1, FALSE
```

Voir aussi

`backColor`, `bottom`, `constraint`, `cursor (commande)`, `foreColor`, `height`, `ink`, `left`, `lineSize`, `locH`, `locV`, `memberNum`, `puppet`, `right`, `top`, `type (propriété d'image-objet)`, `width`

puppetTempo

Syntaxe

```
puppetTempo imagesParSeconde
```

Description

Commande ; asservit la piste des cadences et règle la cadence sur le nombre d'images spécifié par *`imagesParSeconde`*. Lorsque la piste des cadences est asservie, Lingo peut avoir priorité 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 commande `puppetTempo`, cela ne serait possible qu'avec très peu d'animation et une machine très puissante.

Exemples

L'instruction suivante fixe la cadence de l'animation à 30 images par seconde :

```
puppetTempo 30
```

L'instruction suivante augmente l'ancienne cadence de l'animation de 10 images par seconde :

```
puppetTempo ancienneCadence + 10
```

puppetTransition

Syntaxe

```
puppetTransition member quelActeur
puppetTransition quelleTransition {,temps} {, tailleDeBlocs} {, zoneModifiée}
```

Description

Commande ; exécute la transition spécifiée entre l'image courante et la suivante.

Pour utiliser un Xtra de transition, utilisez `puppetTransition member` suivi du nom ou du numéro de l'acteur.

Pour utiliser une transition Director intégrée, remplacez *quelleTransition* 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 *tailleDeBlocs* 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*
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 *tailleDeBlocs* et *durée*. Par exemple, si *tailleDeBlocs* a une valeur d'un pixel, la transition prend plus de temps, quelle que soit la valeur de durée, car cette opération représente un travail intense l'ordinateur. Pour accélérer les transitions, augmentez la taille des blocs au lieu de réduire la durée.

Remplacez *zoneModifiée* 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 *zoneModifiée* est une zone dans laquelle les images-objets ont changé.

Exemple

L'instruction suivante effectue une transition de type balayage vers la droite. Comme aucune valeur n'est spécifiée pour *zoneModifiée*, la transition se produit sur toute la scène, qui correspond au choix par défaut.

```
puppetTransition 1
```

L'instruction suivante effectue une transition sur toute la scène, de type balayage, d'une durée d'une seconde, la taille de blocs étant de 20 :

```
puppetTransition 2, 4, 20, FALSE
```

purgePriority

Syntaxe

```
member(quelActeur).purgePriority
the purgePriority of member quelActeur
```

Description

Propriété d'acteur ; spécifie la priorité de purge de l'acteur spécifié par *quelActeur*.

Les priorités de purge des acteurs déterminent 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

Le paramètre Normale (valeur par défaut) 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 distribution de l'animation dépasse 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 :

```
member("Arrière-plan").purgePriority = 3
```

put

Syntaxe

`put` *`expression`*

Description

Commande ; évalue l'expression spécifiée par *`expression`* et affiche le résultat dans la fenêtre Messages. Cette commande peut servir d'outil de débogage pour suivre la valeur des variables au cours de la lecture d'une animation.

Le lecteur Director pour Java affiche le message dans la fenêtre de la console Java du navigateur web après la commande `put`. L'accès à la fenêtre de la console dépend du navigateur web.

Exemples

L'instruction suivante affiche l'heure dans la fenêtre Messages :

```
put the time
-- "9:10"
```

L'instruction suivante affiche la valeur affectée à la variable *`Devis`* dans la fenêtre Messages :

```
put Devis
-- "Dubois"
```

Voir aussi

`put...after`, `put...before`, `put...into`

put...after

Syntaxe

`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.

Exemples

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

Syntaxe

`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.

Exemples

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

Syntaxe

`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`.

Exemples

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...=`

qtRegisterAccessKey

Syntaxe

```
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

Syntaxe

```
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`

quad

Syntaxe

```
sprite(quelNuméroDimageObjet).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. Ces points 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'esclave de l'image-objet au moyen de `puppetSprite quelNuméroDimageObjet, FALSE`. La désactivation du quadrilatère d'une image-objet interdit d'autre part de la faire pivoter ou de l'incliner.

Exemples

L'instruction suivante affiche une liste type décrivant une image-objet :

```
put sprite(1).quad
-- [point(153.0000, 127.0000), point(231.0000, 127.0000), \
   point(231.0000, 242.0000), point(153.0000, 242.0000)]
```

Lorsque vous modifiez la propriété d'image-objet `quad`, n'oubliez pas que 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 :

```
-- obtenir le contenu de la propriété courante
listeQuadCourante = sprite(5).quad
-- ajouter 50 pixels aux positions horiz et vert du premier point de la liste
listeQuadCourante[1] = listeQuadCourante[1] + point(50, 50)
-- réinitialiser la propriété réelle sur la nouvelle position calculée
sprite(5).quad = listeQuadCourante
```

Voir aussi

`rotation`, `skew`

quality

Syntaxe

```
sprite(quelleImageObjetFlash).quality
the quality of sprite quelleImageObjetFlash
member(quelleImageObjetFlash).quality
the quality of member quelleImageObjetFlash
```

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`.

```
on beginSprite me
  if the colorDepth <= 8 then
    sprite(1).quality = #low
  end if
end
```

quality (3D)

Syntaxe

```
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`

queue()

Syntaxe

```
sound(numéroDePiste).queue(member(quelActeur))
sound(numéroDePiste).queue([#member: member(quelActeur), {#startTime:
  millisecondes, #endTime: millisecondes, #loopCount: nombreDeBoucles,
  #loopStartTime: millisecondes, #loopEndTime: millisecondes, #preloadTime:
  millisecondes}])
queue(sound(numéroDePiste), member(quelActeur))
queue(objetAudio, [#member: member(quelActeur), {#startTime: millisecondes,
  #endTime: millisecondes, #loopCount: nombreDeBoucles, #loopStartTime:
  millisecondes, #loopEndTime: millisecondes, #preloadTime: millisecondes}])
```

Description

Fonction ; ajoute l'acteur son à la file d'attente de la piste audio *numéroDePiste*.

Dès qu'un son est placé en file d'attente, il peut être lu immédiatement à l'aide de la commande `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 commande `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 commande `queue()`.

Vous pouvez spécifier ces propriétés à l'aide de la commande `queue()` :

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. Pour plus d'informations, consultez `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`.

Ces paramètres peuvent également être passés avec la commande `setPlayList()`.

Vous pourrez voir un exemple de `queue()` dans une animation en consultant l'animation Sound Control du dossier Learning/Lingo_Examples, lui-même dans le dossier de Director.

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 de ses 3 secondes et exécute cinq lectures en boucle successives, de 8 secondes à 8,9 secondes, et s'arrête au point 10 secondes.

```
on lireLaMusique
  sound(2).queue([#member: member("Carillons")])
  sound(2).queue([#member: member("Intro"), #startTime: 3000,\
  #endTime: 10000, #loopCount: 5, #loopStartTime: 8000, #loopEndTime: 8900])
  sound(2).play()
end
```

Voir aussi

`startTime, endTime, loopCount, loopStartTime, loopEndTime, preLoadTime, setPlaylist(), play() (audio)`

queue() (3D)

Syntaxe

```
member(quelActeur).model(quelModèle).bonesPlayer.queue\
  (nomDuMouvement {, enBoucle, posInitiale, posFinale, échelle, décalage})
member(quelActeur).model(quelModèle).keyframePlayer.\
  queue(nomDuMouvement {, enBoucle, posInitiale, posFinale, échelle,
  décalage})
```

Description

Commande 3D de modificateur `keyframePlayer` et `bonesPlayer` ; ajoute le mouvement spécifié par *`nomDeMouvement`* 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.

Les paramètres facultatifs de cette commande sont les suivants :

`enBoucle` spécifie si le mouvement est lu une seule fois (`FALSE`) ou continuellement (`TRUE`).

`posInitiale` 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 *`posFinale`*. Lorsque *`enBoucle`* a pour valeur `TRUE`, la première itération de la boucle démarre à *`décalage`* et se termine à *`posFinale`*. Toutes les répétitions suivantes démarrent à *`posInitiale`* et se terminent à *`posFinale`*.

`posFinale` 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 *`posFinale`*. Lorsque *`enBoucle`* a pour valeur `TRUE`, la première itération de la boucle démarre à *`décalage`* et se termine à *`posFinale`*. Toutes les répétitions suivantes démarrent à *`posInitiale`* et se terminent à *`posFinale`*. Donnez à *`posFinale`* la valeur `-1` si le mouvement doit être lu jusqu'à la fin.

`échelle` est multiplié par la propriété `playRate` du modificateur `keyframePlayer` ou `bonesPlayer` du modèle, pour déterminer la cadence réelle de la lecture du mouvement.

`décalage` 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 *`posFinale`*. Lorsque *`enBoucle`* a pour valeur `TRUE`, la première itération de la boucle démarre à *`décalage`* et se termine à *`posFinale`*. Toutes les répétitions suivantes démarrent à *`posInitiale`* et se terminent à *`posFinale`*.

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`

quickTimeVersion()

Syntaxe

```
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.

Exemple

L'instruction suivante utilise `quickTimeVersion()` pour afficher la version de QuickTime courante dans la fenêtre Messages :

```
put quickTimeVersion()
```

quit

Syntaxe

```
quit
```

Description

Commande ; permet de quitter Director ou une projection et de passer au bureau de Windows ou au Finder du Macintosh.

Exemple

L'instruction suivante indique à l'ordinateur de quitter vers le bureau de Windows ou le Finder du Macintosh lorsque l'utilisateur appuie sur Ctrl+Q (Windows) ou sur Cmd+Q (Macintosh) :

```
if the key = "q" and the commandDown then quit
```

Voir aussi

`restart`, `shutDown`

QUOTE

Syntaxe

```
QUOTE
```

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 :

```
put "Comment épelez-vous" && QUOTE & "Macromedia" & QUOTE && "?"
```

Le résultat est un jeu de guillemets placés autour du mot *Macromedia* :

```
Comment épelez-vous "Macromedia" ?
```

radius

Syntaxe

```
référenceDobjetDeRessDeModèle.radius
member(quelActeur).modelResource(quelleRessDeMod).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
```

ramNeeded()

Syntaxe

```
ramNeeded (premièreImage, dernièreImage)
```

Description

Fonction ; 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.

Exemples

L'instruction suivante affecte à la variable *tailleDimage* le nombre d'octets requis pour afficher les images 100 à 125 de l'animation :

```
put ramNeeded (100, 125) into tailleDimage
```

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 :

```
if ramNeeded (100, 125) > the freeBytes then play frame "8 bits"
```

Voir aussi

`freeBytes()`, `size`

random()

Syntaxe

`random(`*expressionEntière*`)`

Description

Fonction ; renvoie un nombre entier aléatoire compris entre 1 et la valeur spécifiée par *expressionEntière*. 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`.

Exemples

L'instruction suivante affecte des valeurs aléatoires à la variable *dés* :

```
set dés = random(6) + random(6)
```

L'instruction suivante modifie de façon aléatoire la couleur de premier plan de l'image-objet 10 :

```
sprite(10).forecolor = random(256) - 1
```

Le gestionnaire suivant choisit de manière aléatoire lequel des deux segments de l'animation va être lu :

```
on sélectionDeScène
  if random(2) = 2 then
    play frame "11a"
  else
    play frame "11-b"
  end if
end
```

Les instructions suivantes produisent des résultats dans une plage spécifique.

L'instruction suivante produit un multiple aléatoire de cinq compris entre 5 et 100 :

```
score = 5 * random(20)
```

L'instruction suivante produit un multiple aléatoire de cinq compris entre 0 et 100 :

```
score = 5 * (random(21) - 1)
```

L'instruction suivante génère des nombres entiers compris entre -10 et +10 :

```
dirH = random(21) - 11
```

L'instruction suivante produit une valeur décimale aléatoire à deux chiffres après la virgule :

```
the floatPrecision = 2
lesCentimes = random(100)/100.0 - .01
```

randomSeed

Syntaxe

```
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()`, `ticks`

randomVector

Syntaxe

```
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.

Exemples

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()

Syntaxe

```
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`.

Exemples

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`

realPlayerNativeAudio()

Syntaxe

```
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 lecteur Shockwave 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 (RealMedia)` et d'utiliser les méthodes et propriétés audio standard de Lingo en vue du traitement du train 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. Pour plus d'informations sur l'utilisation de RealAudio, consultez Utilisation de contenu RealMedia dans Director, dans le mode d'emploi de Director. Pour plus d'informations sur l'audio dans Director, consultez les entrées correspondantes dans le dictionnaire Lingo.

Exemples

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.

```
put realPlayerNativeAudio()
-- 0
```

Le code suivant donne à la fonction `realPlayerNativeAudio()` la valeur `TRUE`, ce qui signifie que l'audio du train RealMedia sera traitée par RealPlayer et que le contrôle de Lingo sur la piste audio ne sera pas pris en compte.

```
realPlayerNativeAudio(TRUE)
```

Voir aussi

`soundChannel (RealMedia)`, `audio (RealMedia)`

realPlayerPromptToInstall()

Syntaxe

```
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()` 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 pour déterminer si RealPlayer 8 est installé. Sous Windows, les numéros de version 6.0.8.132 ou supérieurs indiquent que RealPlayer 8 est installé. Sur le 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.

Exemples

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.

```
put 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.

```
realPlayerPromptToInstall(FALSE)
```

realPlayerVersion()

Syntaxe

```
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 supérieurs indiquent que RealPlayer 8 est installé. Sur le 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, si vous ne souhaitez pas utiliser celui qui est fourni par la fonction `realPlayerPromptToInstall()`.

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 le 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()`.

Exemple

Le code suivant indique que le numéro de version de RealPlayer installé sur le système est 6.0.9.357.

```
put realPlayerVersion()
-- "6.0.9.357"
```

recordFont

Syntaxe

```
recordFont(quelActeur, police {[,style]} {,[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.

Exemples

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)

Syntaxe

`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.

Veuillez noter que 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 dans Lingo.

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()

Syntaxe

```
rect(gauche, haut, droite, bas)
rect(point1, point2)
```

Description

Fonction et type de données ; définit un rectangle.

Le format `rect(gauche, haut, droite, bas)` définit un rectangle dont les côtés sont spécifiés par *gauche*, *haut*, *droite* et *bas*. Les valeurs *gauche* et *droite* spécifient le nombre de pixels à partir du bord gauche de la scène. Les valeurs *haut* et *bas* spécifient le nombre de pixels à partir du bord supérieur de la scène.

Le format `rect(point1, point2)` définit un rectangle qui renferme les points spécifiés par *point1* et *point2*.

Vous pouvez faire référence aux composants de rectangles avec les syntaxes de liste ou de propriétés. Par exemple, les deux expressions suivantes sont équivalentes :

```
largeurCible = rectCible.right - rectCible.left
largeurCible = rectCible[3] - rectCible[1]
```

Vous pouvez effectuer des opérations arithmétiques sur les rectangles. Si vous ajoutez une seule valeur à un rectangle, Lingo l'ajoute à chaque élément du rectangle.

Vous pourrez voir un exemple de `rect()` dans une animation en consultant l'animation Imaging du dossier Learning/Lingo_Examples, lui-même dans le dossier de Director.

Exemples

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 :

```
set nouvelleZone = rect(100, 150, 300, 400)
```

L'instruction suivante définit la variable *nouvelleZone* comme un rectangle défini par les points `premierPoint` et `deuxièmePoint`. Les coordonnées de `premierPoint` sont (100, 150) ; celles de `deuxièmePoint` sont (300, 400). Notez que cette instruction crée le même rectangle que celui créé dans l'exemple précédent :

```
put rect(premierPoint, deuxièmePoint)
```

Les instructions suivantes ajoutent et soustraient des valeurs aux rectangles :

```
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)
```

L'instruction suivante ajoute 80 à chaque coordonnée d'un rectangle :

```
put rect(60, 40, 120, 200) + 80
--rect (140, 120, 200, 280)
```

L'instruction suivante divise chaque coordonnée d'un rectangle par 3 :

```
put rect(60, 40, 120, 200) / 3
-- rect(20, 13, 40, 66)
```

Voir aussi

`point()`, `quad`

rect (image)

Syntaxe

```
objetImage.rect
```

Description

Propriété en lecture seule ; renvoie un rectangle décrivant la taille de l'objet image concerné. Les coordonnées 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.

Exemples

L'instruction suivante affiche le rectangle de l'acteur Lever de soleil de 300 x 400 pixels dans la fenêtre Messages :

```
put member("Lever de soleil").image.rect
--rect (0, 0, 300, 400)
```

L'instruction Lingo suivante examine les 50 premiers acteurs et affiche le rectangle et le nom de chaque acteur bitmap :

```
on afficherTousLesRects
  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
```

Voir aussi

`height`, `width`

rect (acteur)

Syntaxe

```
member(quelActeur).rect
the rect of member quelActeur
```

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.

Pour un bitmap, la propriété d'acteur `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 d'acteur `rect` est un rectangle dont le coin supérieur gauche est à (0,0).

Le lecteur Director pour Java ne peut pas définir la propriété d'acteur `rect`.

Cette propriété peut être testée. Elle ne peut être définie que pour les acteurs champ.

Exemples

L'instruction suivante affiche les coordonnées de l'acteur bitmap 20 :

```
put member(20).rect
```

L'instruction suivante définit les coordonnées de l'acteur bitmap Bandeau :

```
member("Bandeau").rect = rect(100, 150, 300, 400)
```

Voir aussi

`rect()`, `rect (image-objet)`

rect (image-objet)

Syntaxe

```
sprite quelleImageObjet.rect
the rect of sprite quelleImageObjet
```

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.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante affiche les coordonnées de l'image-objet bitmap 20 :

```
put sprite(20).rect
```

rect (fenêtre)

Syntaxe

```
window quelleFenêtre.rect
the rect of window quelleFenêtre
```

Description

Propriété de fenêtre ; spécifie les coordonnées gauche, supérieure, droite et inférieure, renvoyées sous la forme d'un rectangle, de la fenêtre spécifiée par *quelleFenêtre*.

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.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante affiche les coordonnées de la fenêtre Tableau_de_commande :

```
put window("Tableau_de_commande").rect
```

Voir aussi

`drawRect`, `sourceRect`

ref

Syntaxe

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.

Exemples

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, tel que :

```
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

Syntaxe

```
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

Syntaxe

```
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 de degré de brillant du matériau par défaut à 50 % pour l'acteur Séquence :

```
member("Séquence").reflectivity = 50
```

region

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).\
  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

registerForEvent()

Syntaxe

```
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().

Le paramètre *nomDeGestionnaire* est le nom du gestionnaire qui sera appelé ; ce gestionnaire se trouve dans l'objet de script indiqué par *objetScript*.

Si 0 est spécifié pour *objetScript*, c'est le premier gestionnaire d'événement portant le nom donné qui est appelé.

Le paramètre *nomDévénement* peut être n'importe lequel des événements Lingo prédéfinis suivants ou tout événement personnalisé :

- #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.

  Le gestionnaire que vous spécifiez 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 un 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.

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()`.

Exemples

L'instruction suivante enregistre le gestionnaire d'événement `interrogerUtil` détecté dans un script d'animation pour un appel à deux reprises, à un intervalle de cinq secondes :

```
member("Séquence").registerForEvent(#timeMS, #interrogerUtil, 0, \
  5000, 5000, 2)
```

L'instruction suivante enregistre le gestionnaire d'événement `interrogerUtil` détecté dans un script d'animation pour un appel à chaque fois qu'une collision a lieu au sein de l'acteur Séquence :

```
member("Séquence").registerForEvent(#collideAny, #interrogerUtil, 0)
```

L'instruction suivante déclare que le gestionnaire `on interrogerUtil` 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 Séquence :

```
member("Séquence").registerForEvent(#collideWith, #interrogerUtil, me, \
  member("Séquence").model("Pluton"))
```

Voir aussi

`setCollisionCallback()`, `registerScript()`, `play() (3D)`, `playNext() (3D)`, `autoblend`, `blendTime`, `sendEvent`, `unregisterAllEvents`

registerScript()

Syntaxe

```
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()`.

Le paramètre *nomDeGestionnaire* est le nom du gestionnaire qui sera appelé ; ce gestionnaire se trouve dans l'objet de script indiqué par *objetScript*.

Si 0 est spécifié pour *objetScript*, c'est le premier gestionnaire d'événement portant le nom donné qui est appelé.

Le paramètre *nomDévénement* peut être n'importe lequel des événements Lingo prédéfinis suivants ou tout événement personnalisé :

- `#collideAny` est un événement de collision généré lorsque deux entités du système entrent en collision et que le modificateur `#collision` est associé à ces deux entités.
- `#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.

  Le gestionnaire que vous spécifiez 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 un 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.

Exemples

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("Séquence").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("Séquence").model("Lecteur").registerScript(#collideWith, \
  #réponseCollision, me)
```

Voir aussi

registerForEvent(), sendEvent, setCollisionCallback(), collisionData

regPoint

Syntaxe

```
member(quelActeur).regPoint
the regPoint of member quelActeur
```

Description

Propriété d'acteur ; spécifie le point d'alignement d'un acteur. 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 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`.

Cette propriété peut être testée et définie.

Exemples

L'instruction suivante affiche le point d'alignement de l'acteur bitmap Bureau dans la fenêtre Messages :

```
put member("Bureau").regPoint
```

L'instruction suivante change le point d'alignement de l'acteur bitmap Bureau en fonction des valeurs de la liste :

```
member("Bureau").regPoint = point(300, 400)
```

Voir aussi

`centerRegPoint`, `mask`

regPoint (3D)

Syntaxe

```
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

Syntaxe

```
acteurVecteur.regPointVertex
the regPointVertex of acteurVecteur
```

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 :

```
member("quelconque").regPointVertex=3
```

Voir aussi

`centerRegPoint`, `regPoint`

relative

Consultez

`@ (chemin d'accès)`

removeBackdrop

Syntaxe

```
member(quelActeur).camera(quelleCaméra).removeBackdrop(index)
```

Description

Commande 3D ; supprime le fond de la position spécifiée par *index* de la liste des fonds de la caméra.

Exemple

L'instruction suivante retire le troisième fond de la liste des fonds de la caméra 1 de l'acteur Séquence. Le fond disparaîtra de la scène si des images-objets utilisent actuellement cette caméra.

```
member("Séquence").camera[1].removeBackdrop(3)
```

removeFromWorld

Syntaxe

```
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.

Exemple

La commande suivante retire le modèle gbCyl de l'univers 3D de l'acteur Séquence.

```
member("Séquence").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.

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

Syntaxe

```
member(quelActeur).model(quelModèle).removeModifier.\
  (#quelModificateur)
```

Description

Commande 3D ; supprime le modificateur identifié par *#quelModificateur* 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.

Exemple

L'instruction suivante retire le modificateur `#toon` du modèle Boîte.

```
member("formes").model("Boîte").removeModifier(#toon)
```

Voir aussi

`addModifier`, `modifier`, `modifier[]`, `modifiers`

removeOverlay

Syntaxe

```
member(quelActeur).camera(quelleCaméra).removeOverlay(index)
```

Description

Commande 3D ; retire le recouvrement de la position spécifiée par *index* de la liste des recouvrements de la caméra.

Exemple

L'instruction suivante retire le troisième 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`

renderer

Syntaxe

```
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 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

Syntaxe

```
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

Syntaxe

```
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 que vous choisissez détermine le niveau de fidélité des couleurs (y compris la précision de la couche alpha facultative) et par conséquent 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

Syntaxe

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
```

repeat while

Syntaxe

```
repeat while conditionTest
     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.

Lorsque Lingo est dans une boucle, le lecteur Director pour Java ne détecte pas les mouvements de la souris et ne met pas à jour les propriétés qui indiquent la position de la souris, ni l'état d'enfoncement des boutons.

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 lui fait compter jusqu'à 60 battements :

```
on countTime
  startTimer
  repeat while the timer < 60
  -- en attente
  end repeat
end countTime
```

Voir aussi

exit, exit repeat, repeat with, keyPressed()

repeat with

Syntaxe

```
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).

Lorsque Lingo est dans une boucle, le lecteur Director pour Java ne détecte pas les mouvements de la souris et ne met pas à jour les propriétés qui indiquent la position de la souris, ni l'état d'enfoncement des boutons.

Exemple

Le gestionnaire suivant transforme les images-objets 1 à 30 en esclaves :

```
on esclaves
  repeat with channel = 1 to 30
    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

Syntaxe

`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).

Lorsque Lingo est dans une boucle, le lecteur Director pour Java ne détecte pas les mouvements de la souris et ne met pas à jour les propriétés qui indiquent la position de la souris, ni l'état d'enfoncement des boutons.

Exemple

Le gestionnaire suivant contient une boucle qui décompte de 20 à 15 :

```
on décompte
  repeat with i = 20 down to 15
    sprite(6).memberNum = 10 + i
    updateStage
  end repeat
end
```

repeat with...in list

Syntaxe

```
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).

Lorsque Lingo est dans une boucle, le lecteur Director pour Java ne détecte pas les mouvements de la souris et ne met pas à jour les propriétés qui indiquent la position de la souris, ni l'état d'enfoncement des boutons.

Exemple

L'instruction suivante affiche quatre valeurs dans la fenêtre Messages :

```
repeat with i in [1, 2, 3, 4]
  put i
end repeat
```

resetWorld

Syntaxe

```
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 être `0` (déchargé), `4` (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.

Exemple

L'instruction suivante redonne aux propriétés de l'acteur Séquence les valeurs utilisées lorsque l'acteur a été mis en mémoire pour la première fois.

```
member("Séquence").resetWorld()
```

Voir aussi

`revertToWorldDefaults`

on resizeWindow

Syntaxe

```
on resizeWindow
  instruction(s)
end
```

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 Lingo se rapportant aux dimensions de la fenêtre, comme le positionnement des images-objets et le recadrement des animations de type vidéo numérique.

Exemple

Le gestionnaire suivant déplace l'image-objet 3 vers les coordonnées stockées dans la variable *`emplacementCentral`* lorsque la fenêtre lue par l'animation est redimensionnée :

```
on resizeWindow emplacementCentral
  sprite(3).loc = emplacementCentral
end
```

Voir aussi

`drawRect`, `sourceRect`

resolution

Syntaxe

`member(`*`quelActeur`*`).modelResource(`*`quelleRessDeMod`*`).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
```

resolve

Syntaxe

```
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`

resolveA

Syntaxe

```
collisionData.resolveA(bRésolution)
```

Description

Méthode 3D de collision ; annule le comportement de collision défini par la propriété `collision.resolve` pour `collisionData.modelA`. Si *`bRésolution`* est `TRUE`, la collision est résolue pour modelA et, si *`bRésolution`* est `FALSE`, la collision n'est pas résolue pour modelA. N'appelez cette fonction que si vous souhaitez remplacer le comportement défini pour modelA à l'aide de `collision.resolve`.

Voir aussi

`collisionData`, `registerScript()`, `resolve`, `modelA`, `setCollisionCallback()`

resolveB

Syntaxe

```
collisionData.resolveB(bRésolution)
```

Description

Méthode 3D de collision ; annule le comportement de collision défini par la propriété `collision.resolve` pour `collisionData.modelB`. Si *`bRésolution`* est `TRUE`, la collision est résolue pour modelB et, si *`bRésolution`* est `FALSE`, la collision n'est pas résolue pour modelB. N'appelez cette fonction que si vous souhaitez remplacer le comportement défini pour modelB à l'aide de `collision.resolve`.

Voir aussi

`collisionData`, `resolve`, `registerScript()`, `modelB`, `setCollisionCallback()`

resource

Syntaxe

```
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
```

restart

Syntaxe

```
restart
```

Description

Commande ; ferme toutes les applications ouvertes et redémarre l'ordinateur.

Exemple

L'instruction suivante redémarre l'ordinateur lorsque l'utilisateur appuie sur la combinaison de touches Cmd+R (Macintosh) ou Ctrl+R (Windows) :

```
if the key = "r" and the commandDown then restart
```

Voir aussi

`quit`, `shutDown`

result

Syntaxe

```
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.

Cette fonction n'a aucun effet dans le lecteur Director pour Java.

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 maVariable = fonction()`, où *`fonction()`* est le nom d'une fonction spécifique.

Exemples

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()
```

Notez que `set dés = lancerLesDés` n'appelle pas le gestionnaire car, en l'absence de parenthèses après `lancerLesDés`, `lancerLesDés` est considéré comme une référence de variable.

Voir aussi

`return (mot-clé)`

resume sprite

Syntaxe

```
sprite(quelNuméroDimageObjetGIF ).resume()
resume(sprite quelNuméroDimageObjetGIF)
```

Description

Commande 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.

Voir aussi

`pause sprite`, `rewind sprite`

RETURN (constante)

Syntaxe

```
RETURN
```

Description

Constante ; représente un retour de chariot.

Exemples

L'instruction suivante reprend la lecture d'une animation en pause lorsque l'utilisateur appuie sur le retour de chariot :

```
if (the key = RETURN) then go to the 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 :

```
alert "Dernière ligne du fichier." & RETURN & "Cliquer 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)
```

return (mot-clé)

Syntaxe

```
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_Examples, 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 :

```
set score to 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)`

revertToWorldDefaults

Syntaxe

```
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 être `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.

Exemple

L'instruction suivante redonne aux propriétés de l'acteur Séquence les valeurs enregistrées lorsque l'acteur a été créé.

```
member("Séquence").revertToWorldDefaults()
```

Voir aussi

`resetWorld`

rewind()

Syntaxe

```
sound(numéroDePiste).rewind()
rewind(sound(numéroDePiste))
```

Description

Fonction ; interrompt la lecture du son sur la piste audio *numéroDePiste* 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`.

Exemple

L'instruction suivante entraîne une nouvelle lecture de l'acteur son de la piste audio 1, depuis la position de départ.

```
sound(1).rewind()
```

Voir aussi

```
pause() (lecture audio), play() (audio), playNext(), queue(), stop() (audio)
```

rewind sprite

Syntaxe

```
sprite(quelNuméroDimageObjetGIFanimOuFlash).rewind()
rewind sprite quelNuméroDimageObjetGIFanimOuFlash
```

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.

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.

```
property spriteNum

on exitFrame
  if sprite(spriteNum).playing then
    go the frame
  else
    sprite(spriteNum).rewind()
    updateStage
  end if
end
```

rgb()

Syntaxe

```
rgb(valeurRouge, valeurVerte, valeurBleue)
```

Description

Fonction et type de données ; définit une couleur en fonction de la valeur spécifiée pour rouge, vert et bleu. La plage de valeurs de chacune des trois couleurs est 0 - 255.

Exemple

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)
```

Voir aussi

```
color()
```

right

Syntaxe

```
sprite(quelleImageObjet).right
the right of sprite quelleImageObjet
```

Description

Propriété d'image-objet ; indique la distance, en pixels, séparant le bord droit de l'image-objet spécifiée du bord gauche de la scène.

Lorsqu'une animation est lue en tant qu'applet, la valeur de cette propriété est déterminée par rapport au coin supérieur gauche de l'applet.

Les coordonnées des images-objets sont exprimées par rapport au coin supérieur gauche de la scène.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante appelle le gestionnaire *débordementAdroite* lorsque le bord droit de l'image-objet 3 dépasse le bord droit de la scène :

```
if sprite(3).right > (the stageRight - the stageLeft) then débordementAdroite
```

Voir aussi

bottom, height, left, locH, locV, top, width

right (3D)

Syntaxe

```
member(quelActeur).modelResource
(quelleRessDeMod).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

Syntaxe

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

on rightMouseDown (gestionnaire d'événement)

Syntaxe

```
on rightMouseDown
  instruction(s)
end
```

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 le 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 le Macintosh.

Exemple

Le gestionnaire suivant ouvre la fenêtre Aide lorsque l'utilisateur appuie sur le bouton droit de la souris sous Windows :

```
on rightMouseDown
  window("Aide").open()
end
```

rightMouseDown (propriété système)

Syntaxe

the rightMouseDown

Description

Propriété système ; 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).

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 :

```
if the rightMouseDown then puppetSound 2, "Désolé"
```

Voir aussi

`emulateMultiButtonMouse`

on rightMouseUp (gestionnaire d'événement)

Syntaxe

```
on rightMouseUp
  instruction(s)
end
```

Description

Message système et gestionnaire d'événements ; sous Windows, spécifie les instructions exécutées lorsque l'utilisateur relâche le bouton droit de la souris. Sur le 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 le Macintosh.

Exemple

Le gestionnaire suivant ouvre la fenêtre Aide lorsque l'utilisateur relâche le bouton droit de la souris sous Windows :

```
on rightMouseUp
  window("Aide").open()
end
```

rightMouseUp (propriété système)

Syntaxe

```
the rightMouseUp
```

Description

Propriété système ; indique si le bouton droit de la souris (Windows) ou le bouton de la souris et la touche Ctrl (Macintosh) sont relâchés (`TRUE`) ou enfoncés (`FALSE`).

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 :

```
if the rightMouseUp then puppetSound 2, "Cliquez"
```

Voir aussi

`emulateMultiButtonMouse`

rollOver()

Syntaxe

```
rollOver(quelleImageObjet)
the rollOver
```

Description

Fonction ; indique si le curseur se trouve sur le rectangle délimitant l'image-objet spécifiée par *quelleImageObjet* (TRUE) ou non (FALSE).

Cette fonction peut être exprimée avec deux formats de syntaxe :

- Lorsque rollOver n'est pas précédé de the, des parenthèses sont nécessaires.
- Lorsque rollOver est précédé de the, les parenthèses ne sont pas nécessaires.

La fonction rollOver est généralement utilisée avec les scripts d'images et est utile pour créer des gestionnaires qui exécutent une action lorsque l'utilisateur place le curseur sur une image-objet spécifique. Elle peut aussi simuler des pistes d'images-objets supplémentaires en divisant la scène en zones qui envoient la tête de lecture vers une image différente partageant la zone entre les pistes d'images-objets disponibles.

Si l'utilisateur continue à déplacer la souris, la valeur de rollOver peut changer pendant que Lingo exécute un gestionnaire. 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 curseur se trouve sur l'emplacement d'une image-objet qui n'apparaît plus dans la section courante du scénario, 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.

Exemples

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 :

```
if rollover(6) then 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.

```
on exitFrame
  set imageObjetCourante = the rollover
  case imageObjetCourante of
    1: go to frame "Gauche"
    2: go to frame "Milieu"
    3: go to frame "Droite"
  end case
end exitFrame
```

Voir aussi

mouseMember

romanLingo

Syntaxe

```
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 octets. 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
```

rootLock

Syntaxe

```
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

Syntaxe

```
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("univers")`, 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")
```

rotate

Syntaxe

```
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 d'un objet de transformation d'un nœud référencé 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. Cet axe est défini dans l'espace par la valeur de *position* et par la valeur d'*axe*, représentant un axe passant par la position spécifiée dans l'espace. Le degré de rotation autour de cet axe est spécifié par *angle*.

Le paramètre facultatif *parRapportA* détermine les axes du système de coordonnées utilisés pour appliquer les modifications de rotation. Le paramètre *parRapportA* peut avoir les valeurs suivantes :

- `#self` applique les incréments en fonction du système de coordonnées local du nœud (les axes x, y et 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.

Exemples

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("Séquence").model("Lune").rotate(0,0,15)
member("Séquence").model("Lune").rotate(vector(0, 0, 5),
  member("Séquence").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("Séquence 3D").model("Bâton").worldPosition
member("Séquence 3D").model("Balle").rotate(posBâton, vector(0,0,1), \
  5, #world)
```

Voir aussi

`pointAt`, `preRotate`, `rotation (transformation)`, `rotation (matériau de gravure)`, `rotation (fond et recouvrement)`, `preScale()`, `transform (propriété)`

rotation

Syntaxe

```
the rotation of member quelActeurQuickTime
member(quelActeurQuickTime).rotation
sprite(quelleImageObjet).rotation
the rotation of sprite quelleImageObjet
```

Description

Propriété d'acteur et 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). 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 Lingo pour exécuter une rotation continue.

Cette propriété peut être testée et définie. La valeur par défaut est 0.

Exemples

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° :

```
property spriteNum

on prepareFrame me
  sprite(spriteNum).rotation = integer(sprite(spriteNum).rotation + 2) mod 360
end
```

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.

```
on exitFrame
  if sprite(5).rotation < 360 then
    sprite(5).rotation = sprite(5).rotation + 16
    go the frame
  end if
end
```

Le gestionnaire suivant accepte une référence d'image-objet comme paramètre et fait pivoter une image-objet animation Flash de 360° par incréments de 10° :

```
on fairePivoterLanimation quelleImageObjet
  repeat with i = 1 to 36
    sprite(quelleImageObjet).rotation = i * 10
    updatestage
  end repeat
end
```

Voir aussi

`flipH`, `flipV`, `obeyScoreRotation`, `originMode`

rotation (fond et recouvrement)

Syntaxe

```
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)

Syntaxe

```
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("Cyl3").shader.rotation = \
  member("séquence").model("Cyl3").shader.rotation + 1
```

Voir aussi

`transform (propriété)`

rotation (transformation)

Syntaxe

```
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 à l'univers d'un objet à l'aide de *`transformation`*`.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.

Exemples

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, 0.0000, 45.0000)
member("systèmeSolaire").model("Lune").rotate(15,15,15)
put member("systèmeSolaire").model("Lune").getWorldTransform().rotation
--vector(51.3810, 16.5191, 65.8771)
```

Voir aussi

`getWorldTransform()`, `preRotate`, `rotate`, `transform (propriété)`, `position (transformation)`, `scale (transformation)`

rotationReset

Syntaxe

```
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

Syntaxe

```
member(quelActeur).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 :

```
put member("CV").RTF
```

Voir aussi

HTML, importFileInto

runMode

Syntaxe

```
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 ou autre environnement de programmation tel que LiveConnect ou ActiveX.
- `Java Applet` – L'animation est lue en tant qu'applet Java.

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.

Exemple

L'instruction suivante détermine si des paramètres externes sont disponibles et, le cas échéant, les obtient :

```
if the runMode contains "Plugin" then
  -- décoder le paramètre embed
  if externalParamName(swURL) = swURL then
    put externalParamValue(swURL) into maVariable
  end if
end if
```

Voir aussi

environment, platform

on runPropertyDialog

Syntaxe

```
on runPropertyDialog me, listeDinitialisationCourante
  instruction(s)
end
```

Description

Message système et gestionnaire d'événements ; contient un Lingo 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 *`listeDinitialisationCourante`*. 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 :

```
property masse
property constanteGravité
on runPropertyDialog me, listeDinitialisationCourante
  -- forcer la masse à 10
  setaProp listeDinitialisationCourante, #masse, 10
  -- forcer la constanteGravité à 9,8
  setaProp listeDinitialisationCourante, #constanteGravité , 9,8
  return listeDinitialisationCourante
end
```

Voir aussi

`on getBehaviorDescription`, `on getPropertyDescriptionList`

safePlayer

Syntaxe

```
the safePlayer
```

Description

Propriété système ; contrôle si les fonctions de sécurité de Director sont activées ou non.

Dans une animation Shockwave, cette propriété peut être testée, mais pas définie. Elle est toujours `TRUE` dans Shockwave.

Dans l'environnement auteur comme dans les projections, la valeur par défaut est `FALSE`. Cette propriété peut être testée, 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 commande `pasteClipBoardInto` génère une boîte de dialogue d'avertissement et permet à l'utilisateur d'annuler l'opération.
- La manipulation des fichiers de ressources Macintosh avec les commandes obsolètes `openResFile` ou `closeResFile` est désactivée.
- L'enregistrement d'une animation ou d'une distribution avec Lingo est désactivé.
- L'impression avec la commande `printFrom` est désactivée.
- L'ouverture d'une application avec la commande `open` est désactivée.

- La possibilité d'arrêter une application ou d'éteindre l'ordinateur de l'utilisateur avec les commandes `restart` ou `shutDown` est désactivée.
- L'envoi de chaînes à l'interface MCI de Windows avec `mci` est désactivé.
- 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.

sampleCount

Syntaxe

```
sound(numéroDePiste).sampleCount
the sampleCount of sound(numéroDePiste)
```

Description

Propriété en lecture seule ; nombre de sons échantillonnés du son de la piste audio *numéroDePiste*. 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 :

```
put "L'acteur son" && sound(1).member.name && "contient" && \
sound(1).sampleCount && "échantillons."
```

Voir aussi

`channelCount`, `sampleRate`, `sampleSize`

sampleRate

Syntaxe

```
member(quelActeur).sampleRate
the sampleRate of member quelActeur
sound(numéroDePiste).sampleRate
```

Description

Propriété d'acteur ; 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. 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`. 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.

Cette propriété peut être testée, mais pas définie. Des valeurs typiques 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 commande `stop()`. Le prochain son placé en file d'attente ou lu déterminera ensuite les nouveaux paramètres.

Exemples

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 :

```
member("Qualité audio").text = string(member("Paul Robin").sampleRate)"
```

L'instruction suivante affiche le taux d'échantillonnage du son lu dans la piste audio 1 dans la fenêtre Messages :

```
put sound(1).sampleRate
```

sampleSize

Syntaxe

```
member(quelActeur).sampleSize
the sampleSize of member quelActeur
sound(numéroDePiste).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.

Exemples

L'instruction suivante détermine la taille d'échantillonnage de l'acteur son Voix-off et affecte cette valeur à la variable *tailleDuSon* :

```
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 :

```
put sound(1).sampleSize
```

save castLib

Syntaxe

```
castLib(quelleDistrib).save()
save castLib quelleDistrib {,nomDuChemin&nomDeNouveauFichier}
```

Description

Commande ; enregistre les modifications apportées à la distribution dans son fichier d'origine ou dans un nouveau fichier si le paramètre facultatif *nomDuChemin:nomDeNouveauFichier* est présent. Si aucun nom de fichier n'est fourni, la distribution d'origine doit être liée. 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 supporte pas les adresses URL comme références de fichier.

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)

on savedLocal

Syntaxe

```
on savedLocal
  instruction(s)
end
```

Description

Message système et gestionnaire d'événements. Cette propriété est fournie pour permettre des améliorations dans de futures versions de Shockwave.

Voir aussi

allowSaveLocal

saveMovie

Syntaxe

```
saveMovie {nomDuChemin&nomDuFichier}
```

Description

Commande ; enregistre l'animation courante. L'inclusion du paramètre facultatif enregistre l'animation dans le fichier spécifié par *nomDuChemin:NomDuFichier*. 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.

Exemple

L'instruction suivante enregistre l'animation courante dans le fichier MiseAjour :

```
saveMovie the moviePath & "MiseAjour.dir"
```

Voir aussi

@ (chemin d'accès), setPref

scale

Syntaxe

```
member(quelActeur).scale
the scale of member quelActeur
sprite(quelleImageObjet).scale
the scale of sprite quelleImageObjet
```

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.

Exemples

Le script d'image suivant définit une boucle sur l'image courante alors que l'image-objet QuickTime de la piste 5 est mise à l'échelle par incréments de 5 %. Lorsque l'image-objet n'est plus visible (parce que sa mise à l'échelle horizontale est inférieure ou égale à 0 %), la tête de lecture passe à l'image suivante.

```
on exitFrame me
  facteurDéchelle = sprite(spriteNum).scale[1]
  numDacteurCourant = sprite(spriteNum).memberNum
  if member(numDacteurCourant).crop = FALSE then
    member(numDacteurCourant).crop = TRUE
  end if
  if facteurDéchelle > 0 then
    facteurDéchelle = facteurDéchelle - 5
    sprite(spriteNum).scale = [facteurDéchelle, facteurDéchelle]
    go the frame
  end if
end
```

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 %).

```
on miseAlEchelleDAnimation quelleImageObjet
  sprite(quelleImageObjet).scale = 0
  updatestage
  repeat with i = 1 to 20
    sprite(quelleImageObjet).scale = i * 5
    updatestage
  end repeat
end
```

Voir aussi

`scaleMode`, `originMode`

scale (fond et recouvrement)

Syntaxe

```
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 (commande)

Syntaxe

```
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 d'un objet de transformation d'un nœud référencé ou d'un objet de transformation directement référencé. 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.

Exemples

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("Séquence").model("Lune").transform.scale
-- vector(1.0000, 1.0000, 1.0000)
member("Séquence").model("Lune").scale(2.0,1.0,0.5)
put member("Séquence").model("Lune").transform.scale
-- vector( 1.0000, 0,5000, 0.0000 )
```

L'instruction suivante redimensionne le modèle Pluton de façon uniforme sur les trois axes selon la valeur 0.5, ce qui réduit de moitié la taille du modèle affiché.

```
member("Séquence").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("Séquence").model("Pluton").scale(0.0, 0.0, 0.5)
```

Voir aussi

`transform (propriété)`, `preScale()`, `scale (transformation)`

scale (transformation)

Syntaxe

```
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

Syntaxe

```
sprite(quelleImageObjetVecteurOuFlash).scaleMode
the scaleMode of sprite quelleImageObjetVecteurOuFlash
member(quelleImageObjetVecteurOuFlash).scaleMode
the scaleMode of member quelleImageObjetVecteurOuFlash
```

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`.

```
on beginsprite me
  if the stagecolor = 0 then
    sprite(me.spriteNum).scaleMode = #showAll
  else
    sprite(me.spriteNum).scaleMode = #noBorder
  end if
end
```

Voir aussi

`scale`

score

Syntaxe

```
the score
```

Description

Propriété d'animation ; détermine le scénario associé à l'animation courante. 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.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante affecte l'acteur boucle Cascade au scénario de l'animation courante :

```
the score = member("Cascade").media
```

scoreColor

Syntaxe

```
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

Syntaxe

```
the scoreSelection
```

Description

Propriété d'animation ; détermine les pistes sélectionnées dans la fenêtre Scénario. 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

Pour spécifier :	Utilisez :
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.

Cette propriété peut être testée et définie.

Exemples

L'instruction suivante sélectionne les pistes d'images-objets 15 à 25 dans les images 100 à 200 :

```
set the scoreSelection = [[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 :

```
set the scoreSelection = [[15, 25, 100, 200], [40, 50, 100, 200]]
```

L'instruction suivante sélectionne le script d'image des images 100 à 200 :

```
set the scoreSelection = [[0, 0, 100, 200]]
```

script

Syntaxe

```
the script of menuItem quelElément of menu quelMenu
objetEnfant.script
the script of objetEnfant
```

Description

Propriété d'élément de menu et d'objet enfant.

Pour les éléments de menu, détermine l'instruction Lingo exécutée lorsque l'élément de menu spécifié est sélectionné. L'expression *quelElément* peut correspondre au nom ou au numéro d'un élément de menu ; l'expression *quelMenu* peut être le nom ou le numéro d'un menu.

Lorsqu'un menu est installé, le script est défini comme le texte qui suit le caractère « Å » dans la définition du menu.

Cette propriété peut être testée et définie.

Remarque Les menus ne sont pas disponibles dans Shockwave.

Pour les objets enfants, la valeur renvoyée est le nom du script parent de l'objet enfant. Cette propriété peut être testée, mais pas définie.

Vous pourrez voir un exemple de `script` dans une animation en consultant l'animation Parent Scripts du dossier Learning/Lingo_Examples, lui-même dans le dossier de Director.

Exemples

L'instruction suivante fait du gestionnaire *gestGo* le gestionnaire exécuté lorsque l'utilisateur choisit la commande Go dans le menu personnalisé Contrôle :

```
set the script of menuItem "Go" of menu "Contrôle" to "gestGo"
```

L'instruction Lingo suivante vérifie si un objet enfant est une instance du script parent Fourmi :

```
if objetInsecte.script = script("Fourmi") then
  objetInsecte.attaque()
end if
```

Voir aussi

checkMark, installMenu, menu, handlers(), scriptText

scriptInstanceList

Syntaxe

```
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.

Exemples

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

Syntaxe

```
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

Syntaxe

```
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

Syntaxe

```
member(quelActeur).scriptsEnabled
the scriptsEnabled of member quelActeur
```

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 :

```
member("Chronique Jazz").scriptsEnabled = FALSE
```

scriptText

Syntaxe

```
member(quelActeur).scriptText
the scriptText of member quelActeur
```

Description

Propriété d'acteur ; indique le contenu du script éventuellement affecté à l'acteur spécifié par *quelActeur*.

Le texte d'un script est supprimé lorsqu'une animation est convertie en projection, protégée ou compressée au format Shockwave. De telles animations perdent alors les valeurs de leur propriété d'acteur `scriptText`. Par conséquent, les valeurs de la propriété d'acteur `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 d'acteur `scriptText` dans la projection. Ces scripts nouvellement affectés sont compilés automatiquement pour en garantir une exécution rapide.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante fait du contenu de l'acteur champ 20 le script de l'acteur 30 :

```
member(30).scriptText = member(20).text
```

scriptType

Syntaxe

```
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
```

scrollByLine

Syntaxe

```
member(quelActeur).scrollByLine(quantité)
scrollByLine member quelActeur, quantité
```

Description

Commande ; fait défiler l'acteur champ ou texte spécifié vers le haut ou le bas du nombre de lignes spécifié dans *quantité*. Les lignes sont définies comme des lignes séparées par des retours chariot, et non comme des lignes produites par un retour automatique.

- Lorsque *quantité* a une valeur positive, le champ défile vers le bas.
- Lorsque *quantité* a une valeur négative, le champ défile vers le haut.

Exemples

L'instruction suivante fait défiler l'acteur champ Nouvelles du jour de cinq lignes vers le bas :

```
member("Nouvelles du jour").scrollbyline(5)
```

L'instruction suivante fait défiler l'acteur champ Nouvelles du jour de une ligne vers le haut :

```
scrollByLine member "Nouvelles du jour", -1
```

scrollByPage

Syntaxe

```
member(quelActeur).scrollByPage(quantité)
scrollByPage member quelActeur, quantité
```

Description

Commande ; fait défiler l'acteur champ ou texte spécifié vers le haut ou le bas du nombre de pages spécifié dans *quantité*. Une page correspond au nombre de lignes de texte visibles à l'écran.

- Lorsque *quantité* a une valeur positive, le champ défile vers le bas.
- Lorsque *quantité* a une valeur négative, le champ défile vers le haut.

Le lecteur Director pour Java ne supporte pas la propriété d'acteur `scrollByPage`. Utilisez la propriété d'acteur `scrollTop` pour rédiger des instructions Lingo servant à faire défiler le texte.

Exemples

L'instruction suivante fait défiler l'acteur champ Nouvelles du jour d'une page vers le bas :

```
member("Nouvelles du jour").scrollbypage(1)
```

L'instruction suivante fait défiler l'acteur champ Nouvelles du jour d'une page vers le haut :

```
member("Nouvelles du jour").scrollbypage(-1)
```

Voir aussi

`scrollTop`

scrollTop

Syntaxe

```
member(quelActeur).scrollTop
the scrollTop of member quelActeur
```

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, le Lingo suivant 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` :

```
on wa
  member("Crédits").scrollTop = 1
  repeat with count = 1 to 150
    member("Crédits").scrollTop = member("Crédits").scrollTop + 1
    updateStage
  end repeat
end
```

sds (modificateur)

Syntaxe

`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

Syntaxe

```
the searchCurrentFolder
```

Description

Propriété système ; détermine si Director traite le dossier courant lors d'une recherche de noms de fichiers. Cette propriété est `TRUE` par défaut.

- 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 peut être testée et définie.

Exemples

L'instruction suivante affiche l'état de la propriété `searchCurrentFolder` dans la fenêtre Messages :

```
put the searchCurrentFolder
```

Le résultat est 1, équivalent numérique de `TRUE`.

L'instruction suivante donne à la propriété `searchCurrentFolder` la valeur `TRUE`, qui demande à Director de traiter le dossier courant lors de la recherche de noms de fichiers :

```
the searchCurrentFolder = TRUE
```

Voir aussi

`searchPaths`

searchPath

Obsolète. Utilisez `searchPaths`.

searchPaths

Syntaxe

```
the searchPaths
```

Description

Propriété système ; 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. Chaque élément de la liste est un nom de chemin d'accès complet tel qu'il apparaît sur la plate-forme courante pendant l'exécution.

La valeur de `searchPaths` 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`.

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.

Cette propriété peut être testée et définie et correspond par défaut à une liste vide.

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.

Exemples

L'instruction suivante affiche les chemins d'accès que Director prend en compte lors de la recherche de fichiers :

```
put the searchPaths
```

L'instruction suivante affecte deux dossiers à `searchPaths` sous Windows. Cette version comprend des barres obliques inverses de fin facultatives.

```
set the searchPaths = ["c:\director\projets\", "d:\cdrom\sources\"]
```

L'instruction suivante est la même, mais ne contient pas de barres obliques inverses de fin :

```
set the searchPaths = ["c:\director\projets", "d:\cdrom\sources"]
```

L'instruction suivante affecte deux dossiers à `searchPaths` sur un Macintosh. Cette version comprend les deux points facultatifs de fin.

```
set the searchPaths = ["disque dur:director:projets:", "cdrom:sources:"]
```

L'instruction suivante est la même, mais ne contient pas les deux points de fin :

```
set the searchPaths = ["disque dur:director:projets", "cdrom:sources"]
```

Les instructions suivantes indiquent à Director de rechercher à l'intérieur d'un dossier appelé Sons, qui se trouve dans le même dossier que l'animation Director courante :

```
set cheminDesSons = the moviePath & "Sons"
add the searchPaths, cheminDesSons
```

Voir aussi

`searchCurrentFolder`, `@ (chemin d'accès)`

seconds

Syntaxe

objetDate.seconds

Description

Propriété ; indique le nombre de secondes écoulées depuis minuit pour l'objet de date donné. Seuls systemDate, creationDate et modifiedDate ont une valeur seconds par défaut. Vous devez spécifier une valeur en seconds pour les objets de date créés.

Cette propriété peut être utilisée avec les valeurs creationDate et modifiedDate en vue du contrôle de la source.

Exemple

Les instructions suivantes affichent les secondes écoulées depuis minuit sur l'ordinateur de programmation :

```
maDate = the systemdate
put maDate.seconds
-- 1233
```

seek

Syntaxe

sprite(*quelleImageObjet*).seek(*millisecondes*)

member(*quelActeur*).seek(*millisecondes*)

Description

Méthode d'acteur ou d'image-objet RealMedia ; modifie l'emplacement de lecture du train multimédia vers l'emplacement spécifié par le nombre de millisecondes écoulées depuis le début du train. 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 train RealMedia, ou pour avancer ou reculer dans le train. Le nombre entier spécifié dans *millisecondes* correspond au nombre de millisecondes écoulées depuis le début du train. 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 train 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 train repasse en tampon et sa lecture démarre automatiquement au nouvel emplacement du train. 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 (RealMedia) du train, l'argument entier spécifié est ajouté à la plage à partir de 0 pour la durée du train. Vous ne pouvez pas accéder directement à une image-objet RealMedia qui est en cours de lecture en flux continu.

Remarquez que x.seek(n) est l'équivalent de x.currentTime (RealMedia) = n, et l'un ou l'autre de ces appels provoqueront la remise en mémoire tampon du train.

Exemples

Les exemples suivants définissent la position de lecture actuelle du train sur 10 000 millisecondes (10 secondes) :

```
sprite(2).seek(10000)
member("Real").seek(10000)
```

Voir aussi

`duration (RealMedia)`, `currentTime (RealMedia)`, `play (RealMedia)`, `pause (RealMedia)`, `stop (RealMedia)`, `mediaStatus`

selectedText

Syntaxe

`member(`*`quelActeurTexte`*`).selectedText`

Description

Propriété d'acteur texte ; renvoie le bloc de texte sélectionné sous forme de référence à 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.

```
property spriteNum

on mouseUp me
  monObjetSélection = sprite(spriteNum).member.selectedText
  put monObjetSélection.text
  put monObjetSélection.font
  put monObjetSélection.fontSize
  put monObjetSélection.fontStyle
end
```

selection() (fonction)

Syntaxe

`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.

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`

selection (propriété de distribution)

Syntaxe

```
castLib (quelleDistrib).selection
the selection of castLib quelleDistrib
set the selection of castLib quelleDistrib =[ [ acteurInitial1 , acteurFinal1
  ], \
{[ acteurInitial2 , acteurFinal2 ], [ acteurInitial3 , acteurFinal2 ]...]}
castLib(quelleDistrib). selection =[ [ acteurInitial2 , acteurFinal1 ], \
{[ acteurInitial2 , acteurFinal2 ], [ acteurInitial3 , acteurFinal2 ]...]}
```

Description

Propriété d'acteur ; détermine les acteurs sélectionnés dans la fenêtre Distribution spécifiée. La plage apparaît sous la forme d'une liste comprenant les numéros des acteurs initiaux et finaux de la plage sélectionnée. Vous pouvez inclure plus d'une sélection en indiquant des plages supplémentaires d'acteurs. Pour spécifier plus d'une sélection, faites glisser le curseur autour de la sélection tout en appuyant sur la touche Ctrl (Windows) ou Cmd (Macintosh).

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante sélectionne les acteurs 1 à 10 de la distribution numéro 1 :

```
castLib(1).selection = [[1, 10]]
```

L'instruction suivante sélectionne les acteurs 1 à 10 et 30 à 40 de la distribution numéro 1 :

```
castLib(1).selection = [[1, 10], [30, 40]]
```

selection (propriété d'acteur texte)

Syntaxe

```
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

Syntaxe

```
the selEnd
```

Description

Propriété globale ; spécifie le caractère final 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.

Exemples

Les instructions suivantes sélectionnent « cde » à partir du champ « abcdefg » :

```
the selStart = 3
the selEnd = 5
```

L'instruction suivante appelle le gestionnaire `aucuneSélection` si `selEnd` coïncide avec `selStart` :

```
if the selEnd = the selStart then aucuneSélection
```

L'instruction suivante opère une sélection d'une longueur de 20 caractères :

```
the selEnd = the selStart + 20
```

Voir aussi

`editable`, `hilite (commande)`, `selection() (fonction)`, `selStart`, `text`

selStart

Syntaxe

```
the 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.

Exemples

Les instructions suivantes sélectionnent « cde » à partir du champ « abcdefg » :

```
the selStart = 3
the selEnd = 5
```

L'instruction suivante appelle le gestionnaire `aucuneSélection` si `selEnd` coïncide avec `selStart` :

```
if the selEnd = the selStart then aucuneSélection
```

L'instruction suivante opère une sélection d'une longueur de 20 caractères :

```
the selEnd = the selStart + 20
```

Voir aussi

`selection() (fonction)`, `selEnd`, `text`

sendAllSprites

Syntaxe

```
sendAllSprites (#événementPersonnalisé, arguments)
```

Description

Commande ; 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 commande `stopEvent` ne soit utilisée.

Pour de meilleurs résultats, n'envoyez le message qu'aux images-objets capables de gérer correctement le message par le biais de la commande `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 message classique : script d'acteur, script d'image, puis script d'animation.

Lorsque vous utilisez la commande `sendAllSprites`, effectuez les opérations suivantes :

- Remplacez *`événementPersonnalisé`* 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`.

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 :

```
on mouseDown me
  sendAllSprites (#toutesLesImagesObjetsDoiventAugmenterLeCompteur, 2)
end
```

Voir aussi

`sendSprite`

sendEvent

Syntaxe

```
member(quelActeur).sendEvent(#nomDévénement, arg1, arg2,...)
```

Description

Commande 3D ; envoie l'événement *`nomDévénement`* et un nombre arbitraire d'arguments (*`arg1, arg2, ..`*) à tous les scripts enregistrés pour recevoir l'événement. Utilisez `registerForEvent()` ou `setCollisionCallback()` pour enregistrer les scripts pour les événements.

Exemple

La première ligne crée l'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`. Veuillez noter qu'une valeur d'instance de script de 0 indique que vous enregistrez le gestionnaire d'un script d'animation, ce qui est différent du gestionnaire d'une instance de comportement ou de l'enfant d'un script parent.

```
t = new (script "Testeur")
member("Séquence").registerForEvent(#saut, #sautPluton, t)
member("séquence").registerForEvent(#saut, #sautMars, 0)
member("séquence").sendEvent(#saut)
```

Voir aussi

`registerScript()`, `registerForEvent()`, `setCollisionCallback()`

sendSprite

Syntaxe

`sendSprite (`*`quelleImageObjet`*`, `*`#messagePersonnalisé`*`, `*`arguments`*`)`

Description

Commande ; 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`.

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 :

```
on mouseDown me
  sendSprite (1, #augmenterLeCompteur, 2)
end
```

Voir aussi

`sendAllSprites`

on sendXML

Syntaxe

`sendXML "`*`chaîneSendXML`*`", "`*`fenêtre`*`", "`*`données`*`"`

Description

Gestionnaire d'événement ; fonctionne de manière similaire à la méthode `getURL` également disponible avec l'Xtra Flash Asset. Le gestionnaire `on sendXML` est appelé dans 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 dans 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'une animation 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 :

```
on sendXML me, Lurl, fenêtreCible, donnéesXml
  gotoNetPage Lurl, fenêtreCible
  postNetText(Lurl, donnéesXml)
end
```

serialNumber

Syntaxe

```
the serialNumber
```

Description

Propriété d'animation ; renvoie une chaîne comprenant le numéro de série saisi pendant l'installation de Director.

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 :

```
on prepareMovie
  chaîneAffichée = the userName
  put RETURN&the organizationName after chaîneAffichée
  put RETURN&the serialNumber after chaîneAffichée
  member("Infos utilisateur").text = chaîneAffichée
end
```

Voir aussi

organizationName, userName, window

set...to, set...=

Syntaxe

```
set the propriétéLingo to expression
the propriétéLingo = expression
set variable to 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*.

Exemples

L'instruction suivante donne à l'acteur 3 le nom de Coucher de soleil :

```
set 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.

```
set the soundEnabled = not (the soundEnabled)
```

L'instruction suivante affecte à la variable *voyelles* la chaîne « aeiou » :

```
set voyelles = "aeiou"
```

Voir aussi

property

setAlpha()

Syntaxe

```
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.

Exemples

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

Syntaxe

```
setaProp liste, 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, Lingo 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, Lingo ajoute la nouvelle propriété et sa valeur.
- La commande `setaProp` peut également définir des propriétés `ancestor`.

Exemples

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

Syntaxe

```
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*.

Exemples

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()

Syntaxe

`réfDimageObjetFlash``.setCallback(`*`objetAS`*`, `*`nomDévénAS`*`, #`*`gestLingo`*`, `*`objetLingo`*`)`

`setCallback(`*`objetAS`*`, `*`nomDévénAS`*`, #`*`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. Les Xtras sont ajoutés à la liste des Xtras via Modification › Animation › Xtras. Pour plus d'informations, consultez Gestion des Xtras des animations distribuées dans le mode d'emploi de Director.

Exemples

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 :

```
sprite(3).setCallback(tObjetConnexionLocale, "onStatus", #monOnStatus, me)
```

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 global `tObjetConnexionLocale` :

```
setCallback(tObjetConnexionLocale, "onStatus", #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.

```
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
```

Voir aussi

`newObject()`, `clearAsObjects()`

setCollisionCallback()

Syntaxe

```
member(quelActeur).model(quelModèle).collision.\
  setCollisionCallback(#nomDeGestionnaire, instanceDeScript)
```

Description

Commande 3D de collision ; enregistre le gestionnaire *#nomDeGestionnaire* dans l'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`.

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`

setFlashProperty()

Syntaxe

```
sprite(numéroDimageObjet).setFlashProperty("nomDeCible", #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.

nomDeCible est le nom du clip ou du niveau de l'animation dont vous souhaitez définir la propriété dans l'image-objet Flash donnée.

#propriété est le nom de la propriété à définir. Vous pouvez définir les propriétés suivantes : `#posX`, `#posY`, `#scaleX`, `#scaleY`, `#visible`, `#rotate`, `#alpha` et `#name`.

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.

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.

```
sprite(3).setFlashProperty("Etoile", #rotate, 180)
```

Voir aussi

`getFlashProperty()`

setPixel()

Syntaxe

```
objetImage.setPixel(x, y, objetCouleur)
objetImage.setPixel(point(x, y), objetCouleur)
objetImage.setPixel(x, y, valeurEntière)
objetImage.setPixel(point(x, y), valeurEntière)
```

Description

Fonction ; définit la couleur du pixel au point spécifié dans l'objet image donné, sur *objetCouleur* ou sur un nombre entier brut avec *valeurEntière*. 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 bruts.

Pour obtenir des performances maximales avec les objets couleur, utilisez un objet couleur indexée pour les images 8 bits (ou une valeur inférieure) et un objet couleur rvb pour des images 16 bits (ou une valeur supérieure).

La fonction `SetPixel()` renvoie `FALSE` si le pixel spécifié se trouve en dehors de l'image.

Vous pourrez voir un exemple de `setPixel()` dans une animation en consultant l'animation Imaging du dossier Learning/Lingo_Examples, lui-même dans le dossier de Director.

Exemple

L'instruction Lingo suivante dessine une ligne noire horizontale de 50 pixels, de gauche à droite, dans l'acteur 5 :

```
repeat with x = 1 to 50
  member(5).image.setPixel(x, 0, rgb(0, 0, 0))
end repeat
```

Voir aussi

`getPixel()`, `draw()`, `fill()`, `color()`

setPlaylist()

Syntaxe

```
sound(numéroDePiste).setPlaylist([#member: member(quelActeur), {#startTime:
  millisecondes, #endTime: millisecondes, #loopCount: nombreDeBoucles,
  #loopStartTime: millisecondes, #loopEndTime: millisecondes, #preloadTime:
  millisecondes}]. . . )
setPlaylist(sound(numéroDePiste), [#member: member(quelActeur), {#startTime:
  millisecondes, #endTime: millisecondes, #loopCount: nombreDeBoucles,
  #loopStartTime: millisecondes, #loopEndTime: millisecondes, #preloadTime:
  millisecondes}]. . . )
```

Description

Fonction : définit ou réinitialise la liste de lecture de la piste audio concernée. Cette commande est pratique pour placer plusieurs sons en file d'attente en une seule opération.

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`.

Vous pourrez voir un exemple de `setPlaylist()` dans une animation en consultant l'animation Sound Control du dossier Learning/Lingo_Examples, lui-même dans le dossier de Director.

Exemple

Le gestionnaire suivant place en file d'attente et lit l'acteur son Intro, à partir de ses 3 secondes et exécute cinq lectures en boucle successives, de 8 secondes à 8,9 secondes, et s'arrête au point 10 secondes.

```
on lireLaMusique
  sound(2).queue([#member: member("Intro"), #startTime: 3000,\
  #endTime: 10000, #loopCount: 5, #loopStartTime: 8000, #loopEndTime: 8900])
  sound(2).play()
end
```

Voir aussi

`endTime`, `getPlaylist()`, `startTime`, `loopCount`, `loopEndTime`, `loopStartTime`, `member (propriété audio)`, `play() (audio)`, `preLoadTime`, `queue()`

setPref

Syntaxe

```
setPref nomDePréf, valeurDePréf
```

Description

Commande ; écrit la chaîne spécifiée par *valeurDePréf* dans le fichier spécifié par *nomDePréf* sur le disque local de l'ordinateur. Le fichier est un fichier texte standard.

L'argument *nomDePréf* 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 commande `setPref`, si l'animation est lue dans un navigateur web, un dossier nommé Prefs est créé à l'intérieur du dossier Plug-In Support. La commande `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 commande 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 commande 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 web, les données écrites par `setPref` ne sont pas confidentielles. Toute animation 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.

Vous pourrez voir un exemple de `setPref` dans une animation en consultant l'animation Read and Write Text du dossier Learning/Lingo_Examples, lui-même dans le dossier de Director.

Exemple

Le gestionnaire suivant enregistre le contenu de l'acteur champ Saisie dans un fichier nommé PréfsActuelles :

```
on mouseUp me
  setPref "PréfsActuelles", member("Saisie").text
end
```

Voir aussi

`getPref()`

setProp

Syntaxe

```
setProp liste, propriété, nouvelleValeur
liste.propriétéDeListe = nouvelleValeur
liste[propriétéDeListe] = nouvelleValeur
```

Description

Commande ; remplace la valeur affectée à *propriété* par la valeur spécifiée par *nouvelleValeur* dans la liste spécifiée par *liste*. 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.

Exemples

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()

Syntaxe

```
réfDimageObjet.setScriptList(listeDeScripts)
sprite(quelleImageObjet).setScriptList(listeDeScripts)
```

Description

Cette commande définit la liste *listeDeScript* de l'image-objet donnée. La liste *listeDeScript* 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.

Voir aussi

`scriptList, value(), string()`

settingsPanel()

Syntaxe

réfDimageObjet.settingsPanel({*entierIndexPanneau*})

sprite(*quelleImageObjet*).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 valeur de *entierIndexPanneau* peut être 0, 1, 2 ou 3, indiquant le panneau à activer lorsque la boîte de dialogue est ouverte. La valeur 0 ouvre la boîte de dialogue présentant 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 par défaut est 0.

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, disponible en cliquant du bouton droit (Windows) ou en cliquant avec la touche Ctrl enfoncée (Macintosh) dans une animation Shockwave lue dans un navigateur. Pour plus d'informations sur la désactivation de ce menu, consultez Utilisation du panneau des paramètres de Flash dans l'aide de Director (Aide > Utilisation de Director).

Exemple

L'instruction suivante ouvre le panneau des paramètres de Flash, qui présente l'onglet Enregistrement local :

```
sprite(3).settingsPanel(1)
```

Voir aussi

`on mouseDown (gestionnaire d'événement)`, `rightMouseDown (propriété système)`, `controlDown`

setTrackEnabled

Syntaxe

sprite(*quelleImageObjet*).setTrackEnabled(*quellePiste*, *trueOuFalse*)
setTrackEnabled(sprite *quelleImageObjet*, *quellePiste*, *trueOuFalse*)

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`.

Exemple

L'instruction suivante active la piste 3 de la vidéo numérique affectée à la piste d'image-objet 8 :

```
sprite(8).setTrackEnabled(3, TRUE)
```

Voir aussi

`trackEnabled`

setVariable()

Syntaxe

```
setVariable(sprite numéroDimageObjetFlash, "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.

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.fr/software/flash/ ».

```
setVariable(sprite 3, "URLcourante", "http://www.macromedia.fr/software/flash/
  ")
```

Voir aussi

`hitTest()`, `getVariable()`

shader

Syntaxe

```
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`.

Vous trouverez une liste complète des propriétés de matériau dans Lingo 3D par fonction.

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`

shaderList

Syntaxe

```
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
```

Exemples

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

Syntaxe

```
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

Syntaxe

```
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

Syntaxe

```
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

```
the shiftDown
```

Description

Propriété système ; indique si l'utilisateur appuie sur la touche Maj (`TRUE`) ou non (`FALSE`).

Dans le lecteur Director pour Java, cette fonction renvoie la valeur `TRUE` seulement si l'utilisateur appuie simultanément sur la touche Maj et sur une autre touche. S'il appuie uniquement sur la touche Maj, `shiftDown` renvoie la valeur `FALSE`.

Le lecteur Director pour Java supporte les combinaisons de touches avec la touche Maj. Néanmoins, le navigateur web reçoit les touches avant l'animation et, par conséquent, intercepte et répond à des combinaisons de touches qui correspondent également à des raccourcis de clavier du navigateur.

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* :

```
if (the shiftDown) then MajusculeA (the key)
```

Voir aussi

commandDown, controlDown, key(), optionDown

shininess

Syntaxe

```
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
```

short

Consultez

date() (horloge du système), time()

showGlobals

Syntaxe

```
showGlobals
```

Description

Commande ; affiche toutes les variables globales dans la fenêtre Messages. Cette commande est pratique pour déboguer les scripts.

Exemple

L'instruction suivante affiche toutes les variables globales dans la fenêtre Messages :

```
showGlobals
```

Voir aussi

clearGlobals, global, globals

showLocals

Syntaxe

```
showLocals
```

Description

Commande ; 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.

Voir aussi

clearGlobals, global, showGlobals

showProps()

Syntaxe

```
member(quelActeurFlashOuVecteur).showProps
showProps()
sprite(quelActeurFlashOuVecteur).showProps()
sound(numéroDePiste).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 Shockwave.

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 :

```
on AfficherPropDistribution quelleDistrib
  repeat with i = 1 to the number of members of castLib quelleDistrib
    typeDeDistribution = member(i, quelleDistrib).type
    if (typeDeDistribution = #flash) OR (typeDeDistribution = #vectorShape)
  then
      put typeDeDistribution&&"acteur" && i & " :" && member(i,
  quelleDistrib).name
      put RETURN
      member(i ,quelleDistrib).showProps()
    end if
  end repeat
end
```

Voir aussi

queue(), setPlaylist()

showResFile

Description

Ce terme Lingo est obsolète.

showXlib

Syntaxe

```
showXlib {nomDeFichierXlib}
```

Description

Commande ; indique tous les Xtras contenus dans le fichier *nomDeFichierXlib* (qui doit être ouvert) ou toutes les Xlibraries ouvertes si aucun fichier n'est indiqué. Les fichiers Xlibrary sont des fichiers de ressources contenant des ressources DLL (Windows) ou Xtra (Macintosh). Le type des fichiers Xlibrary étant différent sous Windows et Macintosh, la liste de fichiers générée par la commande `showXlib` varie selon les plates-formes.

La commande `showXlib` ne supporte pas les adresses URL comme références de fichiers.

Vous pouvez utiliser `interface()` pour afficher la documentation en ligne concernant un Xtra.

Remarque Cette commande n'est pas supportée dans Shockwave.

Exemple

L'instruction suivante affiche les Xtras de la Xlibrary VidéoDisque :

```
showXlib "Xlibrary VidéoDisque"
```

Voir aussi

`closeXlib`, `interface()`, `openXlib`

shutDown

Syntaxe

```
shutDown
```

Description

Commande ; ferme toutes les applications ouvertes et éteint l'ordinateur.

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 :

```
if the key = "s" and the commandDown then
  shutDown
end if
```

Voir aussi

`quit`, `restart`

silhouettes

Syntaxe

```
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
```

sin()

Syntaxe

```
sin(angle)
```

Description

Fonction mathématique ; calcule le sinus de l'angle spécifié. Celui-ci doit être exprimé en radians sous forme de nombre à virgule flottante.

Exemple

L'instruction suivante calcule le sinus de `pi`/2 :

```
put sin (PI/2.0)
-- 1
```

Voir aussi

`PI`

size

Syntaxe

```
member(quelActeur).size
the size of member quelActeur
```

Description

Propriété d'acteur ; renvoie la taille occupée en mémoire, en octets, d'un acteur indiqué par son nom ou son numéro. Multipliez 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 :

```
member("Taille").text = string(member("Temple").size)
```

sizeRange

Syntaxe

```
member(quelActeur).modelResource
(quelleRessDeMod).sizeRange.start
référenceDobjetDeRessourceDeModèle.sizeRange.start
member(quelActeur).modelResource
(quelleRessDeMod).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, systèmeThermique 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
```

skew

Syntaxe

```
sprite(quelNuméroDimageObjet).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. 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 entre +21 474 836,47° et -21 474 836,48° et 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 Lingo.

Exemple

Le comportement suivant incline l'image-objet continuellement de 2° à chaque fois que la tête de lecture avance, tout en limitant l'angle à 360° :

```
property spriteNum

on prepareFrame me
  sprite(spriteNum).skew = integer(sprite(spriteNum).skew + 2) mod 360
end
```

Voir aussi

flipH, flipV, rotation

smoothness

Syntaxe

```
member(quelActeurTexte).smoothness
member(quelActeur).modelResource(quelleRessDeModExtrudeur)\
  .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

sort

Syntaxe

```
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.

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

Syntaxe

```
member(quelActeur).sound
the sound of member quelActeur
```

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). Dans les acteurs Flash, le nouveau paramètre prend effet après la lecture du son courant.

Cette propriété peut être testée et définie.

Vous pourrez voir un exemple de sound dans une animation en consultant l'animation Sound Control du dossier Learning/Lingo_Examples, 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 :

```
on ToggleSound quelActeur
  member(quelActeur).sound = not member(quelActeur).sound
end
```

soundBusy()

Syntaxe

```
soundBusy(quellePiste)
```

Description

Fonction ; détermine si un son est lu (TRUE) ou non (FALSE) dans la piste audio spécifiée par *quellePiste*.

Vérifiez d'abord que la tête de lecture a bougé avant d'utiliser soundBusy() pour vérifier la piste audio. Si cette fonction continue de renvoyer la valeur FALSE après la lecture présumée d'un son, ajoutez la commande updateStage pour démarrer la lecture du son avant que la tête de lecture ne se remette à bouger.

Cette fonction fonctionne avec les pistes audio occupées par des acteurs son réels. Les composants audio QuickTime, Flash et Shockwave gérant le son différemment, cette fonction 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 soundBusy(). La propriété status s'avère plus appropriée dans la plupart des cas.

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.

```
if soundBusy(1) then go to the frame
```

Voir aussi

`sound playFile`, `sound stop`, `status`

soundChannel (SWA)

Syntaxe

```
member(quelActeur).soundChannel
the soundChannel of member quelActeur
```

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 :

```
member("Frank Zappa").soundChannel = 3
```

soundChannel (RealMedia)

Syntaxe

```
sprite(quelleImageObjet).soundChannel
member(quelActeur).soundChannel
sprite(quelleImageObjet).soundChannel = pisteAudio
member(quelActeur).soundChannel = 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 train RealMedia. La définition de cette propriété permet de contrôler l'audio d'un train RealMedia à l'aide des méthodes et propriétés audio Lingo. La définition de cette propriété à une valeur non comprise entre 0 et 8 entraîne une erreur Lingo. 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.

Exemples

Les exemples suivants indiquent que le son du train RealMedia de l'image-objet 2 et de l'acteur Real sera lu dans la piste audio 2.

```
put sprite(2).soundChannel
-- 2

put member("Real").soundChannel
-- 2
```

Les exemples suivants affectent la piste audio 1 au train RealMedia de l'image-objet 2 et l'acteur Real.

```
sprite(2).soundChannel = 1

member("Real").soundChannel = 1
```

Voir aussi

`realPlayerNativeAudio()`

sound close

Obsolète. Utilisez `puppetSound`.

soundDevice

Syntaxe

```
the soundDevice
```

Description

Propriété système ; permet le réglage du dispositif de mixage audio pendant la lecture de l'animation. Les paramètres possibles sont les dispositifs énumérés par `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.
- 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.

```
set the soundDevice = "MacroMix"
```

Voir aussi

`soundDeviceList`

soundDeviceList

Syntaxe

```
the soundDeviceList
```

Description

Propriété système ; crée une liste linéaire des dispositifs audio disponibles sur l'ordinateur utilisé.

Pour le Macintosh, cette propriété ne recense qu'un dispositif, à savoir MacSoundManager.

Cette propriété peut être testée, mais pas définie.

Exemple

L'instruction suivante affiche une liste de types de dispositifs audio typique sur un ordinateur Windows :

```
Put the soundDeviceList
--["QT3Mix", "MacroMix", "DirectSound"]
```

Voir aussi

`soundDevice`

soundEnabled

Syntaxe

```
the soundEnabled
```

Description

Propriété système ; détermine si le son est activé (`TRUE`, valeur par défaut) ou désactivé (`FALSE`).

Lorsque vous donnez à cette propriété la valeur `FALSE`, le son est désactivé, mais le réglage du volume ne change pas.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante inverse le paramètre courant de `soundEnabled` (active le son s'il est désactivé et vice-versa) :

```
the soundEnabled = not(the soundEnabled)
```

Voir aussi

`soundLevel`, `volume (piste audio)`, `volume (propriété d'image-objet)`

sound fadeIn

Syntaxe

```
sound(quellePiste).fadeIn()
sound(quellePiste).fadeIn(millisecondes)

sound fadeIn quellePiste
sound fadeIn quellePiste, battements
```

Description

Commande ; provoque une augmentation progressive du son sur la durée spécifiée.

La syntaxe `sound( quellePiste).fadeIn()` entraîne le fondu sur un certain nombre d'images ou de millisecondes.

- Lorsque *millisecondes* est spécifié, l'amplification du son a lieu progressivement pendant cette période de temps.
- Lorsque *millisecondes* n'est pas spécifié, le nombre par défaut de battements est calculé comme 15 * (1000 / (paramètre de cadence)) en fonction du paramètre de cadence de la première image de fondu.

La syntaxe `sound fadeIn` *quellePiste* entraîne le fondu sur un certain nombre d'images ou de battements. Un battement est un 60ème de seconde.

- Lorsque *battements* est spécifié, l'amplification du son a lieu progressivement pendant cette période de temps.
- Lorsque *battements* n'est pas spécifié, le nombre par défaut de battements est calculé comme 15 * (60 / (paramètre de cadence)) en fonction du paramètre de cadence de la première image de fondu.

L'amplification continue à un rythme prédéterminé jusqu'à ce que la durée soit écoulée ou que le son de la piste indiqué ait changé.

Exemple

L'instruction suivante amplifie le son de la piste 1 pendant une durée de 5 secondes :

```
sound(1).fadeIn(5000)
```

Voir aussi

`sound fadeOut`, `fadeTo()`

sound fadeOut

Syntaxe

```
sound(quellePiste).fadeOut()
sound(quellePiste).fadeOut(millisecondes)

sound fadeOut quellePiste
sound fadeOut quellePiste, battements
```

Description

Commande ; atténue progressivement le son sur une plage d'images ou de battements dans la piste audio spécifiée.

La syntaxe `sound( quellePiste).fadeOut()` entraîne le fondu sur un certain nombre d'images ou de millisecondes.

- Lorsque *millisecondes* est spécifié, l'amplification du son a lieu progressivement pendant cette période de temps.
- Lorsque *millisecondes* n'est pas spécifié, le nombre par défaut de battements est calculé comme 15 * (1000 / (paramètre de cadence)) en fonction du paramètre de cadence de la première image de fondu.

La syntaxe `sound fadeOut` *quellePiste* entraîne le fondu sur un certain nombre d'images ou de battements. Un battement est un 60ème de seconde.

- Lorsque *battements* est spécifié, l'atténuation du son a lieu progressivement pendant cette période de temps.
- Lorsque *battements* n'est pas spécifié, le nombre par défaut de battements est calculé comme 15 * (60 / (paramètre de cadence)) en fonction du paramètre de cadence de la première image de fondu.

L'atténuation continue à un rythme prédéterminé jusqu'à ce que le nombre de battements soit écoulé ou que le son de la piste indiquée ait changé.

Si le son est interrompu avant qu'il atteigne le volume minimum, il reste au niveau auquel il se trouvait lorsqu'il a été arrêté et sera lu ultérieurement à ce volume. Veillez donc à laisser l'atténuation progressive du son se dérouler normalement.

Remarque Il est conseillé d'utiliser la propriété `volume` pour créer une atténuation personnalisée du son, afin de mieux maîtriser le volume réel de la piste.

Exemple

L'instruction suivante diminue le son de la piste 1 pendant une durée de 5 secondes :

```
sound(1).fadeOut(5000)
```

Voir aussi

`puppetSound`, `sound fadeIn`, `volume (propriété d'image-objet)`, `fadeTo()`

soundKeepDevice

Syntaxe

```
the soundKeepDevice
```

Description

Propriété système ; pour Windows uniquement, évite au pilote audio d'être vidé et rechargé à chaque fois qu'un son doit être lu. La valeur par défaut est `TRUE`.

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 aux performances si le son est lu fréquemment par l'application Director.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante donne à la propriété `soundKeepDevice` la valeur `FALSE` :

```
set the soundKeepDevice = FALSE
```

soundLevel

Syntaxe

```
the soundLevel
```

Description

Propriété système ; définit le niveau de volume du son lu dans le haut-parleur de l'ordinateur. 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 le volume des pistes et des images-objets individuellement grâce à `volume of sound`, `volume of member` et `volume of sprite`.

Ces valeurs correspondent aux paramètres présents dans le tableau de commande Son du Macintosh. L'utilisation de cette propriété permet à Lingo de changer directement le volume du son ou d'effectuer une autre opération lorsque le son est situé à un niveau spécifié.

Cette propriété peut être testée et définie.

Vous pourrez voir un exemple de `soundLevel` dans une animation en consultant l'animation Sound Control du dossier Learning/Lingo_Examples, lui-même dans le dossier de Director.

Exemples

L'instruction suivante définit la variable *`ancienSon`* pour qu'elle corresponde au niveau sonore courant :

```
ancienSon = the soundLevel
```

L'instruction suivante règle le niveau sonore sur 5 :

```
the soundLevel = 5
```

Voir aussi

`soundEnabled`, `volume (piste audio)`

the soundMixMedia

Syntaxe

```
the soundMixMedia
```

Description

Cette propriété globale, lorsque définie sur `TRUE`, entraîne le mixage des sons des acteurs Flash avec les sons des pistes audio du scénario. Lorsqu'elle est définie sur `FALSE`, ces sons ne sont pas mixés et doivent être lus séparément. 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.

Cette propriété ne fonctionne que sous Windows. Lorsque `soundMixMedia` est défini sur `TRUE`, 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 des sons Flash, définissez cette propriété sur `FALSE`.

sound playFile

Syntaxe

```
sound playFile quellePiste, quelFichier
```

Description

Commande ; lit le son AIFF, SWA, AU ou WAV du fichier spécifié par *quelFichier* dans la piste audio spécifiée par *quellePiste*. 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 n'est pas placé dans le fichier de l'animation, *quelFichier* 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 commande `sound 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 commande `sound 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.

Exemples

L'instruction suivante lit le fichier Tonnerre dans la piste 1 :

```
sound playFile 1, "Tonnerre.wav"
```

L'instruction suivante lit le fichier Tonnerre dans la piste 3 :

```
sound playFile 3, the moviePath & "Tonnerre.wav"
```

Voir aussi

```
sound stop
```

sound stop

Syntaxe

```
sound(quellePiste).stop()
sound stop quellePiste
```

Description

Commande ; interrompt le son de la piste spécifiée.

La commande `sound stop` était utilisée dans les versions précédentes de Director. Pour de meilleurs résultats, utilisez plutôt la commande `puppetSound`.

Exemples

Les instructions suivantes arrêtent la lecture de tout son de la piste audio 1 :

```
sound(1).stop()
```

L'instruction suivante vérifie si un son est exécuté sur la piste audio 1 et, le cas échéant, l'interrompt :

```
if soundBusy(1) then sound(1).stop()
```

Voir aussi

puppetSound, soundBusy()

source

Syntaxe

```
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

Syntaxe

```
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

Syntaxe

```
window quelleFenêtre.sourceRect
the sourceRect of window quelleFenêtre
```

Description

Propriété de fenêtre ; détermine les coordonnées initiales de la scène de l'animation exécutée dans la fenêtre indiquée par *quelleFenêtre*.

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 :

```
put window("Tableau_de_commande").sourceRect
```

Voir aussi

drawRect, rect (caméra)

SPACE

Syntaxe

SPACE

Description

Constante ; valeur en lecture seule représentant une espace.

Exemple

L'instruction suivante affiche Age de bronze dans la fenêtre Messages :

```
put "Age"&SPACE&"de"&SPACE&"bronze"
```

specular (lumière)

Syntaxe

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)

Syntaxe

```
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

Syntaxe

```
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

Syntaxe

```
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

Syntaxe

```
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

Syntaxe

```
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

Syntaxe

```
sprite(quelleImageObjet).property
the property of sprite quelleImageObjet
```

Description

Mot-clé ; indique à Lingo que la valeur spécifiée par *quelleImageObjet* est un numéro de piste d'image-objet. Il s'utilise avec toutes les propriétés d'images-objets.

Une image-objet est une représentation d'un acteur dans une piste d'image-objet du scénario.

Ce terme possède une signification particulière dans le lecteur Director pour Java. N'utilisez pas le terme sprite dans le code Java incorporé dans un script Lingo.

Exemples

L'instruction suivante règle la variable intitulée *horizontale* sur the locH of sprite 1 :

```
horizontale = sprite(1).loc
```

L'instruction suivante affiche l'acteur courant de la piste d'image-objet 100 dans la fenêtre Messages :

```
put sprite(100).member
```

Voir aussi

puppetSprite, spriteNum

sprite...intersects

Syntaxe

```
sprite(imageObjet1).intersects(imageObjet2)
sprite imageObjet1 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

Syntaxe

```
sprite(imageObjet1).within(imageObjet2)
sprite imageObjet1 within imageObjet2
```

Description

Mot-clé ; opérateur comparant la position de deux images-objets pour déterminer si le quadrilatère de l'*imageObjet1* est entièrement 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

spriteNum

Syntaxe

```
spriteNum
the spriteNum of me
```

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. 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 `on new` pour créer une instance du comportement, le gestionnaire `on 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 `on new` sous forme d'argument lors de l'appel de ce gestionnaire.

Exemples

Dans le gestionnaire suivant, la propriété `spriteNum` est automatiquement définie pour les instances de script créées par le système :

```
property spriteNum

on mouseDown me
  sprite(spriteNum).member = member("imageEnfoncée")
end
```

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 :

```
property spriteNum, pRéfDimageObjet

on beginSprite me
  pRéfDimageObjet = sprite(me.spriteNum)
end
```

Cette approche permet d'utiliser la référence *`pRéfDimageObjet`* dans la suite du script, le gestionnaire utilisant la syntaxe

```
acteurCourant = pRéfDimageObjet.member
```

à la place de la syntaxe suivante, légèrement plus longue :

```
acteurCourant = sprite(spriteNum).member
```

Cette seconde approche existe uniquement dans un but de facilité et n'offre aucune fonction différente.

Voir aussi

`on beginSprite`, `on endSprite`, `currentSpriteNum`

spriteSpaceToWorldSpace

Syntaxe

```
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. L'emplacement spécifié par *`emplacement`* doit être un point relatif au coin supérieur gauche de l'image-objet.

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 rect d'affichage était de la même taille que 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()

Syntaxe

```
sqrt(nombre)
the sqrt of nombre
```

Description

Fonction mathématique ; renvoie la racine carrée du nombre indiqué par *`nombre`*, qui correspond soit à un nombre à virgule flottante, soit à un nombre entier arrondi à l'entier le plus proche.

La valeur doit être un nombre décimal supérieur à 0. Les valeurs négatives renvoient la valeur 0.

Exemples

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 4 dans la fenêtre Messages :

```
put sqrt(4)
-- 2
```

Voir aussi

`floatPrecision`

stage

Syntaxe

```
the stage
```

Description

Propriété système ; fait référence à l'animation principale.

Cette propriété est utile lorsque la commande `tell` est utilisée pour envoyer un message à l'animation principale à partir d'une animation enfant.

Exemples

L'instruction suivante fait passer l'animation principale vers le repère intitulé Menu. Elle peut être utilisée pour une animation dans une fenêtre :

```
tell the Stage to go to "Menu"
```

L'instruction suivante affiche les paramètres courants de la scène :

```
put the stage.rect
--rect (0, 0, 640, 480)
```

stageBottom

Syntaxe

```
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 cette applet en pixels.

Cette fonction peut être testée, mais pas définie.

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 le mode d'emploi de Director.

Voir aussi

`stageLeft`, `stageRight`, `stageTop`, `locH`, `locV`

stageColor

Syntaxe

```
the stageColor
```

Description

Propriété système ; détermine la couleur de l'arrière-plan de l'animation, pour les couleurs d'index uniquement.

Utilisez `bgColor` pour une définition de couleur de scène plus fiable, plus précise et plus souple avec des valeurs rvb.

La valeur de la propriété `stageColor` est comprise entre 0 et 255 pour une couleur codée sur 8 bits et entre 0 et 15 pour une couleur codée sur 4 bits. Vous pouvez cliquer sur une couleur dans la palette des couleurs pour voir le numéro d'index d'une couleur dans le coin inférieur gauche de la fenêtre. La définition de la propriété `stageColor` dans un script Lingo équivaut à choisir la couleur de la scène dans la palette déroulante du panneau.

Remarque Pour garantir la compatibilité nécessaire lors d'une lecture de l'animation en tant qu'applet Java, utilisez la propriété `bgColor` pour définir la couleur comme valeur rvb.

Exemples

L'instruction suivante affecte à la variable *ancienneCouleur* le numéro d'index de la couleur courante de la scène :

Lingo associé :

```
ancienneCouleur = the stageColor
```

L'instruction suivante affecte la valeur 249 de la palette courante à la couleur de la scène :

Fonction associée :

```
the stageColor = 249
```

Voir aussi

`backColor`, `bgColor`, `foreColor`, `color()`

stageLeft

Syntaxe

```
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.

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

Syntaxe

```
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.

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()

Syntaxe

```
sprite(quelActeurFlash).stageToFlash(pointDeLaScèneDeDirector)
stageToFlash (sprite quelActeurFlash, 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.

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.

```
on vérifDuSurvolFlash
  if sprite(5).stageToFlash(point(the mouseH,the mouseV)) = point(130,10) then
    sprite(5).stop()
  end if
end
```

Voir aussi

`flashToStage()`

stageTop

Syntaxe

```
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.

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`

startAngle

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).
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

Syntaxe

```
sprite(quelleImageObjet).startFrame
```

Description

Fonction ; renvoie le numéro de l'image de départ de l'étendue de l'image-objet.

Cette fonction permet de déterminer la plage 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. En outre, cette valeur ne peut pas être définie.

Exemple

L'instruction suivante affiche l'image de départ de l'image-objet dans la piste 5 dans la fenêtre Messages :

```
put sprite(5).startFrame
```

Voir aussi

```
endFrame()
```

on startMovie

Syntaxe

```
on startMovie
  instruction(s)
end
```

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 :

```
on startMovie
  repeat with compteur = 10 to 50
    sprite(compteur).visible = 0
  end repeat
end startMovie
```

Voir aussi

```
on prepareMovie
```

starts

Syntaxe

```
chaîne1 starts chaîne2
```

Description

Opérateur ; permet de déterminer si la *chaîne1* commence par la *chaîne2* (TRUE) ou non (FALSE).

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

L'instruction suivante affiche dans la fenêtre Messages si le mot *Macromedia* commence par la chaîne Macro :

```
put "Macromedia" starts "Macro"
```

Le résultat est 1, équivalent numérique de TRUE.

Voir aussi

contains

startTime

Syntaxe

```
sprite(quelleImageObjet).startTime
the startTime of sprite quelleImageObjet
sound(numéroDePiste).startTime
```

Description

Propriété d'image objet et de son ; pour les images-objets de vidéos numériques, détermine le moment de départ de l'image-objet de la vidéo numérique. La valeur de startTime est mesurée en battements.

Pour les images-objets vidéo numérique, cette propriété peut être testée et définie. Définissez la position de départ startTime avant le début de la lecture de l'acteur vidéo numérique.

Pour les pistes audio, cette propriété indique le moment de départ du son actuellement en cours ou en pause, lorsque ce son était placé en file d'attente. Elle 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 :

```
sprite(5).startTime = 100
```

Voir aussi

queue(), setPlaylist(), play() (audio)

startTimer

Syntaxe

```
startTimer
```

Description

Commande ; remet la propriété `timer` à 0 et réinitialise aussi tous les compteurs cumulatifs des fonctions `lastClick()`, `lastEvent()`, `lastKey` et `lastRoll`.

Si plusieurs compteurs sont requis, vous devez créer votre compteur et le gérer par vous-même. Ces propriétés peuvent s'appliquer à un comportement, une liste globale ou même un script parent. En général, la propriété `ticks` est utilisée pour surveiller le temps de cette manière.

Exemple

Le gestionnaire suivant réinitialise le compteur lorsque l'utilisateur appuie sur une touche :

```
on keyDown
  startTimer
end
```

Voir aussi

`lastClick()`, `lastEvent()`, `lastKey`, `lastRoll`, `timeoutLength`, `timeoutMouse`, `timeoutPlay`, `timeoutScript`, `timer`

state (3D)

Syntaxe

```
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 qui a été 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)

Syntaxe

```
member(quelActeur).state
state of member quelActeur
```

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 valide 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 :

```
on mouseDown
  if member("Ella Fitzgerald").state = 9 then
    alert "Aucun fichier audio n'a pu être trouvé."
  end if
end
```

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.

```
on exitFrame
  if member("Intro").percentStreamed < 100 then
    put "Etat de téléchargement :" && member("Intro").state
    go the frame
  end if
end
```

Voir aussi

`clearError`, `getError()`

state (RealMedia)

Syntaxe

`sprite(`*`quelleImageObjet`*`).state`

`member(`*`quelActeur`*`).state`

Description

Propriété d'acteur ou image-objet RealMedia ; renvoie l'état courant du train 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` 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 de Lingo sur un acteur. Si la valeur de `member.state` est inférieure à 2, il se peut que certaines propriétés Lingo soient incorrectes et, par conséquent, que les commandes Lingo 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 Lingo auront des valeurs bien définies qui pourront être utilisées pour effectuer des opérations Lingo 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 train 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 train 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 train 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 Lingo ont été actualisées à l'aide du train actuel. Lorsque la valeur de `state` est supérieure ou égale à 2, les propriétés `height`, `width` et `duration` du train 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 Lingo 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 train 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`.

Exemples

Les exemples suivants indiquent que l'état des trains de l'image-objet 2 et de l'acteur Real est 0, qui indique la fermeture.

```
put sprite(2).state
-- 0

put member("Real").state
-- 0
```

Voir aussi

`mediaStatus`, `percentBuffered`, `lastError`

static

Syntaxe

```
sprite(quelleImageObjetFlash).static
the static of sprite quelleImageObjetFlash
member(quelleImageObjetFlash).static
the static of member quelleImageObjetFlash
```

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 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 :

```
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
```

staticQuality

Syntaxe

```
staticQuality of sprite quelleImageObjetQTVR
```

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

Syntaxe

```
objetAudio.status
the status of objetAudio
```

Description

Propriété en lecture seule ; indique l'état du `numéroDePiste` de la piste audio. 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 :

```
put sound(2).status
```

Voir aussi

`isBusy()`, `pause() (lecture audio)`, `play() (audio)`, `stop() (audio)`

on stepFrame

Syntaxe

```
on stepFrame
  instruction(s)
end
```

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 Lingo 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 :

```
property monImageObjet

on new me, LimageObjet
  monImageObjet = LimageObjet
  return me
end

on stepFrame me
  sprite(monImageObjet).loc = point(random(640),random(480))
end
```

stillDown

Syntaxe

```
the stillDown
```

Description

Propriété système ; indique si l'utilisateur appuie sur le bouton de la souris (`TRUE`) ou non (`FALSE`).

Cette fonction est utile dans les scripts `mouseDown` pour ne déclencher certains événements qu'après l'exécution de la fonction `mouseUp`.

Lingo 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` :

```
if the stillDown then procédureDeGlissement
```

Voir aussi

`the mouseDown (propriété système)`

stop (Flash)

Syntaxe

```
sprite(quelleImageObjetFlash).stop()
stop sprite quelleImageObjetFlash
```

Description

Commande Flash ; interrompt une image-objet animation Flash lue dans l'image courante.

Exemple

Le script d'image suivant interrompt les images de l'animation Flash lues dans les pistes 5 à 10 :

```
on enterFrame
  repeat with i = 5 to 10
    sprite(i).stop()
  end repeat
end
```

Voir aussi

`hold`

stop (RealMedia)

Syntaxe

`sprite(`*`quelleImageObjet`*`).stop()`

`member(`*`quelActeur`*`).stop()`

Description

Méthode d'acteur ou d'image-objet RealMedia ; arrête la lecture du train et réinitialise `currentTime` à 0 et `percentBuffered` à 0. La valeur de `mediaStatus` devient `#stopped`.

L'appel de la commande `play` après l'appel de la commande `stop` requiert la remise en tampon du train, et la lecture démarre au début de ce train.

Exemples

Les exemples suivants arrêtent la lecture de l'image-objet 2 ou l'acteur Real.

```
sprite(2).stop()
```

```
member("Real").stop()
```

Voir aussi

`play (RealMedia)`, `pause (RealMedia)`, `stop (RealMedia)`, `mediaStatus`

stop() (audio)

Syntaxe

`sound(`*`numéroDePiste`*`).stop()`
`stop(sound(`*`numéroDePiste`*`))`

Description

Commande ; interrompt le son lu dans la piste spécifiée par *`numéroDePiste`*. L'émission d'une commande `play()` entame la lecture du premier des sons de la file d'attente de la piste audio donnée.

Vous pourrez voir un exemple de `stop() (son)` dans une animation en consultant l'animation Sound Control du dossier Learning/Lingo_Examples, lui-même dans le dossier de Director.

Exemple

L'instruction suivante arrête la lecture de l'acteur son lu dans la piste audio 1 :

```
sound(1).stop()
```

Voir aussi

`getPlaylist()`, `pause() (lecture audio)`, `play() (audio)`, `playNext()`, `rewind()`

stopEvent

Syntaxe

```
stopEvent
```

Description

Commande ; empêche Lingo de passer un message d'événement dans le reste de la hiérarchie de messages. Elle a le même effet que la commande `dontPassEvent` utilisée dans les versions précédentes de Director, mais s'applique également aux scripts d'images-objets.

Utilisez la commande `stopEvent` pour arrêter le message dans un script d'événement principal ou un script d'image-objet, rendant 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 commande `stopEvent` ne s'applique qu'à l'événement courant. 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.

Exemple

L'instruction suivante montre l'événement `mouseUp` interrompu dans un comportement lorsque la variable globale *`totalGénéral`* est égale à 500 :

```
global totalGénéral

on mouseUp me
  if totalGénéral = 500 then
    stopEvent
  end if
end
```

Les autres comportements ou les scripts ultérieurs ne reçoivent pas l'événement s'il est interrompu de cette manière.

Voir aussi

`pass`

stop member

Syntaxe

```
member(quelActeur ).stop()
stop member (quelActeur)
```

Description

Commande ; interrompt la lecture d'un acteur Shockwave Audio (SWA) en flux continu. Lorsque l'acteur est interrompu, la propriété `state` est égale à 0.

Pour changer des propriétés telles que `streamName`, `preLoadTime` et `URL`, la lecture de l'acteur SWA lu en flux continu doit être arrêtée.

Exemple

L'instruction suivante stoppe la lecture de l'acteur SWA Orchestre :

```
member("Orchestre").stop()
```

Voir aussi

`play member`, `pause member`

on stopMovie

Syntaxe

```
on stopMovie
  instruction(s)
end
```

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 efface les variables globales et ferme deux fichiers de ressources lorsque l'animation est interrompue :

```
global gScoreActuel
on stopMovie
  set gScoreActuel = 0
  closeResFile "Polices spéciales"
  closeResFile "Curseurs spéciaux"
end
```

Voir aussi

`on prepareMovie`

stopTime

Syntaxe

```
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
```

stream

Syntaxe

```
member(quelleImageObjetFlash).stream(nombreDoctets )
stream(member quelleImageObjetFlash, nombreDoctets)
```

Description

Commande ; permet de faire passer manuellement en mémoire une partie d'un acteur animation Flash spécifié. Vous avez la possibilité d'indiquer le nombre d'octets du train sous la forme d'un nombre entier. Si vous omettez le paramètre *nombreDoctets*, Director essaie de lire le nombre d'octets défini par la propriété `bufferSize` de l'acteur.

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`.

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 chargé 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.

```
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."
      updateStage
    else
      member("Ligne de message").text = "Les 32 000 octets ont été reçus."
    end if
    go the frame
  end if
end
```

streaming

Syntaxe

```
member(quelActeur).streaming
the streaming of member quelActeur
```

Description

Propriété d'acteur texte QuickTime. Lorsque définie sur `TRUE`, cette propriété permet la lecture immédiate de contenu QuickTime lu depuis Internet pendant le chargement de l'acteur sur l'ordinateur de l'utilisateur. Si la propriété est définie sur `FALSE`, l'acteur doit être entièrement chargé pour que la lecture puisse commencer.

Cette propriété est définie par défaut sur `TRUE` pour les animations créées dans la version 7.02 et les versions ultérieures de Director. Cette propriété est définie par défaut sur `FALSE` pour les animations créées dans les versions antérieures de Director.

Si l'acteur QuickTime contient une piste texte avec des points de repère, la piste texte doit être définie pour un préchargement si Director doit en utiliser les points de repère. Utilisez un éditeur vidéo pour définir la piste texte.

Exemple

L'instruction suivante définit la lecture en flux continu de l'acteur VidéoDeLeverDeSoleil sur `FALSE`, ce qui entraîne son téléchargement complet avant la lecture dans Shockwave ou ShockMachine :

```
member("VidéoDeLeverDeSoleil").streaming = 0
```

streamMode

Syntaxe

```
member(quelleActeurFlash).streamMode
the streamMode of member quelActeurFlash
```

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` :

```
on startMovie
  repeat with i = 1 to the number of members of castLib 1
    if member(i, 1).type = #flash then
      member(i, 1).streamMode = #manual
    end if
  end repeat
end
```

streamName

Syntaxe

```
member(quelActeur).streamName
the streamName of member quelActeur
```

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.

```
member("trainSWA").streamName = "monDisque/Sons/Orchestre.swa"
```

streamSize

Syntaxe

```
member(quelleImageObjetFlash).streamSize
the streamSize of member quelActeurFlash
```

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.

```
on exitFrame
  if member("Intro").percentStreamed < 100 then
    member("Ligne de message").text = string(member("Intro").bytesStreamed) \
    && "sur" && string(member("Intro").streamSize) \
    &&"octets ont été téléchargés pour l'instant."
    go to the frame
  end if
end
```

streamSize (3D)

Syntaxe

```
member(quelActeur).streamSize
```

Description

Propriété 3D ; permet d'obtenir la taille du train 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)`

on streamStatus

Syntaxe

```
on streamStatus URL, état, octetsTéléchargés, totalDesoctets, erreur
  instruction(s)
end
```

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 chargement en cours. Les valeurs possibles sont `Connecting` (connexion), `Started` (commencé), `InProgress` (en cours), `Complete` (terminé) et `Error` (erreur).
octetsTéléchargés	Affiche le nombre d'octets récupérés depuis le réseau.
totalDesOctets	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 si le téléchargement a abouti avec succès 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 :

```
on streamStatus URL, état, octetsTéléchargés, totalDesOctets
  if état = "Complete" then
    put URL && ". Téléchargement terminé."
  end if
end streamStatus
```

Voir aussi

`getStreamStatus()`, `tellStreamStatus()`

string()

Syntaxe

`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.

Exemples

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()

Syntaxe

`stringP(`*`expression`*`)`

Description

Fonction ; détermine si une expression est une chaîne (`TRUE`) ou non (`FALSE`).

Le *P* de `stringP` signifie *prédicat*.

Exemples

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()

strokeColor

Syntaxe

member(*quelActeur*).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_Examples, lui-même dans le dossier de Director.

Exemple

L'instruction suivante donne à la propriété strokeColor de l'acteur Trait la couleur rouge :

```
member("trait").strokeColor = rgb(255, 0, 0)
```

Voir aussi

color(), fillColor, endColor, backgroundColor

strokeWidth

Syntaxe

member(*quelActeur*).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_Examples, 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 :

```
member("Trait").strokeWidth = 10
```

style

Syntaxe

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

Syntaxe

```
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`

substituteFont

Syntaxe

```
réfDacteurTexte.substituteFont(policeDorigine, nouvellePolice)
substituteFont(réfDacteurTexte, policeDorigine, nouvellePolice)
```

Description

Commande d'acteur texte ; remplace toutes les occurrences de la *`policeDorigine`* par la *`nouvellePolice`* dans la *`réfDacteurTexte`*.

Exemple

Le script suivant vérifie si la police Bonneville est disponible dans un acteur texte et, dans la négative, la remplace par Arial :

```
property spriteNum

on beginSprite me
  acteurCourant = sprite(spriteNum).member
  if acteurCourant.missingFonts contains "Bonneville" then
    acteurCourant.substituteFont("Bonneville", "Arial")
  end if
end
```

Voir aussi

`missingFonts`

suspendUpdates

Syntaxe

`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.

swing()

Syntaxe

`quelleImageObjetQTVR``.swing(`*`pan, inclin, champDeVue, vitesseDeMouv`*`)`
`swing (`*`quelleImageObjetQTVR, 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.

- *`quelleImageObjetQTVR`* – Numéro de l'image-objet contenant l'acteur QuickTime VR.
- *`pan`* – Nouvelle position panoramique, en degrés.
- *`inclin`* – Nouvelle inclinaison, en degrés.
- *`champDeVue`* – Nouveau champ de vue, en degrés.
- *`vitesseDeMouv`* – Vitesse du mouvement. Spécifiez un nombre entier compris entre 1 et 10 (de lent à rapide).

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.

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°.

```
sprite(1).swing(300, -15, 40, 1)
```

Voir aussi

`pan (propriété QTVR)`

switchColorDepth

Syntaxe

```
the switchColorDepth
```

Description

Propriété système ; 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`). La valeur par défaut est `FALSE`.

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.

Cette propriété peut être testée et définie. La valeur par défaut correspond à la case d'option Adapter le moniteur aux couleurs de l'animation dans la boîte de dialogue Préférences générales.

Exemples

L'instruction suivante affecte à la variable intitulée *`commutateur`* le paramètre courant de `switchColorDepth` :

```
commutateur = the switchColorDepth
```

L'instruction suivante vérifie si le codage des couleurs courant est de 8 bits et, le cas échéant, active la propriété `switchColorDepth` :

```
if the colorDepth = 8 then the switchColorDepth = TRUE
```

Voir aussi

`colorDepth`

symbol()

Syntaxe

```
valeurDeChaîne.symbol
symbol(valeurDeChaîne)
```

Description

Fonction ; prend une chaîne, *valeurDeChaîne*, et renvoie un symbole.

Exemples

L'instruction suivante affiche le symbole #bonjour :

```
put ("bonjour").symbol
-- #bonjour
```

L'instruction suivante affiche le symbole #auRevoir :

```
x = "auRevoir"
put x.symbol
-- #auRevoir
```

Voir aussi

value(), string()

symbolP()

Syntaxe

```
expression.symbolP
symbolP(expression)
```

Description

Fonction ; détermine si l'expression spécifiée par *expression* est un symbole (TRUE) ou non (FALSE).

Le *P* de symbolP signifie *prédicat*.

Exemple

L'instruction suivante vérifie si la variable *maVariable* est un symbole :

```
put maVariable.symbolP
```

Voir aussi

ilk()

systemDate

Syntaxe

```
the systemDate
```

Description

Propriété système ; renvoie la date courante sous un format standard et peut être utilisée conjointement à d'autres opérations de date pour assurer la gestion des dates sous un format international ou sur d'autres plates-formes.

Les opérations mathématiques de la date sont exécutées en jours.

Cette propriété peut être testée, mais pas définie.

Exemple

Le script suivant affiche la date courante récupérée, puis détermine la date en comptant 90 jours à partir de la date courante :

```
on afficherLaFinDePériodeDéval
  set aujourdhui = the systemDate
  set dateDeFinDePériodeDéval = aujourdhui + 90
  put "Fin de la période d'évaluation :"&&dateDeFinDePériodeDéval
end
```

Voir aussi

`date() (horloge du système)`

TAB

Syntaxe

`TAB`

Description

Constante ; représente la touche Tab.

Exemples

L'instruction suivante vérifie si le caractère saisi est le caractère de tabulation et, le cas échéant, appelle le gestionnaire *champSuivant* :

```
if the key = TAB then champSuivant
```

Les instructions suivantes font avancer ou reculer la tête de lecture selon que l'utilisateur appuie sur les touches Tab ou Majuscule-Tab :

```
if the key = TAB then
  if the shiftDown then
    go the frame -1
  else
    go the frame +1
  end if
end if
```

Voir aussi

`BACKSPACE`, `EMPTY`, `RETURN (constante)`

tabCount

Syntaxe

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

Syntaxe

```
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]]
```

tan()

Syntaxe

```
tan(angle)
```

Description

Fonction mathématique ; renvoie la tangente de l'angle spécifié exprimée en radians sous forme de nombre à virgule flottante.

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`

target

Syntaxe

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 (propriété de temporisation), timeout(), timeoutHandler, timeoutList

targetFrameRate

Syntaxe

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

tell

Syntaxe

```
tell quelleFenêtre to instruction(s)
tell quelleFenêtre
  instruction(s)
end
```

Description

Commande ; communique des instructions à la fenêtre spécifiée par *quelleFenêtre*.

La commande `tell` est pratique pour faire dialoguer les animations. Elle peut être utilisée au sein d'une animation principale pour envoyer un message à une animation lue dans une fenêtre ou pour envoyer un message d'une animation dans une fenêtre en cours d'exécution à l'animation principale. Par exemple, la commande `tell` permet à un bouton du tableau de commande d'appeler un gestionnaire dans une animation lue dans une fenêtre. L'animation lue dans une fenêtre peut alors réagir au gestionnaire de la première animation en exécutant ce gestionnaire. L'animation lue dans la fenêtre peut ainsi dialoguer avec l'animation principale en lui renvoyant une valeur donnée.

Lorsque vous utilisez la commande `tell` pour envoyer un message à une animation lue dans une fenêtre, identifiez l'objet fenêtre en donnant son chemin complet ou son numéro dans `windowList`. Si vous utilisez `windowList`, utilisez l'expression `getAt(the windowList, numDeFenêtre)`, où *`numDeFenêtre`* est une variable contenant le numéro de la position de la fenêtre dans la liste. L'ouverture et la fermeture de fenêtres pouvant changer l'ordre de `windowList`, il est toujours judicieux de conserver le chemin complet comme variable globale pour faire référence aux fenêtres contenant l'expression `getAt` dans `windowList`.

Exemples

Une commande `tell` à plusieurs lignes ressemble à un gestionnaire et requiert une instruction `end tell` :

```
global animationEnfant

tell window animationEnfant
  go to frame "Intro"
  the stageColor = 100
  sprite(4).member = member "Diana Ross"
  updateStage
end tell
```

Lorsqu'un message appelle un gestionnaire, une valeur renvoyée par ce gestionnaire est présente dans la propriété globale `result` une fois que le gestionnaire appelé a terminé. Les instructions suivantes envoient à la fenêtre `animationEnfant` le message `calculerLeSolde`, puis renvoie le résultat :

```
global animationEnfant

tell window animationEnfant to calculerLeSolde
-- un nom de gestionnaire
monSolde = result()
-- renvoi de la valeur du gestionnaire calculerLeSolde
```

Lorsque vous utilisez la commande `tell` pour envoyer un message d'une animation lue dans une fenêtre à l'animation principale, utilisez la propriété système `stage` comme nom d'objet :

```
tell the stage to go frame "Main Menu"
```

Lorsque vous utilisez la commande `tell` pour appeler un gestionnaire situé dans une autre animation, assurez-vous qu'il n'en existe pas un du même nom dans le script utilisé par l'animation locale. Si cela était le cas, le script local serait exécuté. Cette restriction ne s'applique qu'aux gestionnaires placés dans le script à partir duquel vous utilisez la commande `tell`.

L'instruction suivante utilise la fenêtre Tableau de commande pour provoquer le branchement de l'animation Simulation sur une autre image :

```
tell window "Simulation" to go frame "Enregistrer"
```

tellStreamStatus()

Syntaxe

```
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.

Exemples

Le gestionnaire on prepareMovie suivant active le gestionnaire on streamStatus au démarrage de l'animation :

```
on prepareMovie
  tellStreamStatus(TRUE)
end
```

L'instruction suivante détermine l'état du gestionnaire d'état de la lecture en flux continu :

```
on mouseDown
  put tellStreamStatus()
end
```

Voir aussi

on streamStatus

tellTarget()

Syntaxe

```
sprite(quelleImageObjet).tellTarget("nomDeCible")
sprite(quelleImageObjet).endTellTarget()
```

Définition

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.

Exemples

La commande suivante définit le clip comme cible :

```
sprite(1).tellTarget("\monClip")
```

La commande suivante arrête le clip :

```
sprite(1).stop()
```

La commande suivante entraîne la lecture du clip :

```
sprite(1).play()
```

La commande suivante entraîne le retour au scénario principal :

```
sprite(1).endTellTarget()
```

La commande suivante arrête l'animation principale :

```
sprite(1).stop()
```

tension

Syntaxe

```
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

Syntaxe

```
member(quelActeur).text
the text of member quelActeur
```

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 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_Examples, lui-même dans le dossier de Director.

Exemples

L'instruction suivante place Merci dans l'acteur vide Réponse :

```
if member("Réponse").text = EMPTY then
  member("Réponse").text = "Merci."
```

L'instruction suivante affecte à l'acteur Remarque la phrase « Vous avez pris la bonne décision ! » :

```
member("Remarque").text = "Vous avez pris la bonne décision !"
```

Voir aussi

`selEnd`, `selStart`

texture

Syntaxe

```
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(quelleRessource\
  DeModèleDeSystèmeDeParticules).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 une liste complète des propriétés de texture dans Lingo 3D par fonction.

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

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).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

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).\
  face[indexDeFace].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

Syntaxe

```
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

Syntaxe

```
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

Syntaxe

```
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

Syntaxe

```
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

Syntaxe

```
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]` est appliquée à l'espace de placage avant la génération des coordonnées de texture dans l'espace du modèle. Avec une `wrapTransform[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[indexDeCoucheDeTexture]` est appliquée à l'espace de placage avant la génération des coordonnées de texture dans l'espace du modèle. Avec une `wrapTransform List[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[indexDeCoucheDeTexture]` est appliquée à l'espace de placage avant la génération des coordonnées de texture dans l'espace du modèle. Avec une `wrapTransform List[indexDeCoucheDeTexture]`, 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 positif 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

Syntaxe

```
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 (i.e. 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 (i.e. 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 pour améliorer la fidélité des couleurs ou pour faire tenir un plus grand nombre de textures sur la carte. Le même espace peut contenir deux fois plus de textures 16 bits que de textures 32 bits. Director passe au rendu logiciel lorsqu'une animation utilise plus de textures que la carte ne peut en contenir.

Vous pouvez spécifier n'importe laquelle les 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

Syntaxe

```
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

Syntaxe

```
référenceDeMatériau.textureRepeatList[indexDeCoucheDeTexture]
member(quelActeur).shader(quelMatériau).textureRepeatList\
  [ indexDeCoucheDeTexture ]
member(quelActeur).shader[indexDeListeDeMatériaux].textureRepeatList\
  [ indexDeCoucheDeTexture ]
member(quelActeur).model(quelModèle).shader.textureRepeatList\
  [ indexDeCoucheDeTexture ]
member(quelActeur).model(quelModèle).shaderList\
  [indexDeListeDeMatériaux].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 `FALSE`, la texture est appliquée à une portion plus réduite des surfaces plutôt que juxtaposée lorsque `réfDeMatériau.textureTransform [indexDeCoucheDeTexture].scale` est une valeur 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[texture LayerIndex].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

Syntaxe

```
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 y est ignoré.

Pour décaler l'image de `point(décalageX,décalageY)`, utilisez `référenceDeMatériau.textureTransform.translate(décalageX,décalageY,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 `réfDeMatériau.texture Transform.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`.

Exemples

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

Syntaxe

```
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(décalageX,décalageY)`, utilisez `textureTransformList [quelleCoucheDeTexture].translate(décalageX,décalageY,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`.

Exemples

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

Syntaxe

`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`

the

Syntaxe

`the` *propriété*

Description

Mot-clé ; doit précéder un grand nombre de fonctions et toutes les propriétés Lingo rédigées en syntaxe verbose. Ce mot-clé permet également de distinguer la propriété ou la fonction d'un nom de variable ou d'objet.

Dans les versions précédentes de Director, vous deviez obligatoirement utiliser le mot-clé `the` pour exprimer les propriétés d'acteurs et d'images-objets. Cette syntaxe est toujours acceptée comme alternative.

Les propriétés sont disponibles globalement pour les gestionnaires, même sans déclaration globale. Comme les variables globales, les propriétés système Lingo restent disponibles d'une animation à l'autre au sein de la même présentation. Les propriétés d'images-objets changent à chaque fois qu'une nouvelle animation est chargée.

thumbnail

Syntaxe

```
member(quelActeur).thumbnail
the thumbnail of member quelActeur
```

Description

Propriété d'acteur ; contient l'image utilisée pour afficher l'aperçu d'un acteur dans la fenêtre Distribution. Cette image peut être personnalisée pour n'importe quel acteur.

Cette propriété peut être testée et définie uniquement pendant la programmation.

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 :

```
member("Repère d'emplacement").picture = member(10).thumbnail
```

Voir aussi

`picture (propriété d'acteur)`

ticks

Syntaxe

```
the ticks
```

Description

Propriété système ; renvoie la position temporelle actuelle en battements (1 battement = 1/60ème de seconde). Le compte des battements commence dès le démarrage de l'ordinateur.

Exemple

L'instruction suivante convertit les battements en secondes et en minutes en divisant deux fois le nombre de battements par 60, puis affecte le résultat à la variable *minutesAllumé* :

```
secondesCourantes = the ticks/60
minuteCourantes = secondesCourantes/60
```

Voir aussi

`time()`, `timer`, `milliseconds`

tilt

Syntaxe

```
tilt of sprite (quelleImageObjetQTVR)
```

Description

Propriété d'image-objet QuickTime VR ; inclinaison courante, en degrés, de l'animation QuickTime VR.

Cette propriété peut être testée et définie.

time()

Syntaxe

```
the time
the short time
the long time
the abbreviated time
the abbrev time
the abbr time
```

Description

Fonction ; renvoie l'heure courante fournie par l'horloge du système sous la forme d'une chaîne pouvant prendre trois formats : `short` (court), `long` ou `abbreviated` (abrégé). Si aucun format n'est précisé, le format par défaut est le format `short`. Le format abrégé peut également être abrégé en `abbrev` et `abbr`. En France, les formats `short` et `abbreviated` sont les mêmes.

Exemple

Les instructions suivantes permettent d'afficher l'heure sous différents formats dans la fenêtre Messages. Les résultats possibles apparaissent sous chaque instruction.

```
put the short time
--"11:15"
put the long time
--"11:15:15"
put the abbreviated time
--"11:15"
```

Les trois formats varient en fonction du format utilisé par l'ordinateur. Les exemples donnés ci-dessus concernent la France.

Voir aussi

`date() (horloge du système)`

time (propriété d'objet de temporisation)

Syntaxe

`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.

Remarquez qu'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`

timeout()

Syntaxe

`timeout("`*`nomDeTemporisation`*`")`

Description

Fonction ; renvoie l'objet de temporisation appelé *`nomDeTemporisation`*. Utilisez `timeout("nom").new` pour ajouter un nouvel objet de temporisation à la liste `the timeoutList`. Pour plus d'informations, consultez `new()`.

Exemple

Le gestionnaire suivant supprime l'objet de temporisation appelé Foudre aléatoire :

```
on exitFrame
  timeout("Foudre aléatoire").forget()
end
```

Voir aussi

`forget()`, `new()`, `timeoutHandler`, `timeoutList`

on timeOut

Syntaxe

```
on timeOut
  instruction(s)
end
```

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 `the 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 la durée de temps définie dans la propriété `timeoutLength`. Il peut être utilisé pour répondre lorsque les utilisateurs quittent l'ordinateur.

```
on timeOut
  play movie "Boucle"
end timeOut
```

Voir aussi

`timeoutScript`, `timeoutLength`

timeoutHandler

Syntaxe

`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`

timeoutKeyDown

Syntaxe

```
the timeoutKeyDown
```

Description

Propriété système ; détermine si les événements `keyDown` remettent la propriété `timeoutLapsed` à zéro (`TRUE`, valeur par défaut) ou non (`FALSE`). Cette propriété est très pratique pour reprendre le compte à rebours d'une temporisation à chaque fois que l'utilisateur appuie sur une touche.

Cette propriété peut être testée et définie.

Exemples

L'instruction suivante affecte à la variable *`Minutage`* la valeur de la propriété `timeoutKeyDown` :

```
Minutage= timeoutKeyDown
```

L'instruction suivante désactive la propriété `timeoutKeyDown` :

```
timeoutKeyDown = FALSE
```

Voir aussi

`keyDownScript`

timeoutLapsed

Syntaxe

```
the timeoutLapsed
```

Description

Propriété système ; indique le nombre de battements écoulés depuis la dernière temporisation. Un événement de temporisation survient lorsque la propriété `timeoutLapsed` atteint le délai spécifié par la propriété `timeoutLength`.

La propriété `timeoutLapsed` peut être testée et définie.

Exemple

L'instruction suivante attribue au champ CompteArebours la valeur de la propriété `timeoutLapsed`. La division de la propriété `timeoutLapsed` par 60 convertit la valeur en secondes.

```
member("CompteArebours").text = string(the timeoutLapsed / 60)
```

timeoutLength

Syntaxe

```
the timeoutLength
```

Description

Propriété système ; détermine le nombre de battements au terme duquel une temporisation se produit. L'événement de temporisation survient lorsque la propriété `timeoutLapsed` atteint le délai spécifié par la propriété `timeoutLength`.

Cette propriété peut être testée et définie. La valeur par défaut est de 10 800 battements, soit 3 minutes.

Exemple

L'instruction suivante fixe `timeoutLength` sur 10 secondes :

```
set the timeoutLength to 10 * 60
```

– ou –

```
the timeoutLength = 10 * 60
```

timeoutList

Syntaxe

```
the timeoutList
```

Description

Propriété système ; liste linéaire contenant tous les objets de temporisation actuellement actifs. Utilisez la fonction `forget()` pour supprimer un objet de temporisation.

Les objets de temporisations sont ajoutés à la liste `timeoutList` avec la fonction `new()`.

Exemple

L'instruction suivante supprime le troisième objet de temporisation de la liste `timeoutList` :

```
the timeoutList[3].forget()
```

Voir aussi

`forget()`, `new()`, `timeout()`, `target`, `timeoutHandler`

timeoutMouse

Syntaxe

```
the timeoutMouse
```

Description

Propriété système ; détermine si les événements `mouseDown` remettent la propriété `timeoutLapsed` à zéro (`TRUE`, valeur par défaut) ou non (`FALSE`).

Cette propriété peut être testée et définie.

Exemples

L'instruction suivante enregistre le paramètre courant de `timeoutMouse` en affectant à la variable *`Minutage`* la valeur `the timeoutMouse` :

```
Minutage = the timeoutMouse
```

L'instruction suivante affecte à la propriété `timeoutMouse` la valeur `FALSE`. Il en résulte que la propriété `timeoutLapsed` conserve sa valeur lorsque le bouton de la souris est enfoncé.

```
the timeoutMouse = FALSE
```

Voir aussi

`mouseDownScript`, `mouseUpScript`

timeoutPlay

Syntaxe

```
the timeoutPlay
```

Description

Propriété système ; détermine si la propriété `timeoutLapsed` sera définie sur `TRUE` lorsque l'animation sera mise en pause par l'intermédiaire de la commande `pause`. Si elle est définie sur `TRUE`, les temporisations surviendront lors de la mise en pause de l'animation. Si elle est définie sur `FALSE`, les temporisations ne surviendront pas lors de la mise en pause de l'animation. La valeur par défaut est `FALSE`.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante affecte à `timeoutPlay` la valeur `TRUE`, ce qui indique à Lingo de remettre la propriété `timeoutLapsed` à 0 après la lecture d'une animation :

```
set the timeoutPlay to TRUE
```

– ou –

```
the timeoutPlay = TRUE
```

timeoutScript

Syntaxe

```
the timeoutScript
```

Description

Propriété système ; détermine les éléments Lingo exécutés par Director comme gestionnaires d'événements principaux lors d'une temporisation. L'instruction Lingo est rédigée sous forme de chaîne encadrée de guillemets droits. La valeur par défaut est `EMPTY`.

Pour définir un gestionnaire d'événement principal pour les temporisations, définissez `timeoutScript` comme une chaîne d'éléments Lingo appropriés : une simple instruction Lingo ou une instruction d'appel à un gestionnaire. Si le script d'événement que vous avez affecté n'est plus valable, désactivez-le au moyen de l'instruction `set the timeoutScript to EMPTY`.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante affecte `timeoutScript` à un script d'appel du gestionnaire *`procédureDeTemporisation`* :

```
the timeoutScript = "procédureDeTemporisation"
```

Voir aussi

`on timeOut`

timer

Syntaxe

```
the timer
```

Description

Propriété système ; permet de compter le temps en nombre de battements (1 battement = 1/60ème de secondes). Cette propriété n'a aucun rapport avec la propriété `timeoutScript`. Elle sert uniquement à minuter certains événements. La commande `startTimer` remet le compteur à 0.

La propriété `timer` est pratique pour déterminer le temps écoulé depuis l'émission de la commande `startTimer`. Par exemple, vous pouvez utiliser `timer` pour synchroniser des images et une piste audio en insérant un délai qui suspend l'exécution de l'animation jusqu'à ce qu'une certaine durée se soit écoulée.

Exemples

Le comportement de script d'image suivant crée un délai de 2 secondes :

```
on beginSprite
  startTimer
end
on exitFrame
  if (the timer < 60 * 2) then go the frame
end
```

L'instruction suivante affecte à la variable *`topDépart`* la valeur `timer` :

```
topDépart = the timer
```

Voir aussi

`lastClick()`, `lastEvent()`, `lastKey`, `lastRoll`, `startTimer`

timeScale

Syntaxe

```
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

Syntaxe

```
window (quelleFenêtre.title)
the title of window quelleFenêtre
```

Description

Propriété de fenêtre ; affecte un titre à la fenêtre spécifiée par *`quelleFenêtre`*.

Cette propriété peut être testée et définie pour les fenêtres autres que la scène.

Exemple

L'instruction suivante donne le titre Action à la fenêtre X :

```
window("X").title = "Action"
```

titleVisible

Syntaxe

```
window (quelleFenêtre.titleVisible)
the titleVisible of window quelleFenêtre
```

Description

Propriété de fenêtre ; indique si la fenêtre spécifiée par *quelleFenêtre* contient un titre dans sa barre de titre.

Cette propriété peut être testée et définie pour les fenêtres autres que la scène.

Exemple

L'instruction suivante affiche le titre de la fenêtre Tableau_de_commande en affectant à la propriété `titleVisible` de la fenêtre la valeur `TRUE` :

```
window("Tableau_de_commande").titleVisible = TRUE
```

to

Le mot `to` entre dans plusieurs constructions de Lingo.

Voir aussi

`char...of`, `item...of`, `line...of`, `word...of`, `repeat with`, `set...to`, `set...=`

toon (modificateur)

Syntaxe

```
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 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 au 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 au 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

Syntaxe

```
sprite(quelleImageObjet).top
the top of sprite quelleImageObjet
```

Description

Propriété d'image-objet ; renvoie la coordonnée verticale du bord supérieur du rectangle délimitant l'image-objet spécifiée par *quelleImageObjet*, en pixels à partir du coin supérieur gauche de la scène.

Lorsqu'une animation est lue en tant qu'applet, la valeur de cette propriété est déterminée par rapport au coin supérieur gauche de l'applet.

Cette propriété peut être testée et définie.

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` :

```
if sprite(3).top < 0 then débordementEnHaut
```

Voir aussi

`bottom`, `height`, `locH`, `left`, `locV`, `right`, `width`

top (3D)

Syntaxe

```
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`.

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

Syntaxe

```
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.

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

Syntaxe

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.

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
```

topSpacing

Syntaxe

expressionSousChaîne.topSpacing

Description

Propriété d'acteur texte ; permet de spécifier tout espacement supplémentaire applicable au haut de chaque paragraphe dans la partie *expressionDeSousChaîne* 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

trace

Syntaxe

```
the trace
```

Description

Propriété d'animation ; indique si la fonction de suivi de l'animation est activée (`TRUE`) ou non (`FALSE`). Lorsque la fonction de suivi est activée, la fenêtre Messages affiche chaque ligne Lingo exécutée.

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante active la propriété `trace` :

```
the trace = TRUE
```

Voir aussi

`traceLogFile`

traceLoad

Syntaxe

```
the traceLoad
```

Description

Propriété d'animation ; indique la quantité d'informations sur les acteurs affichée pendant leur chargement, de la manière suivante :

- 0 – N'affiche aucune information.
- 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).

La valeur par défaut de la propriété `traceLoad` est 0 ; cette propriété peut être lue et modifiée.

Exemple

L'instruction suivante entraîne l'affichage des noms des acteurs lorsqu'ils sont chargés :

```
the traceLoad = 1
```

traceLogFile

Syntaxe

```
the traceLogFile
```

Description

Propriété système ; spécifie le nom du fichier dans lequel le contenu de la fenêtre Messages est enregistré. Vous pouvez fermer ce fichier en affectant à la propriété `traceLogFile` la valeur `EMPTY` (""). 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.

Exemples

L'instruction suivante demande à Lingo d'écrire le contenu de la fenêtre Messages dans le fichier Messages.txt dans le dossier de l'animation courante :

```
the traceLogFile = the moviePath & "Messages.txt"
```

L'instruction suivante ferme le fichier dans lequel est écrit le contenu de la fenêtre Messages :

```
the traceLogFile = ""
```

trackCount (propriété d'acteur)

Syntaxe

```
member(quelActeur).trackCount()
trackCount(member quelActeur)
```

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 :

```
put member("Chronique Jazz").trackCount()
```

trackCount (propriété d'image-objet)

Syntaxe

```
sprite(quelleImageObjetVidéoNumérique).trackCount()
trackCount(sprite quelleImageObjet)
```

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 :

```
put sprite(10).trackCount()
```

trackEnabled

Syntaxe

```
sprite(quelleImageObjetVidéoNumérique).trackEnabled(quellePiste)
trackEnabled(sprite quelleImageObjet, 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 :

```
put sprite(1).trackEnabled(2)
```

Voir aussi

`setTrackEnabled`

trackNextKeyTime

Syntaxe

```
sprite(quelleImageObjetVidéoNumérique).trackNextKeyTime(quellePiste)
trackNextKeyTime(sprite quelleImageObjet, 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 :

```
put sprite(15).trackNextKeyTime(5)
```

trackNextSampleTime

Syntaxe

```
sprite(quelleImageObjetVidéoNumérique).trackNextSampleTime(quellePiste)
trackNextSampleTime(sprite quelleImageObjet, 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 :

```
put sprite(15).trackNextSampleTime(5)
```

trackPreviousKeyTime

Syntaxe

```
sprite(quelleImageObjetVidéoNumérique).trackPreviousKeyTime(quellePiste)
trackPreviousKeyTime(sprite quelleImageObjet, 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 :

```
put sprite(2).trackPreviousKeyTime(1)
```

trackPreviousSampleTime

Syntaxe

```
sprite(quelleImageObjetVidéoNumérique).trackPreviousSampleTime(quellePiste)
trackPreviousSampleTime(sprite quelleImageObjet, 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 :

```
put sprite(15).trackPreviousSampleTime(5)
```

trackStartTime (propriété d'acteur)

Syntaxe

```
member(quelActeurVidéoNumérique).trackStartTime(quellePiste)
trackStartTime(member quelActeur, 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 :

```
put member("Chronique Jazz").trackStartTime(5)
```

trackStartTime (propriété d'image-objet)

Syntaxe

```
sprite(quelleImageObjetVidéoNumérique).trackStartTime(quellePiste)
trackStartTime(sprite quelleImageObjet, 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).

```
put sprite(10).trackStartTime(5)
-- 120
```

Voir aussi

`duration`, `movieRate`, `movieTime`

trackStopTime (propriété d'acteur)

Syntaxe

```
member(quelActeurVidéoNumérique).trackStopTime(quellePiste)
trackStopTime(member quelActeur, 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 :

```
put member("Chronique Jazz").trackStopTime(5)
```

trackStopTime (propriété d'image-objet)

Syntaxe

```
sprite(quelleImageObjetVidéoNumérique).trackStopTime(quellePiste)
trackStopTime(sprite, quelleImageObjet, 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 :

```
put sprite(6).trackStopTime(5)
```

Voir aussi

`movieRate`, `movieTime`, `trackStartTime (propriété d'acteur)`

trackText

Syntaxe

```
sprite(quelleImageObjetVidéoNumérique).trackText(quellePiste)
trackText(sprite quelleImageObjet, 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 :

```
member("Archives").text = string(sprite(20).trackText(5))
```

trackType (propriété d'acteur)

Syntaxe

```
member(quelActeurVidéoNumérique).trackType(quellePiste)
trackType(member quelActeur, 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 *formatTexte* :

```
on vérifDeTexte
  if member("Nouvelles du jour").trackType(5) = #text then formatTexte
end
```

trackType (propriété d'image-objet)

Syntaxe

```
sprite(quelleImageObjetVidéoNumérique).trackType(quellePiste)
trackType(sprite quelleImageObjet, 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 *formatTexte* :

```
on vérifDeTexte
  if sprite(10).trackType(5) = #text then formatTexte
end
```

trails

Syntaxe

```
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, celle-ci doit être asservie.

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 (commande)

Syntaxe

```
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.

Exemples

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)`

transform (propriété)

Syntaxe

```
member(quelActeur).node(quelNœud).transform
member(quelActeur).node(quelNœud).transform.propriétéDe\
  Transformation
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 Univers, 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
```

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( -94.1448, 119.0128, 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

Syntaxe

```
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
```

translate

Syntaxe

```
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 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.

Le paramètre facultatif *parRapportA* détermine les axes du système de coordonnées utilisés pour appliquer les modifications de translation. Le paramètre *parRapportA* peut avoir les valeurs suivantes :

- `#self` applique les incréments en fonction du système de coordonnées local du nœud (les axes x, y et 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.

Exemples

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 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("séquence").model("Mars")
gbModel.transform = t
put gbModèle.transform.position
-- vector( 100.0000, 0.0000, 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("Séquence").model("Bip").position
--vector(-38.5000, 21.2500, 2.0000)
member("Séquence").model("Bip").translate(20, 10, -0.5)
put member("Séquence").model("Bip").position
--vector(-18.5000, 31.2500, 1.5000)
```

Voir aussi

`transform (propriété)`, `preTranslate()`, `scale (commande)`, `rotate`

translation

Syntaxe

```
member(quelActeurQuickTime).translation
the translation of member quelActeurQuickTime
sprite(quelleImageObjetQuickTime).translation
the translation of sprite quelleImageObjetQuickTime
```

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 la propriété `center` est réglée sur `TRUE`, l'image-objet est décalée par rapport au centre du cadre de délimitation ; lorsque la propriété `center` est réglée sur `FALSE`, l'image-objet est décalée par rapport au coin supérieur gauche du cadre 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.

```
on exitFrame
  positionHorizontale = sprite(5).translation[1]
  if positionHorizontale < 320 then
    sprite(5).translation = sprite(5).translation + [10, 0]
    go the frame
  end if
end
```

transparent

Syntaxe

```
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 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

Syntaxe

```
sprite(quelleImageObjetQTVR).triggerCallback
triggerCallback of sprite quelleImageObjetQTVR
```

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.

```
property pMonNuméroDimageObjet, spriteNum

on beginSprite me
  pMonNuméroDimageObjet = me.spriteNum
  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
```

trimWhiteSpace (propriété)

Syntaxe

`member(`*`quelActeur`*`).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 dans le volet Bitmap de l'inspecteur des propriétés.

trimWhitespace() (fonction)

Syntaxe

`objetImage``.trimWhitespace()`

Description

Fonction ; supprime tous les pixels blancs situés en dehors du rectangle de délimitation minimal et renvoie le résultat dans un nouvel objet image.

Exemple

L'instruction suivante supprime l'espace blanc de l'acteur Fleur et renvoie le nouvel objet image, recadré, dans la variable *imageRecadrée* :

```
imageRecadrée = member("fleur").image.trimWhitespace()
```

Voir aussi

crop() (commande d'acteur)

TRUE

Syntaxe

```
TRUE
```

Description

Constante ; représente la valeur d'une expression logique, telle que 2 < 3. Elle est traditionnellement une valeur numérique de 1, mais tout entier non nul donne TRUE dans une comparaison.

Exemple

L'instruction suivante active la propriété soundEnabled en lui donnant la valeur TRUE :

```
the soundEnabled = TRUE
```

Voir aussi

FALSE, if

tunnelDepth

Syntaxe

```
member(quelActeurTexte).tunnelDepth
member(quelActeur).modelResource(quelleRessource\
  deModèleExtrudeur).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

Syntaxe

```
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

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).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 = \
  #velocity
```

Voir aussi

`type (lumière)`

type (propriété d'acteur)

Syntaxe

```
member(quelActeur).type
the type of member quelActeur
member( quelActeur, quelleDistrib). type
member quelActeur of castLib quelleDistrib.type
the type of member quelActeur of castLib quelleDistrib
```

Description

Propriété d'acteur ; indique le type de l'acteur spécifié. Cette propriété remplace la propriété `castType`, utilisée dans les versions précédentes de Director.

La propriété d'acteur `type` peut avoir l'une des valeurs suivantes :

`#animgif`	`#palette`
`#bitmap`	`#picture`
`#button`	`#QuickTimeMedia`
`#cursor`	`#script`
`#digitalVideo`	`#shape`
`#empty`	`#shockwave3D`
`#field`	`#sound`
`#filmLoop`	`#swa`
`#flash`	`#text (#richText est désormais obsolète)`
`#font`	`#transition`
`#movie`	`#vectorShape`
`#ole`	

Cette liste comprend les types d'acteurs disponibles dans Director et les Xtras l'accompagnant. Vous pouvez également définir des types d'acteurs spéciaux correspondant pour des acteurs personnalisés.

Lorsqu'une animation est exécutée sous la forme d'applet, la propriété d'acteur `type` est uniquement valide pour les types d'acteurs supportés par le lecteur.

Pour les animations créées sous Director 5 et 6, la propriété d'acteur `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.

Cette propriété peut être testée, mais pas définie.

Exemple

Le gestionnaire suivant vérifie si l'acteur Nouvelles du jour est un acteur champ et, dans la négative, affiche un message d'alerte :

```
on vérifDuFormat
  if member("Nouvelles du jour").type <> #field then alert \
  "Désolé, cet acteur doit être un champ."
end
```

type (lumière)

Syntaxe

```
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 (ressource de modèle)

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).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)

Syntaxe

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.

Exemples

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)

Syntaxe

```
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.

Exemples

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 (propriété d'image-objet)

Syntaxe

```
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)

Syntaxe

```
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`

union()

Syntaxe

```
rect(1).union(rect(2))
union (rect1, rect2)
```

Description

Fonction ; renvoie le plus petit rectangle qui inclut les deux rectangles *rect1* et *rect2*.

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)
```

– ou –

```
put rect(0, 0, 10, 10).union(rect(15, 15, 20, 20))
--rect (0, 0, 20, 20)
```

Voir aussi

map(), rect()

unLoad

Syntaxe

```
unLoad
unLoad numéroDimage
unLoad numéroDimageInitiale, numéroDimageFinale
```

Description

Commande ; 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 commandes `preLoad` ou le chargement normal des acteurs.

- Si vous ne précisez pas d'argument, la commande `unLoad` purge de la mémoire les acteurs figurant dans toutes les images d'une animation.
- Si vous utilisez un argument, *numéroDimage*, la commande `unLoad` purge de la mémoire les acteurs contenus dans cette image.
- Si vous utilisez deux arguments, *numéroDimageInitiale* et *numéroDimageFinale*, la commande `unLoad` purge de la mémoire tous les acteurs contenus dans la plage spécifiée. Vous pouvez spécifier une plage d'images par numéros ou par noms.

Exemples

L'instruction suivante supprime de la mémoire les acteurs utilisés dans l'image 10 :

```
unLoad 10
```

L'instruction suivante purge les acteurs compris entre l'image intitulée Première et celle intitulée Dernière :

```
unLoad "Première", "Dernière"
```

Voir aussi

preLoad (commande), preLoadMember, unLoadMember, purgePriority

unLoadMember

Syntaxe

```
unLoadMember
member(quelActeur).unLoad()
unLoadMember member quelActeur
member(quelActeur, quelleDistribution).unLoad()
unLoadMember member quelActeur of castLib quelleDistribution
member(premierActeur).unLoad(dernierActeur)
unLoadMember member premierActeur, dernierActeur
```

Description

Commande ; force Director à purger les acteurs spécifiés de la mémoire. Director purge automatiquement les acteurs les moins récemment utilisés pour permettre l'utilisation des commandes `preLoad` ou le chargement normal des acteurs.

- Si elle est utilisée sans argument, la commande `unLoadMember` purge de la mémoire les acteurs présents dans toutes les images d'une animation.
- Si elle est utilisée avec l'argument *`quelActeur`* et *`quelleDistribution`*, la commande `unLoadMember` purge de la mémoire l'acteur portant le nom ou le numéro spécifié.
- Si vous utilisez deux arguments, *`premierActeur`* et *`dernierActeur`*, la commande `unLoadMember` purge de la mémoire tous les acteurs contenus dans la plage spécifiée.

Si elle est utilisée dans une nouvelle animation sans acteurs chargés, cette commande renvoie une erreur.

Les acteurs qui ont été modifiés au cours de la programmation ou par la définition de `picture`, `pasteClipBoardInto`, et ainsi de suite, ne peuvent pas être purgés.

Exemples

L'instruction suivante purge de la mémoire l'acteur Ecran1 :

```
unLoadMember member "Ecran1"
```

– ou –

```
member("Ecran1").unload()
```

L'instruction suivante purge de la mémoire tous les acteurs compris entre l'acteur 1 et l'acteur Cinémascope :

```
unLoadMember 1, member "Cinémascope"
```

– ou –

```
member(1).unload("Cinémascope")
```

Voir aussi

preLoad (commande), preLoadMember, purgePriority

unloadMovie

Syntaxe

```
unloadMovie quelleAnimation
```

Description

Commande ; purge 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.

Exemples

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 :

```
if (the freeBlock < (100 * 1024)) then unLoadMovie "Parsifal"
```

L'instruction suivante purge l'animation à http://www.cbDemille.com/SunsetBlvd.dir :

```
unLoadMovie "http://www.cbDemille.com/SunsetBlvd.dir"
```

unregisterAllEvents

Syntaxe

```
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.

Exemple

L'instruction suivante annule l'enregistrement de l'acteur Séquence pour toutes les notifications d'événements.

```
member("Séquence").unregisterAllEvents()
```

Voir aussi

```
registerForEvent()
```

update

Syntaxe

```
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 Lingo.

updateFrame

Syntaxe

```
updateFrame
```

Description

Commande ; 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 commande `updateFrame` pour chaque image que vous mettez à jour.

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 Lingo atteint la fin de la boucle de répétition. Le nombre des images est déterminé par l'argument `nombreDimages`.

```
on animBalle nombreDimages
  beginRecording
    horizontal = 0
    vertical = 300
    repeat with i = 1 to nombreDimages
      go to frame i
      sprite(20).memberNum = member("Balle").number
      sprite(20).locH = horizontal
      sprite(20).locV = vertical
      sprite(20).type = 1
      sprite(20).foreColor = 255
      horizontal = horizontal + 3
      vertical = vertical + 2
      updateFrame
    end repeat
  endRecording
end
```

Voir aussi

`beginRecording`, `endRecording`, `scriptNum`, `tweened`

updateLock

Syntaxe

```
the updateLock
```

Description

Propriété d'animation ; détermine si la scène est mise à jour pendant l'enregistrement du scénario (`FALSE`) ou non (`TRUE`).

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 Lingo 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.

updateMovieEnabled

Syntaxe

```
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
```

updateStage

Syntaxe

```
updateStage
```

Description

Commande ; rafraîchit la scène immédiatement au lieu de la rafraîchir entre les images.

La commande `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`).

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 :

```
on déplacementVersLaDroite quelleImageObjet, quelleDistance
  sprite(quelleImageObjet).locH = sprite(quelleImageObjet).locH +
  quelleDistance
  updateStage
end déplacementVersLaDroite
```

URL

Syntaxe

```
member(quelActeur).URL
the URL of member quelActeur
```

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 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. :

```
on mouseDown
  member("Benny G.").URL = "http://audio.mm.com/exemples/classic.swa"
end
```

URLEncode

Syntaxe

`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.

Utilisez le paramètre facultatif *`chaîneOSduServeur`* pour encoder tout caractère renvoyé dans *`listeDePropriétésOuChaîne`*. La valeur est par défaut `"Unix"` mais peut recevoir `"Win"` ou `"Mac"` comme valeur et convertit les retours chariot de l'argument *`listeDePropriétésOuChaîne`* 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 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`

useAlpha

Syntaxe

```
member(quelActeur).useAlpha
objetImage.useAlpha
```

Description

Propriété d'acteur bitmap et 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.

```
member("Premier plan").useAlpha=not member("Premier plan").useAlpha
```

useDiffuseWithTexture

Syntaxe

```
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 aux 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, blendFunction List, blendSource, blendSourceList, blendConstant et blendConstantList.

```
member("Niveau2").model("Mystère").shaderlist.useDiffuseWith\
  Texture = TRUE
```

Voir aussi

blendFunction, blendConstant

useFastQuads

Syntaxe

```
the useFastQuads
```

Description

Propriété globale ; 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. La méthode de calcul de quadrilatères par défaut de Director, plus lente, fournit des résultats visuellement plus attractifs, lorsque vous utilisez les quadrilatères pour des effets de distorsion et d'autres effets aléatoires. La valeur par défaut est `FALSE`.

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 `the 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 :

```
the useFastQuads = TRUE
```

Voir aussi

`quad`

useHypertextStyles

Syntaxe

```
member(quelActeurTexte).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 :

```
on mouseUp
  member("monTexte").usehypertextStyles = not
  member("monTexte").usehypertextStyles
end
```

useLineOffset

Syntaxe

```
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

Syntaxe

```
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.

Exemples

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

Syntaxe

```
the userName
```

Description

Propriété d'animation ; renvoie une chaîne comprenant le nom d'utilisateur saisi pendant l'installation de Director.

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.

```
on prepareMovie
  chaîneAffichée = the userName
  put RETURN&the organizationName after chaîneAffichée
  put RETURN&the serialNumber after chaîneAffichée
  member("Infos utilisateur").text = chaîneAffichée
end
```

Voir aussi

organizationName, serialNumber, window

userName (RealMedia)

Syntaxe

```
sprite(quelleImageObjet).userName
member(quelActeur).userName
sprite(quelleImageObjet).userName = nomDutilisateur
member(quelActeur).userName = nomDutilisateur
```

Description

Propriété d'acteur et image-objet RealMedia ; permet de définir le nom d'utilisateur nécessaire à l'accès à un train RealMedia protégé. Vous ne pouvez pas utiliser cette propriété pour récupérer un nom d'utilisateur spécifié auparavant. Si aucun nom d'utilisateur n'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.

Exemples

Les exemples suivants indiquent que le nom d'utilisateur pour le train RealMedia de l'acteur Real ou de l'image-objet 2 a été défini.

```
put sprite(2).userName
-- "********"

put member("Real").userName
-- "********"
```

Les exemples suivants indiquent que le nom d'utilisateur pour le train RealMedia de l'acteur Real ou de l'image-objet 2 n'a jamais été défini.

```
put sprite(2).userName
-- ""

put member("Real").userName
-- ""
```

Les exemples suivants indiquent que le nom d'utilisateur pour le train RealMedia de l'acteur Real et de l'image-objet 2 est Marcel.

```
member("Real").userName = "Marcel"

sprite(2).userName = "Marcel"
```

Voir aussi

`password`

useTargetFrameRate

Syntaxe

`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`

value()

Syntaxe

`value(`*`expressionChaîne`*`)`

Description

Fonction ; renvoie la valeur d'une chaîne. Cette chaîne peut être constituée de n'importe quelle expression reconnue par Lingo. 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()`.

Exemples

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 Message 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()

Syntaxe

```
vector(x, y, z)
```

Description

Type de données et fonction 3D ; un vecteur décrit un point dans l'espace 3D en fonction des paramètres *x*, *y* et *z*, qui sont les distances spécifiques des points de référence le long des axes *x*, *y* et *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 fonction renvoie un objet vecteur.

Les valeurs des vecteurs peuvent être manipulées à l'aide des opérateurs `+`, `-`, `*` et `/`. Consultez la définition des différents opérateurs pour plus d'informations.

Exemples

L'instruction suivante crée un vecteur et l'affecte à la variable *monVecteur* :

```
monVecteur = vector(10.0, -5.0, 0.0)
```

L'instruction suivante ajoute deux vecteurs et affecte la valeur résultante à la variable *ceVecteur* :

```
ceVecteur = vector(1.0, 0.0, 0.0) + vector(0.0, -12.5, 2.0)
put ceVecteur
--vector(1.0000, -12.5000, 2.0000)
```

Voir aussi

`+ (addition) (3D)`, `- (soustraction)`, `* (multiplication)`, `/ (division) (3D)`

version

Syntaxe

```
version
```

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 version
```

vertex

Syntaxe

```
member(quelActeurFormeVectorielle).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.

Exemples

Le code suivant présente la définition du nombre de points de sommet dans un acteur :

```
put member("Archie").vertex.count
-- 2
```

Pour obtenir le second sommet de l'acteur, vous pouvez utiliser la syntaxe suivante :

```
put member("Archie").vertex[2]
-- point(66.0000, -5.0000)
```

Vous pouvez également définir la valeur d'une poignée de contrôle :

```
member("Archie").vertex[2].handle1 = point(-63.0000, -16.0000)
```

Voir aussi

vertexList

vertexList

Syntaxe

```
member(quelActeurFormeVectorielle).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, 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 :

```
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)

Syntaxe

`member(`*`quelActeur`*`).modelResource(`*`quelleRessDeMod`*`).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)

Syntaxe

```
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

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).\
  face[indexDeFace].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)

Syntaxe

```
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)

Syntaxe

```
sprite(quelleImageObjet).video
member(quelActeur).video
```

Description

Propriété RealMedia ; permet de savoir ou de définir si l'image-objet ou l'acteur rend la vidéo (`TRUE`) ou seulement l'audio (`FALSE`). 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 pour activer/désactiver la vidéo pendant la lecture.

Exemples

Les exemples suivants indiquent que la propriété vidéo de l'image-objet 2 et de l'acteur Real a pour valeur `TRUE`.

```
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`.

```
sprite(2).video = FALSE

member("Real").video = FALSE
```

videoForWindowsPresent

Syntaxe

```
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

Syntaxe

```
sprite(quelleImageObjetFlashOuFormeVectorielle).viewH
the viewH of sprite quelleImageObjetFlashOuFormeVectorielle
member(quelActeurFlashOuFormeVectorielle).viewH
the viewH of member quelActeurFlashOuFormeVectorielle
```

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 :

```
on panoramiqueDroite quelleImageObjet
  repeat with i = 120 down to -120
    sprite(quelleImageObjet).viewH = i
    updateStage
  end repeat
end
```

Voir aussi

scaleMode, viewV, viewPoint, viewScale

viewPoint

Syntaxe

```
sprite(quelleImageObjetFlashOuFormeVectorielle).viewPoint
the viewPoint of sprite quelleImageObjetFlashOuFormeVectorielle
member(quelActeurFlashOuFormeVectorielle).viewPoint
the viewPoint of member quelActeurFlashOuFormeVectorielle
```

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 :

```
on panoramiqueTransversal quelleImageObjet
  repeat with i = 1 to 10
    sprite(quelleImageObjet).viewPoint = sprite(quelleImageObjet).\
  viewPoint + point(i * -5, i * -5)
    updateStage
  end repeat
end
```

Voir aussi

`scaleMode`, `viewV`, `viewH`, `viewScale`

viewScale

Syntaxe

```
sprite(quelleImageObjetFlashOuFormeVectorielle).viewScale
the viewScale of sprite quelleImageObjetFlashOuFormeVectorielle
member(quelActeurFlashOuFormeVectorielle).viewScale
the viewScale of member quelActeurFlashOuFormeVectorielle
```

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 :

```
on beginSprite me
  sprite(spriteNum of me).viewScale = 200
end
```

Voir aussi

`scaleMode`, `viewV`, `viewPoint`, `viewH`

viewV

Syntaxe

```
sprite(quelleImageObjetFlashOuFormeVectorielle).viewV
the viewV of sprite quelleImageObjetFlashOuFormeVectorielle
member(quelActeurFlashOuFormeVectorielle).viewV
the viewV of member quelActeurFlashOuFormeVectorielle
```

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 :

```
on panoramiqueVersLeBas quelleImageObjet
  repeat with i = 120 down to -120
    sprite(quelleImageObjet).viewV = i
    updateStage
  end repeat
end
```

Voir aussi

`scaleMode`, `viewV`, `viewPoint`, `viewH`

visible (propriété d'image-objet)

Syntaxe

```
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
```

visible (propriété de fenêtre)

Syntaxe

```
window quelleFenêtre.visible
the visible of window quelleFenêtre
```

Description

Propriété de fenêtre ; détermine si la fenêtre spécifiée par *quelleFenêtre* est visible (TRUE) ou non (FALSE).

Cette propriété peut être testée et définie.

Exemple

L'instruction suivante rend la fenêtre Tableau_de_commande visible :

```
window("Tableau_de_commande").visible = TRUE
```

visibility

Syntaxe

```
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
```

voiceCount()

Syntaxe

```
voiceCount()
```

Description

Fonction : renvoie le nombre de voix installées disponibles pour la fonction 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.

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()

Syntaxe

```
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.

Exemples

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()

Syntaxe

```
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 :

- `#nom` 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.

Exemples

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()

Syntaxe

```
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.

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 :

```
if voiceGetPitch() > 10 then
  voiceSetPitch(10)
end if
```

Voir aussi

`voiceSpeak()`, `voicePause()`, `voiceResume()`, `voiceStop()`, `voiceGetRate()`, `voiceSetRate()`, `voiceSetPitch()`, `voiceGetVolume()`, `voiceSetVolume()`, `voiceState()`, `voiceWordPos()`

voiceGetRate()

Syntaxe

`voiceGetRate()`

Description

Fonction ; renvoie la cadence de lecture courante 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 et du logiciel vocal. Il s'agit généralement de valeurs comprises entre -10 et 10.

Exemple

Les instructions suivantes vérifient que la cadence de synthèse vocale est inférieure à 50 et la définit à 50 le cas échéant :

```
if voiceGetRate() < 50 then
  voiceSetRate(50)
end if
```

Voir aussi

`voiceSpeak()`, `voicePause()`, `voiceResume()`, `voiceStop()`, `voiceSetRate()`, `voiceGetPitch()`, `voiceSetPitch()`, `voiceGetVolume()`, `voiceSetVolume()`, `voiceState()`, `voiceWordPos()`

voiceGetVolume()

Syntaxe

`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.

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 :

```
if voiceGetVolume() < 55 then
  voiceSetVolume(55)
end if
```

Voir aussi

`voiceSpeak()`, `voicePause()`, `voiceResume()`, `voiceStop()`, `voiceGetRate()`, `voiceSetRate()`, `voiceGetPitch()`, `voiceSetPitch()`, `voiceSetVolume()`, `voiceState()`, `voiceWordPos()`

voiceInitialize()

Syntaxe

```
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.

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. » :

```
err = voiceInitialize()
if err = 1 then
  voiceSpeak("Bienvenue dans Shockwave")
else
  alert "Le chargement du logiciel vocal a échoué."
end if
```

Voir aussi

`voiceCount()`, `voiceSet()`, `voiceGet()`

voicePause()

Syntaxe

```
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.

Exemple

Les instructions suivantes provoquent l'interruption de la fonction de conversion de texte en voix lorsque l'utilisateur clique avec la souris :

```
on mouseUp
  voicePause()
end mouseUp
```

Voir aussi

`voiceSpeak()`, `voiceResume()`, `voiceStop()`, `voiceGetRate()`, `voiceSetRate()`, `voiceGetPitch()`, `voiceSetPitch()`, `voiceGetVolume()`, `voiceSetVolume()`, `voiceState()`, `voiceWordPos()`

voiceResume()

Syntaxe

```
voiceResume()
```

Description

Commande ; reprend la sortie vocale. La commande renvoie la valeur 1 en cas de réussite, et 0 dans le cas contraire.

Exemple

Les instructions suivantes réactivent la fonction vocale lorsque la tête de lecture se déplace sur l'image suivante dans le scénario :

```
on exitFrame
  voiceResume()
end exitFrame
```

Voir aussi

`voiceSpeak()`, `voicePause()`, `voiceStop()`, `voiceGetRate()`, `voiceSetRate()`, `voiceGetPitch()`, `voiceSetPitch()`, `voiceGetVolume()`, `voiceSetVolume()`, `voiceState()`, `voiceWordPos()`

voiceSet()

Syntaxe

`voiceSet(`*`entier`*`)`

Description

Commande : définit la voix courante de la fonction de conversion de texte en voix. La valeur spécifiée doit être un nombre entier. La plage des valeurs valides dépend du nombre de voix installées sur l'ordinateur de l'utilisateur. Si une autre valeur est spécifiée, la voix sera définie sur la valeur admise la plus proche. Dans ce cas, la commande renvoie la nouvelle valeur définie. Utilisez `voiceCount()` pour déterminer le nombre de voix disponibles.

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()

Syntaxe

`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. 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()

Syntaxe

`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. 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 autre valeur 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()

Syntaxe

`voiceSetVolume(`*`entier`*`)`

Description

Commande ; définit le volume courant de la fonction de conversion de texte en voix. La valeur spécifiée doit être un nombre entier. 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()

Syntaxe

`voiceSpeak("`*`chaîne`*`")`

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.

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()

Syntaxe

```
voiceState()
```

Description

Fonction ; renvoie l'état actuel de la voix sous forme d'un symbole. Les valeurs possibles sont `#playing`, `#paused` et `#stopped`.

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 :

```
if voiceState() <> #playing then
  voiceSet(1)
end if
```

Voir aussi

`voiceSpeak()`, `voicePause()`, `voiceResume()`, `voiceStop()`, `voiceGetRate()`, `voiceSetRate()`, `voiceGetPitch()`, `voiceSetPitch()`, `voiceGetVolume()`, `voiceSetVolume()`, `voiceWordPos()`, `voiceSpeak()`

voiceStop()

Syntaxe

```
voiceStop()
```

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.

Exemple

Les instructions suivantes arrêtent la fonction vocale lorsque la tête de lecture se déplace sur l'image suivante dans le scénario :

```
on exitFrame
  voiceStop()
end exitFrame
```

Voir aussi

`voiceSpeak()`, `voicePause()`, `voiceResume()`, `voiceGetRate()`, `voiceSetRate()`, `voiceGetPitch()`, `voiceSetPitch()`, `voiceGetVolume()`, `voiceSetVolume()`, `voiceState()`, `voiceWordPos()`, `voiceSpeak()`

voiceWordPos()

Syntaxe

```
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.

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.

```
voiceSpeak("Bonjour, comment allez-vous ?")
put voiceWordPos()
-- 1
```

Voir aussi

`voiceSpeak()`, `voicePause()`, `voiceResume()`, `voiceStop()`, `voiceGetRate()`, `voiceSetRate()`, `voiceGetPitch()`, `voiceSetPitch()`, `voiceGetVolume()`, `voiceSetVolume()`, `voiceState()`, `voiceSpeak()`

VOID

Syntaxe

```
VOID
```

Description

Constante ; indique la valeur `VOID`.

Exemple

L'instruction suivante vérifie si la variable *`variableCourante`* a la valeur `VOID` :

```
if variableCourante = VOID then
  put "Cette variable n'a aucune valeur"
end if
```

Voir aussi

`voidP()`

voidP()

Syntaxe

`voidP(`*`nomDeVariable`*`)`

Description

Fonction ; détermine si la variable spécifiée par *`nomDeVariable`* 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`.

Exemple

L'instruction suivante vérifie si la variable *réponse* a une valeur initiale :

```
put voidP(réponse)
```

Voir aussi

ilk(), VOID

volume (propriété d'acteur)

Syntaxe

```
member(quelActeur).volume
the volume of member quelActeur
```

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 :

```
member("fichierSWA").volume = 128
```

volume (piste audio)

Syntaxe

```
sound(quellePiste).volume
the volume of sound numéroDePiste
```

Description

Propriété système ; détermine le volume de la piste audio spécifiée par *quellePiste*. Les pistes audio sont numérotées 1, 2, 3, etc. 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, 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.

Cette propriété ne supporte pas la syntaxe à point. Utilisez la présentation de la syntaxe exposée ici.

Vous pourrez voir un exemple de volume (piste audio) dans une animation en consultant l'animation Sound Control du dossier Learning/Lingo_Examples, 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 :

```
sound(2).volume = 130
```

Voir aussi

`fadeIn()`, `fadeOut()`, `soundEnabled`, `soundLevel`, `fadeTo()`, `pan (propriété audio)`

volume (propriété d'image-objet)

Syntaxe

```
sprite(quelleImageObjet).volume
the volume of sprite quelleImageObjet
```

Description

Propriété d'image-objet ; contrôle le volume d'un acteur animation vidéo numérique 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 :

```
sprite(7).volume = 256
```

Voir aussi

`soundLevel`

warpMode

Syntaxe

```
sprite(quelleImageObjetQTVR).warpMode
warpMode of sprite quelleImageObjetQTVR
```

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` :

```
sprite(1).warpMode = #full
```

width

Syntaxe

```
member(quelActeur).width
the width of member quelActeur
objetImage.width
sprite(quelleImageObjet).width
the width of sprite quelleImageObjet
```

Description

Propriété d'acteur, d'objet image et d'image-objet ; pour les acteurs forme vectorielle, Flash, GIF animé, bitmap et forme, détermine la largeur, en pixels, de l'acteur spécifié par *quelActeur*, *objetImage* ou *quelleImageObjet*. Les acteurs champ et bouton ne sont pas affectés.

Pour les acteurs et les objets image, cette propriété peut être testée ; pour les images-objets, elle peut être testée et définie.

La définition de cette propriété pour les images-objets bitmap définit automatiquement la propriété `stretch` de l'image-objet sur `TRUE`.

Exemples

L'instruction suivante affecte la largeur de l'acteur 50 à la variable *hauteur* :

```
hauteur = member(50).width
```

L'instruction suivante règle la largeur de l'image-objet 10 sur 26 pixels :

```
sprite(10).width = 26
```

L'instruction suivante affecte la largeur de l'image-objet numéro i + 1 à la variable *largeur* :

```
largeur = sprite(i + 1).width
```

Voir aussi

`height`

width (3D)

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).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

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).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

Syntaxe

```
member(quelActeur).modelResource(quelleRessDeMod).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

Syntaxe

```
window quelleFenêtre
```

Description

Mot-clé ; fait référence à la fenêtre d'animation, une fenêtre contenant une animation Director, spécifiée par *quelleFenêtre*.

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.

Exemples

L'instruction suivante ouvre une fenêtre intitulée Navigation :

```
open window "Navigation"
```

L'instruction suivante place la fenêtre Navigation au premier plan :

```
moveToFront window "Navigation"
```

– ou –

```
window("Navigation").moveToFront()
```

Voir aussi

`close window`, `moveToBack`, `moveToFront`, `open window`

windowList

Syntaxe

```
the windowList
```

Description

Propriété système ; affiche une liste des références à toutes les fenêtres d'animation recensées.

Exemples

L'instruction suivante affiche une liste de toutes les fenêtres d'animation recensées dans la fenêtre Messages :

```
put the windowList
```

L'instruction suivante efface la liste `windowList` :

```
the windowList = [ ]
```

Voir aussi

`windowPresent()`

windowPresent()

Syntaxe

`windowPresent(`*`nomDeFenêtre`*`)`

Description

Fonction ; indique si l'objet spécifié par *`nomDeFenêtre`* est exécuté sous la forme d'une animation dans une fenêtre (`TRUE`) ou non (`FALSE`). Si une fenêtre a été ouverte, la valeur de `windowPresent` reste `TRUE` pour cette fenêtre, jusqu'à ce qu'elle soit supprimée de la propriété `windowList`.

L'argument *`nomDeFenêtre`* doit correspondre au nom de la fenêtre tel qu'il apparaît dans la propriété `windowList`.

Exemple

L'instruction suivante vérifie si l'objet maFenêtre est une animation dans une fenêtre et affiche le résultat dans la fenêtre Messages :

```
put windowPresent(maFenêtre)
```

Voir aussi

`windowList`

windowType

Syntaxe

```
window quelleFenêtre.windowType
the windowType of window quelleFenêtre
```

Description

Propriété de fenêtre ; contrôle le style d'affichage de la fenêtre spécifiée par *quelleFenêtre*, de la manière suivante :

- 0 – Fenêtre déplaçable et redimensionnable, sans case de zoom.
- 1 – Boîte de dialogue d'alerte ou modale
- 2 – Boîte simple, sans barre de titre
- 3 – Boîte simple ombrée, sans barre de titre
- 4 – Fenêtre déplaçable, sans case de redimensionnement ou de zoom
- 5 – Boîte de dialogue modale déplaçable
- 8 – Fenêtre de document standard
- 12 – Fenêtre avec case de zoom, ne pouvant pas être redimensionnée
- 16 – Fenêtre avec angles arrondis
- 49 – Palette flottante, en phase auteur (dans les projections Macintosh, la valeur 49 spécifie une fenêtre statique)

Vous pouvez définir cette propriété avant d'ouvrir la fenêtre. Les numéros 6, 7, 9, 10, 11, 13, 14, 15 et 17 à 48 n'ont aucun effet lorsque utilisés comme valeur windowType.

Vous pouvez changer le paramètre windowType après l'ouverture d'une fenêtre, mais un retard risque de se produire pendant le rafraîchissement de la fenêtre.

Si aucun paramètre windowType n'est spécifié, la valeur 0 est utilisée.

Sous Microsoft Windows, ces chiffres créent des fenêtres avec les mêmes fonctionnalités, mais avec une apparence Windows. D'autres valeurs sont utilisables pour windowType, mais doivent être utilisées avec prudence, certaines fenêtres modales ne pouvant être fermées qu'en redémarrant l'ordinateur.

Exemple

L'instruction suivante donne la valeur 8 au style d'affichage de la fenêtre Tableau de commande :

```
window("Tableau de commande").windowType = 8
```

word...of

Syntaxe

```
member(quelActeur).word[quelMot]
expressionActeurTexte.word[quelMot]
expressionSousChaîne.word[quelMot]
word quelMot of variableChampOuChaîne
variableChampOuChaîne.word[quelMot]
expressionActeurTexte.word[premierMot..dernierMot]
member(quelActeur).word[premierMot..dernierMot]
word premierMot to dernierMot of expressionSousChaîne
expressionSousChaîne.word[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 pourrez voir un exemple de `word...of` dans une animation en consultant l'animation Text du dossier Learning/Lingo_Examples, lui-même dans le dossier de Director.

Exemples

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)`

wordWrap

Syntaxe

```
member(quelActeur).wordWrap
the wordWrap of member quelActeur
```

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 :

```
member("Pierre").wordWrap = FALSE
```

worldPosition

Syntaxe

```
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(-1333.2097, 0.0000, -211.0973)
```

Voir aussi

`getWorldTransform()`, `position (transformation)`

worldSpaceToSpriteSpace

Syntaxe

```
member(quelActeur).camera(quelleCaméra).worldSpaceToSprite\
  Space(vecteur)
```

Description

Commande 3D ; renvoie le point du cadre de la caméra auquel la position, relative à l'univers, spécifiée par *vecteur*, 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`.

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.

```
put sprite(5).camera.worldSpaceToSpriteSpace(vector(0, 0, 0))
-- point(250, 281)
```

Voir aussi

`spriteSpaceToWorldSpace`, `rect (caméra)`

worldTransform

Syntaxe

```
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 *`transformFinale`* :

```
transformFinale =
  member("3D").model("biped").bonesPlayer.bone[3].worldTransform
```

Voir aussi

`bone`, `getWorldTransform()`, `transform (propriété)`

wrapTransform

Syntaxe

```
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 même n'est pas affectée ; la transformation ne modifie que l'orientation de la texture appliquée par le matériau.

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

Syntaxe

```
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 (propriété de vecteur)

Syntaxe

```
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

Syntaxe

```
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( 1.0000, 0.0000, 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

Syntaxe

```
xtra quelXtra
```

Description

Fonction ; renvoie une instance de l'Xtra de script spécifié par *quelXtra*.

Vous pourrez voir un exemple de `xtra` dans une animation en consultant l'animation Read and Write Text du dossier Learning/Lingo_Examples, lui-même dans le dossier de Director.

Exemple

L'instruction suivante utilise la fonction `new()` pour créer une nouvelle instance de l'Xtra Multiuser et l'affecte à la variable *outil* :

```
outil = new(xtra "Multiuser")
```

Voir aussi

`new()`

xtraList

Syntaxe

```
the xtraList
```

Description

Propriété système ; affiche une liste linéaire des propriétés de tous les Xtras disponibles et des versions de leurs fichiers. 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` :

#name	Spécifie le nom de fichier de l'Xtra sur la plate-forme courante. Il est possible que la liste ne contienne pas d'entrée `#name`, 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.

Exemples

L'instruction suivante affiche la liste `xtraList` dans la fenêtre Messages :

```
put the xtraList
```

Le gestionnaire suivant vérifie la version d'un Xtra donné :

```
-- ce gestionnaire vérifie tous les Xtras disponibles
-- pour renvoyer la version de l'Xtra requis
-- le NomDeFichierXtra peut ne présenter qu'une correspondance partielle
-- pour une utilisation plus précise, utiliser le nom de fichier intégral
-- la chaîne renvoyée correspond soit à la version de l'Xtra,
-- soit à une chaîne vide
-- elle peut être vide s'il n'existe aucun numéro de version ou
-- si l'Xtra est introuvable dans la liste disponible
on obtenirLaVersionDeLxtra nomDeFichierXtra
  -- obtenir la liste intégrale des Xtras et leurs informations
  listeDesXtras = the xtraList
  -- initialiser la variable locale pour qu'elle contienne la version
  laVersion = ""
  -- recommencer pour tous les Xtras énumérés
  repeat with XtraCourant in listeDesXtras
    -- si le nom de l'Xtra courant contient l'Xtra introduit,
    -- vérifier la version
    if XtraCourant.name contains nomDeFichierXtra then
      -- déterminer d'abord si la propriété de version existe pour cet Xtra
      indicateurDeVersion = getaProp(XtraCourant, #version)
      -- si la propriété de version n'a pas la valeur VOID,
      -- régler la variable locale sur
  -- la chaîne contenue dans cette valeur de propriété
      if not voidP(indicateurDeVersion) then
        laVersion = XtraCourant.version
      end if
    end if
  end repeat
--  renvoyer les informations de version trouvées ou une chaîne vide
  return laVersion
end
```

Voir aussi

`movieXtraList`, `getaProp`

xtras

Consultez

`number of xtras`

y (propriété de vecteur)

Syntaxe

```
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

Syntaxe

```
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 *y* 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(0.0000, 1.0000, 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

Syntaxe

```
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 tronc de cône 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 (propriété de vecteur)

Syntaxe

```
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

Syntaxe

```
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( 1.0000, 0.0000, 0.0000 )
member("Moteur").model("cylindreMod").rotate(0, 90, 0)
put member("Moteur").model("cylindreMod").transform.zAxis
--vector(0.0000, 0.0000, -1.0000)
```

zoomBox

Syntaxe

```
zoomBox imageObjetSource, imageObjetDestination {,délaiEnBattements}
```

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 démarre au rectangle délimitant l'*imageObjetSource* et se termine au rectangle délimitant l'*imageObjetDestination*. 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*.

La variable *délaiEnBattements* correspond au 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 :

```
zoomBox 7, 3
```

on zoomWindow

Syntaxe

```
on zoomWindow
  instruction(s)
end
```

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

Le gestionnaire suivant déplace l'image-objet 3 vers les coordonnées stockées dans la variable *emplacementCentral* lorsque la fenêtre lue par l'animation est redimensionnée :

```
on zoomWindow
  emplacementCentral = point(10, 10)
  sprite(3).loc = emplacementCentral
end
```

Voir aussi

`drawRect`, `sourceRect`, `on resizeWindow`

INDEX

Symboles

" (constante de chaîne) 56
" (guillemets) 557
(définition de symbole, opérateur) 45, 551
#engraver, Lingo pour le matériau 38
#inker
    couleurs 158
    modificateur 39
#newsprint, Lingo pour le matériau 38
#painter, Lingo pour le matériau 38
#toon
    couleurs 158
    modificateur 39
& (concaténation, opérateur) 47
&& (concaténation, opérateur) 48
() (parenthèses, opérateur) 49
* (multiplication, opérateur) 49
+ (addition, opérateur) 50
- (opérateur de soustraction) 46
-- (séparateur de commentaires) 47
/ (barre oblique) 51
/ (division, opérateur) 51
< (inférieur à, opérateur) 52
<= (inférieur ou égal à, opérateur) 52
<> (différent de, opérateur) 53
= (égal à, opérateur) 53
> (supérieur à, opérateur) 53
>= (supérieur ou égal à, opérateur) 54
@ (chemin d'accès, opérateur) 57
[] (crochets de listes) 54
\ (continuation, symbole) 57
¬ (continuation, symbole) 56

Chiffres

3D
    détermination des capacités avec Lingo 43
    Lingo pour le texte 43

A

abbreviated, fonction horaire 705
acteurs
    acteurs vides 252
    affectation aux images-objets 396
    affectés à la première piste audio 270
    affectés à la seconde piste audio 270
    affichage d'informations
        pendant le chargement 716
    annulation du chargement 126
    bordures 366
    castLibNum 129
    changement des acteurs utilisés
        comme curseurs 180
    chargement 253, 317, 318, 319
    collage du contenu du Presse-papiers 493
    copie 167, 216, 391, 505
    création 450
    cursor, commande 180
    dans la fenêtre Distribution 620
    date de modification 413
    définition des couleurs d'arrière-plan 87
    déplacement 431
    dernier acteur d'une distribution 470
    durée des 216
    flux continu, Lingo pour 37
    hauteur 304
    importation de fichiers 330
    interligne 365
    largeur 769
    liés 246
    lignes dans 364
    locToCharPos, fonction 371
    locVToLinePos, fonction 373
    média dans les pistes d'acteurs 721
    media, fonction d'acteur 391, 505
    member, mot-clé 394
    modification 219, 412

modification des noms d'acteurs 440
modified by, propriété 412
modified, fonction d'acteur 412
noms 440
numéros d'acteurs 252, 465
palettes associées 486, 487
pause dans la lecture 499
pictureP, fonction 505
police utilisée pour l'affichage 261
préchargement 318, 528, 530
priorité de purge 547
purge de la mémoire 547, 737, 738
remplacement du contenu 330
Shockwave Audio 278, 280
suppression 231
taille 642
téléchargement depuis Internet 392
texte du script affecté 613
vérification du chargement en mémoire 369
vides 252
zone de texte 113
acteurs champ
chaînes 692
défilement 613, 614
field, mot-clé 244
hauteur des lignes 365
installation de menus définis dans 336
lineHeight, fonction 365
lineHeight, propriété d'acteur 365
locToCharPos, fonction 371
locVToLinePos, fonction 373
ombres portées 214
position des lignes 366
style de police 262
taille de la police 261
zone de texte 113
acteurs transition
affectés à l'image courante 272
durée des 216
propriétés, chunkSize of member 139
acteurs vidéo numérique
activation/désactivation de la lecture 753, 757
compte des pistes 717
échelle temporelle 201
média dans les pistes 721
PausedAtStart, propriété d'acteur 497
position temporelle d'arrêt des pistes 720
position temporelle de début des pistes 719
préchargement en mémoire 528
activation du rapport de l'état de la lecture en flux continu 691
activation ou désactivation du son 648
activation/désactivation de la lecture des acteurs vidéo numérique 753, 757
addition (+), opérateur 50
affectation
acteurs aux images-objets 396
de noms aux fenêtres 442
palettes à des acteurs 486, 487
AIFF, fichiers 652
ajout
aux listes de propriétés 68
aux listes linéaires 63, 68, 80
alert, commande 69
alertHook, gestionnaire d'événement 227
alignment, propriété de champ 71
allowZooming, propriété d'animation 73
Alt, touche (Windows) 478
ancêtre, envoi de messages à 123
and, opérateur logique 76
animation, Lingo pour 33
animations
arrêt de la lecture 189, 303
dernière image 358
durée 437
enregistrement 604, 741
espaces inutilisés 434
go loop, commande 299
inactivité 316
indication de la version de création 435
insertion d'images 334
lecture à partir d'un repère 299, 300
lecture avec la commande play 507, 512
liées 612
mode d'exécution 600
nombre de distributions 465
nombre de menus 470
noms des animations 433, 436, 495
on stopMovie, gestionnaire d'événement 680
passage aux images 297
pause 163, 496, 500
préchargement 531
propriétés, Lingo pour 43
purge de la mémoire 739
recherche 286
réglages du codage des couleurs (Macintosh) 685
réponses aux événements 404
scénario associé 609
Shockwave 238, 239, 301

suppression d'images 192
taille de fichier 435
vitesse de lecture 437
Xtras disponibles pour 470
animations MIAW 247, 264, 689, 771
masquer 81
réduction 81
annulation des opérations réseau 445
applications
démarrage 476
shutDown, commande 641
arrêt
lecture d'une animation 189, 303
lecture des acteurs 675
lecture des sons 647, 652
messages d'événements 476
arrière-plan, traitement des événements 169
arrondir les nombres à virgule flottante 259
ASCII, codes 135, 472
astérisque (*), opérateur 49
augmentation progressive du son 649

B

BACKSPACE, constante de caractère 90
barre oblique (/) 51
battements
conversion du temps 705
lastClick, fonction 357
lastKey, fonction 358
lastRoll, fonction 358
movieTime, propriété d'image-objet 437
nombre de battements avant une temporisation 708
ticks, fonction 705
timeoutLength, propriété 708
timer, propriété 711
bip sonore, éléments
alert, commande 69
beep, commande 90
beepOn, propriété 91
bitmap, acteurs
codage de couleurs 197
palettes associées 487
point d'alignement 573
taille 642
bitmaps
compression 329, 435, 436
picture, propriété d'acteur 504, 505
traitement des espaces vierges 728
boîtes
Lingo pour 41
pour les acteurs champ 386
bonesPlayer, modificateur 40
bordures
acteurs champ 110
des acteurs forme 366
boucles
loop, mot-clé 376
next repeat, mot-clé 460
repeat with, mot-clé 580
repeat with...down to, mot-clé 581
repeat with...in list, mot-clé 582
boucles de répétition
exit repeat, mot-clé 236
next repeat, mot-clé 460
repeat with, mot-clé 580
repeat with...down to, mot-clé 581
repeat with...in list, mot-clé 582
boutons
buttonStyle, propriété 118
buttonType, propriété 119
images-objets de boutons radio 308
boutons radio, propriété checkBoxAccess 136
branchement
case, mot-clé 127
end case, mot-clé 225
if...then, instructions 320
otherwise, mot-clé 483
repeat while, mot-clé 580
brouillard, Lingo pour 33

C

cache
rafraîchissement du contenu des pages web 120
suppression dans les navigateurs web 140
taille dans les navigateurs web 121
cadence
affectée aux images 271
paramètres 271
pistes des cadences comme esclaves 545
caméras
ajout 65
Lingo pour 34
canonique 777, 779, 780
caractères, recherche dans les acteurs champ 371, 373
cases à cocher
checkBoxAccess, propriété 136
checkBoxType, propriété 137
hilite, propriété d'acteur 308
images-objets de cases à cocher 308
CGI, requête 284
chaînes
affichage dans la fenêtre du navigateur 449

ASCII, codes 135, 472
caractère initial dans les sélections 621
char...of, mot-clé 133
chars, fonction 134
comparaison 162
compte des caractères 470
compte des éléments 467
compte des lignes 467
compte des mots dans les expressions de sous-chaînes 469
conversion des expressions 681
dans les acteurs champ 692
date de la dernière modification 447
do, commande 208
écritures dans des fichiers 632
EMPTY, constante de caractères 222
envoi aux navigateurs web 237
et guillemets droits 56
expressions sous forme de 681
field, mot-clé 244
integer, fonction 336
item...of, mot-clé 345
last, fonction 356
length, fonction 360
line...of, mot-clé 363
offset, fonction de chaîne 474
put...after, commande 548
put...before, commande 549
put...into, commande 549
sélection 307, 619
sélections des caractères finaux 621
starts, opérateur de comparaison 666
suppression d'expressions de sous-chaînes 190
valeur numérique 747
chaînes de caractères
ASCII, codes 135, 472
caractère final dans les sélections 621
caractère initial dans les sélections 621
char...of, mot-clé 133
chars, fonction 134
comparaison 162
compte des caractères 470
compte des éléments 467
compte des lignes 467
compte des mots dans les expressions de sous-chaînes 469
conversion des expressions 681
dans les acteurs champ 692
do, commande 208
EMPTY, constante de caractères 222
expressions sous forme de 681
field, mot-clé 244
integer, fonction 336
item...of, mot-clé 345
last, fonction 356
length, fonction 360
line...of, mot-clé 363
offset, fonction de chaîne 474
put...after, commande 548
put...before, commande 549
put...into, commande 549
sélection 307, 619
sélection de mots 773
starts, opérateur de comparaison 666
suppression d'expressions de sous-chaînes 190
valeurs numériques 747
champ, propriétés des acteurs
autoTab 86
border 110
boxDropShadow 113
dropShadow 214
editable 219
lineCount, propriété d'acteur 364
margin, propriété d'acteur 386
pageHeight, propriété d'acteur 485
wordWrap, propriété d'acteur 773
chargement des acteurs
affichage d'informations 716
cancelIdleLoad, commande 126
finishIdleLoad, commande 253
idleLoadDone, fonction 317
idleLoadPeriod, propriété 318
idleLoadTag, propriété système 318
idleReadChunkSize, propriété 319
préchargement d'acteurs 528, 530
priorité de purge des acteurs 547
purge des acteurs 737, 738
vérification du chargement 369
chemin d'accès (@), opérateur 57
chemins d'accès
applicationPath, propriété 80
et navigateurs web 115
recherche par Director 617
chemins relatifs, opérateur de chemin d'accès (@) 57
clavier, et gestionnaire d'événements on timeOut 476
clics de la souris
affectation de scripts 418, 427
clickLoc, fonction 143
clickOn, fonction 144

comment déterminer si le bouton de la souris est enfoncé 418, 424
doubleClick, fonction 210
emulateMultiButtonMouse, propriété 222
état du bouton droit de la souris 591, 592
lastClick, fonction 357
lastEvent, fonction 357
mouseDown, fonction 418
mouseUp, fonction 424
on mouseDown, gestionnaire d'événement 417, 709
on mouseEnter, gestionnaire d'événement 422
on mouseLeave, gestionnaire d'événement 422
on mouseUp, gestionnaire d'événement 429
on mouseUpOutSide, gestionnaire d'événement 429
on mouseWithin, gestionnaire d'événement 429
on rightMouseUp, gestionnaire d'événements 592
stillDown, fonction 673
Cmd
touche (Macintosh) 158
collage du contenu du Presse-papiers dans des acteurs 493
Coller, activation de la commande 220
collisions
détection, et Lingo 35
résolution 584
commentaires (--), séparateur 47
comparaison de valeurs 127
comparaison, opérateurs
contains 162
différent de, opérateur (<>) 53
inférieur à, opérateur (<) 52
inférieur ou égal à (<=), opérateur 52
signe égal à (=), opérateur 53
sprite...within 658
starts 666
supérieur à (>), opérateur 53
supérieur ou égal à (>=), opérateur 54
comportements, association 342
compression 160
compte
caractères dans les chaînes 360
éléments dans les listes 167
paramètres transmis au gestionnaire 489
pistes des images-objets vidéo numérique 717
concaténation (& ou &&), opérateurs 47, 48
conditions
comparaison 127
end case, mot-clé 225
if...then, instructions 320
otherwise, mot-clé 483
repeat while, mot-clé 580
constante de chaîne (") 56
constantes
BACKSPACE 90
EMPTY 222
ENTER 229
FALSE 244
QUOTE 557
RETURN 587
SPACE 654
TAB 687
TRUE 729
VOID 766
constantes de caractères
BACKSPACE 90
EMPTY 222
ENTER 229
QUOTE 557
RETURN 587
TAB 687
constantes logiques
FALSE 244
TRUE 729
conversion
caractères en codes ASCII 135
codes ASCII en caractères 472
de palettes 486
durée en images 309
expressions en nombres à virgule flottante 258
images en durée 270
temps en battements 705
coordonnées
de fenêtres 567
de points 516
des images-objets 110, 160, 162, 359, 590, 714
des rectangles 566, 567, 653
copie
acteurs 167, 216, 391, 505
images 214
listes 215
Copier, activation de la commande 220
couleurs
couleur de scénario affectée aux images-objets 609
de premier plan pour les acteurs champ 263
définition de la couleur de la scène 661
définition des couleurs d'arrière-plan avec les acteurs 87
couleurs, codage
colorDepth, propriété 154
depth, propriété 197

des acteurs bitmap 197
des moniteurs 154
paramètre (Macintosh) 685
Couper, activation de la commande 220
courbe, ajout 453
création
acteurs 450
listes linéaires 368
menus personnalisés 398
objets enfants 450
rectangles 564
séparateurs d'éléments 346
Xtras 450
crochets ([]) 54
crochets de listes ([]) 54
Ctrl
touche (Macintosh) 163
touche (Windows) 158, 163
curseurs
changement des acteurs utilisés comme 180
cursor, commande 180
cursor, propriété 181
identification du caractère sous 416
mouseChar, fonction 416
mouseMember, fonction 424
pointeur, position de la souris 420, 428, 429
ressources de curseur 181
tests de survol 593
cylindres
Lingo pour 42
ouverts et fermés 714

D

date, objet 618
débits de téléchargement 97
débogage
avec la commande put 548
avec la commande showGlobals 639
avec la commande showLocals 639
défilement d'acteurs champ 613, 614
définition des couleurs d'arrière-plan avec les acteurs 87
démarrage
applications 476
caractère initial dans les sélections 621
sessions de mise à jour du scénario 91
déplacement
acteurs 431
fenêtres 431, 432
images-objets 430
désactivation
rapport de l'état de lecture en flux continu 691
son 648
dessin 212
dièse (#) 45, 551
différent de, opérateur (<>) 53
dimensionnement des rectangles et des points 384
dimensions des rectangles 331
diminution progressive du son 649
Director, sortie 556, 641
distance des lignes 366
distributions
acteurs sélectionnés dans 620
activeCastLib, propriété système 61
castLib, mot-clé 128
castLibNum 129
dernier acteur 470
distributions 465
enregistrement des modifications 603
mode de préchargement 531
nombre dans les animations 465
noms 440
noms de fichiers 246
division (/), opérateur 51
division, restes des 403
documents, ouverture d'applications 476
dossiers
de l'animation courante 436, 495
searchCurrentFolder, fonction 616
double-octet, caractères 334

E

échantillonnage
sampleRate, propriété d'acteur 602
trackNextSampleTime, propriété 718
trackPreviousSampleTime, propriété 719
écrans
centrage de la scène 131
identification des clics de souris dans 143
taille et position 199
éléments abrégés 58
éléments de menu
définition du texte 441
éléments de menu cochés 137
scripts exécutés par 610
sélection 223
éléments, séparation 346
EMBED, paramètres externes des balises 238, 239
emplacement
de la scène sur le bureau 661, 662, 664
des acteurs 134

des images-objets 161, 370, 371, 372, 657, 658
du pointeur de la souris 420, 428, 429
EMPTY, constante de caractères 222
en-tête HTTP, date de la dernière modification 447
encres
effets de traces 722
ink, propriété d'image-objet 332
endTellTarget, commande de l'Xtra Flash Asset 691
enregistrement 570
animations 604, 741
modifications des distributions 603
ENTER, constante de caractères 229
entiers
maxInteger, propriété 389
random 558
Entrée, touche 229
envoi de chaînes aux navigateurs web 237
erreurs
acteurs Shockwave Audio 278, 280
on alertHook, gestionnaire d'événement 227
pendant les opérations réseau 446
esclaves
images-objets 542, 544
pistes audio 544
pistes des cadences 545
pistes des palettes 543
valeurs d'opacité des images-objets esclaves 99
espacement, caractère 654
évaluation d'expressions 208, 548, 549, 625
événements
extérieur des fenêtres 404
identification des images-objets 177
exposants 526
expressions
comme entiers 337
conversion en chaînes 681
conversion en nombres à virgule flottante 258
évaluation 208, 548, 549, 625
FALSE, expressions 244
négation logique 463
nombres à virgule flottante 259
sous forme de chaînes 681
sous forme de symboles 686
expressions de sous-chaînes
char...of, mot-clé 133
compte des caractères 470
compte des éléments 467
compte des lignes 467
compte des mots 469
field, mot-clé 244
item...of, mot-clé 345
last, fonction 356
line...of, mot-clé 363
put...after, commande 548
put...before, commande 549
put...into, commande 549
sélection 307
sélection de mots 773
suppression 190
word...of, mot-clé 773
expressions logiques 76

F

FALSE, constante logique 244
fenêtres
affichage de chaînes dans les fenêtres de navigateurs 449
animations au premier plan 273
appellation 442
coordonnées 567
déplacement de fenêtres derrière d'autres fenêtres 431
déplacement devant toutes les autres fenêtres 432
événements à l'extérieur 404
fenêtre active 62
fermeture 147, 264
forget window, commande 264
on activateWindow, gestionnaire d'événements 61
on closeWindow, gestionnaire d'événement 227
on deactivateWindow, gestionnaire d'événement 227
on moveWindow, gestionnaire d'événements 477
on openWindow, gestionnaire d'événements 477
on zoomWindow, gestionnaire d'événements 476
ouverture 477
stage, propriété 660
visibilité 758
window, mot-clé 770
fenêtres d'animation
animations au premier plan 273
drawRect, propriété 213
exécution d'objets sous forme d'animations 771
liste 771
fermeture
fenêtres 147, 264
pistes audio 647
fichiers
écriture de chaînes 632
journalisation dans la fenêtre Messages 716
préchargement depuis Internet 211, 532

récupération 286
récupération de texte sur un réseau 284
fichiers de ressources
affichage des ressources 641
ouverture 476, 478
filtrage bilinéaire 444
Finder, sortie de Director vers (Macintosh) 556
Flash, animations
définition de variables 635
définition des propriétés 629
mixage des sons 651
flux continu, Lingo pour 37, 502
fonctions logarithmiques 236, 375
fonds 65, 67, 334
manipulation avec Lingo 36
format de temps court 705
format de temps long 705
formes
cadres 366
motifs pour 250, 495
types de 638
fractionnement
modificateur 40, 615
propriétés 615
FTP, définition des valeurs de serveurs proxy 541

G

génération de nombres aléatoires 559
gestionnaires
appel 121
dénombrement des paramètres transmis 489
identification de la fin 225
sortie 58, 234, 303
gestionnaires d'événements
ancestor, propriété 75
dénombrement des paramètres transmis 489
exit repeat, mot-clé 236
identification de la fin 225
on activateWindow 61
on alertHook 227
on closeWindow 227
on cuePassed 227
on deactivateWindow 227
on endSprite 227
on enterFrame 234, 501
on exitFrame 234
on idle 316
on keyDown 708
on keyUp 353
on mouseDown 417, 709
on mouseEnter 422
on mouseLeave 422
on mouseUp 429
on mouseUpOutSide 429
on mouseWithin 429
on moveWindow 477
on openWindow 477
on prepareFrame 534
on rightMouseUp 592
on stopMovie 680
on zoomWindow 476
on, mot-clé 476
result, fonction 586
return, mot-clé 587
sortie 58, 234, 303
tell, commande 689
timeOut 476
groupes (3D), Lingo pour 36
guillemets (") 557
guillemets droits (") 56

H

hauteur
acteurs champ 485
des acteurs 304
des lignes des acteurs champ 365
height, propriété d'acteur 304
lineHeight, fonction 365
horloge système 705
HTTP, définition des valeurs de serveurs proxy 541

I

identification de la fin des gestionnaires 225
if...then...else, instructions 320, 464
image-objets champ
modification 222
mouseChar, fonction 416
images
affichage du numéro de l'image courante 265
clearFrame, commande 141
conversion de durée 309
conversion en durée 270
copie 214
framesToHMS, fonction 270
HMStoFrames, fonction 309
impression 539
insertion 334
lecture 507
libellés affectés 266
liste des noms d'images 355
marker, fonction 386
mémoire nécessaire pour l'affichage 557

mise à jour 740
nombre de palettes 266
noms de repères 355
on enterFrame, gestionnaire d'événements 234
on exitFrame, gestionnaire d'événement 234
on prepareFrame, gestionnaire d'événements 534
paramètres de cadence 271
passage à 297
repères avant et après 386
suppression 192
suppression du contenu des images-objets des images 141
transitions 272, 546
images-clés
images-objets 730
trackNextKeyTime, propriété 718
trackPreviousKeyTime, propriété 719
images-objets
3D, Lingo pour 36
affectation d'acteurs 396
affichage des acteurs vidéo numérique 173
beepOn, propriété 91
clearFrame, commande 141
clickOn, fonction 144
compte des pistes des images-objets vidéo numérique 717
constrainV, fonction 162
coordonnées 110, 160, 162, 359, 590, 714
currentSpriteNum, propriété 177
déplacement 430
déplacement sur la scène 430
effets de traces 722
esclaves 542, 544
images-clés et 730
images-objets actives 144
lecture des pistes 717
média dans les pistes d'images-objets vidéo numérique 722
messages d'événements personnalisés 622, 623
modification d'images-objets champ 222
mouseChar, fonction 416
mouseMember, fonction 424
numéro de piste 658
numéro de script affecté 612
on cuePassed, gestionnaire d'événement 227
on endSprite, gestionnaire d'événement 227
points d'alignement 371, 372
points de repère 343, 414
position 161, 370, 371, 372, 657, 658
position temporelle d'arrêt des pistes 720
position temporelle de début des animations dans les pistes d'images-objets 720
position temporelle de lecture 177
redimensionnement 681
ressources de curseur utilisées lorsque le curseur les survole 181
scripts de comportement liés 611
scripts liés 633
spécification du numéro de position 395
sprite, mot-clé 657
suppression du contenu des images-objets des images 141
tests de survol 593
valeurs d'opacité des images-objets esclaves 99
visibilité 757
images-objets actives 144
images-objets audio, position temporelle de lecture 177
images-objets vidéo numérique
compte des pistes 717
lecture des pistes 717
média dans les pistes 722
texte dans les pistes 721
importation 330
impression
images 539
Scène 539
inactivité, durée 316
inférieur à, opérateur (<) 52
inférieur ou égal à (<=), opérateur 52
insertion d'images 334
installation de menus définis dans les acteurs champ 336
interligne des acteurs 365
Internet
lecture d'animations Shockwave 301
préchargement de fichiers 211, 532
téléchargement des acteurs 392
types de fichiers MIME 448
interprètes pour Lingo 594
intersection des images-objets 657
inverse, valeur 444

L

lancement d'applications 476
largeur, des acteurs 769
lecture
AIFF, fichiers 652
animations Shockwave à partir d'Internet 301
AVI, lecture des fichiers (Windows) 652
de pistes vidéo numérique 634
détermination des capacités avec Lingo 43

son 647, 652
WAVE, fichiers 652
lecture d'animations
après une pause 163
avec la commande play 507, 512
définition de la cadence 267
passage à un repère 299, 300
passage à une image 297
liaison
acteurs 246
animations 247, 612
scripts 367
libellés 266, 355
ligne, retour à la 773
lignes
dans les acteurs 364
dessin dans un objet image 212
distance 366
hauteur 365
symbole de continuation (\) 57
symbole de continuation (¬) 56
Lingo
éléments 3D, par catégorie 33
erreurs 227
interprètes 594
Xtras 470
listes
ajout à des listes linéaires 63, 68
ajout aux 80
ajout aux listes de propriétés 68
compte des éléments 167
copie 215
count, fonction 167
création de listes linéaires 368
de noms d'images 355
de références d'images-objets 611
fenêtres contenant des animations 771
findPos, commande 252
findPosNear, commande 253
getaProp, commande 274
getAt, commande 275
getLast, commande 283
getOne, commande 286
getPos, commande 289
getProp, commande 290
getPropAt, commande 290
identification des éléments 274, 275, 283, 286, 289, 290
ilk, fonction 322
noms de points de repère 175
opérateurs de listes ([]) 54
position des points de repère 176
position des propriétés dans les listes 252, 253
remplacement des valeurs de propriétés 626, 633
setaProp, commande 626
setAt, commande 627
setProp, commande 633
suppression d'éléments ou de valeurs 191, 195
suppression de tous les éléments 191
tri 644
types de 322, 368
valeur des paramètres 488
valeur maximale 388
valeur minimale 402
windowList, propriété 771
listes de propriétés
ajout 68
findPos, commande 252
findPosNear, commande 253
position des propriétés 252, 253
remplacement des valeurs de propriétés 626, 633
setaProp, commande 626
setProp, commande 633
suppression de valeurs 195
listes linéaires
ajout 63, 68
ajout aux 80
création 368
suppression de valeurs 195
logarithme, fonctions 236, 375
lumières 82
Lingo pour 37
lumière ambiante 199
lumière directionnelle 203

M

Macintosh, ordinateurs
arrêt 641
bip sonore 90
Lingo, interprètes 594
platform, propriété système 506
redémarrage 585
maille
couleurs 156
Lingo pour 42
masquage des contrôleurs d'acteurs 164
matériaux, Lingo pour 37, 637
matériel, obtention d'informations 282
Media Control Interface (MCI) 390
mémoire
allocation au programme 398

libre 272, 398, 434, 557
nécessaire pour l'affichage d'images 557
préchargement d'acteurs 528, 530
préchargement d'animations 531
purge des acteurs de la mémoire 737, 738
purge des animations 739
ramNeeded, fonction 557
taille des blocs disponibles 272
vérification du chargement des acteurs 369
mémoire libre 272
menus
de l'animation courante 470
définition de menus personnalisés 398
installation de menus définis dans les acteurs champ 336
name, propriété de menu 441
nombre de menus 470
messages
appel des gestionnaires par 121
commande alert 69
envoi à l'ancêtre de l'enfant 123
error 278, 280
événements 476
inactivité 316
messages d'événements personnalisés pour les images-objets 622, 623
stage, propriété système 660
tell, commande 689
transmission 492
messages d'événements
arrêt 476
personnalisés 622, 623
transmission 492
Messages, fenêtre
affichage des variables globales 639
affichage du résultat d'une expression 548
écriture de son contenu dans des fichiers 716
méthodes
identification de la fin des gestionnaires 225
return, mot-clé 587
sortie 234
MIAW. *Voir* animations MIAW
MIME, fichiers 301, 448
mipmapping 553
mise à jour
images 740
points d'alignement 504
Scénario 91, 227, 740
Scène 741
mod (modulo), opérateur 403
modèles
Lingo pour 38
sélection, Lingo pour 43
modificateurs 66
#inker 39
#toon 39
déformation de maille 39
fractionnement 40
fractionnement de surface 615
lecteur d'images-clés 40
lecteur de segments 40
Lingo pour 39
niveau de détail 41, 374
modification
acteurs 219
image-objets champ 222
moniteurs
centrage de la scène 131
codage de couleurs 154
taille et position 199
motifs, remplissage d'acteurs forme avec 250
mots dans les expressions de sous-chaînes 469
mouvement 415
lecture 508
minutage 178
multiplication (*), opérateur 49

N

navigateurs
affichage de chaînes 449
emplacement 115
envoi de chaînes 237
préchargement de fichiers depuis Internet 532
récupération de fichiers 286
suppression du cache 140
taille de mémoire cache 121
négation logique des expressions 463
niveau de détail
modificateur 41
propriétés du modificateur 374
nœuds
enfants, Lingo pour 41
gestion avec Lingo 41
parents, Lingo pour 41
suppression 192
nombres entiers aléatoires 558
noms
de repères, images associées 355
des acteurs 440
des animations 433, 436, 495

des chemins d'accès 57, 436, 495, 617
des distributions 440
noms de fichiers
acteurs liés 247
des distributions 246
recherche 286, 616, 617
noms des animations 436
noms des chemins d'accès 617
normales 274, 400, 461
liste 462
not, opérateur logique 463
nouvelle ligne (\), symbole 57
nouvelle ligne (¬), symbole 56
numéro d'emplacement des images-objets 395
numéros
affichage du numéro de l'image courante 265
conversion d'expressions en nombres à virgule flottante 258
distributions 465
exposants 526
génération de nombres aléatoires 559
les plus élevés supportés par le système 389
numéro d'acteur des scripts d'images 269
numéro de script affecté aux images-objets 612
virgule flottante, nombres 258, 259
numéros de piste, comportements des images-objets 658

O

OBJECT, balises 238, 239
objets
création et suppression avec Lingo 35
suppression 196, 575
objets enfants
ancestor, propriété 75
création 450
envoi de messages à l'ancêtre 123
me, mot-clé 390
suivi 62
objets image
copie 215
dessin dans 212
taille 566
ombres portées des acteurs champ 113, 214
opérateurs
concaténation (& ou &&) 47, 48
listes ([]) 54
mod 403
not 463
or 479
sprite...intersects 657
opérations réseau
annulation 445
erreurs 446
texte renvoyé 449
types de fichiers MIME 448
Option, touche (Macintosh) 478
Option-Retour (\), caractère 57
Option-Retour (¬), caractère 56
or, opérateur logique 479
ordinateurs
arrêt 641
platform, propriété système 506
redémarrage sur le Macintosh 585
ouverture
applications 476
fenêtres 477
MIME, fichiers 301
Shockwave, animations 301

P

palettes
affectation à des acteurs 486, 487
conversion 486
couleur de scénario affectée aux images-objets 609
dans l'image courante 266
paramètres
dans les listes 488
dénombrement des paramètres transmis aux gestionnaires 489
parenthèses (), opérateur 49
parents 489
particules, Lingo pour 42
pause
dans la lecture d'acteurs 499
dans les animations 163, 496, 500
pistes
acteur affecté à la première piste audio 270
acteur affecté à la seconde piste audio 270
channelCount, propriété 132
fermeture des pistes audio 647
lecture 634
pistes audio comme esclaves 544
pistes des cadences comme esclaves 545
pistes des palettes comme esclaves 543
sélection dans la fenêtre Scénario 609
vérification des pistes audio 645
pistes audio
acteur affecté à la première piste 270
acteur affecté à la seconde piste 270
acteurs 395
arrêt 674

boucle 378, 379, 380
esclaves 544
fermeture 647
lecture du son 645
liste de lecture 631
nombre d'échantillons 602
obtention de l'état 341, 672
rembobinage 588
pistes des palettes comme esclaves 543
placement des rectangles et des points 384
plage d'opacité, début et fin 104
plans, Lingo pour 42
plus (+), signe 50
pointeur, position
de la souris 420, 421, 423, 428, 429
points
coordonnées 516
identification 368
placement et dimensionnement 384
point, fonction 516
types 322
points d'alignement 574
des acteurs bitmap 573
des images-objets 371, 372
mise à jour 504
points de repère
images-objets 343, 414
liste de noms 175
liste des positions 176
polices 262
polices de caractères 261, 262
position
de la scène sur le bureau 661, 662, 664
des acteurs 134
des images-objets 161, 370, 371, 372, 657, 658
du pointeur de la souris 420, 428, 429
position temporelle de lecture
des images-objets audio 177
préchargement
acteurs 318
fichiers depuis Internet 211, 532
Shockwave Audio, acteurs 533
prélèvement, Lingo pour 43
premier-plan, définition des couleurs
avec les acteurs 263
Presse-papiers
collage du contenu dans des acteurs 493
copie d'acteurs dans 167
primitives, Lingo pour 41
priorité de purge des acteurs 547
projections
masquer 81
réduction 81
sortie 235
propriétés d'acteur son, bitRate 97
propriétés d'animation
allowZooming 73
center 130
controller 164
idleHandlerPeriod 316
idleReadChunkSize 319
paletteMapping 486
scoreSelection 609
scriptsEnabled, propriété d'acteur 612
updateLock 740
updateMovieEnabled 741
videoForWindowsPresent 754
propriétés d'images
frameLabel 266
framePalette 266
frameRate, propriété d'acteur 267
frameScript 269
frameSound1 270
frameSound2 270
frameTempo 271
frameTransition 272
lastFrame 358
timeScale 711
trackNextKeyTime 718
trackPreviousKeyTime 719
propriétés d'images-objets vidéo numérique
trackNextKeyTime 718
trackNextSampleTime 718
trackPreviousKeyTime 719
trackPreviousSampleTime 719
trackText 721
trackType, propriété d'image-objet 722
propriétés de boutons
buttonType 119
checkBoxAccess 136
hilite, propriété d'acteur 308
propriétés de champs
alignment 71
font, propriété d'acteur 261
fontSize, propriété d'acteur 261
fontStyle, propriété d'acteur 262
lineHeight, propriété d'acteur 365
selEnd 621
selStart 621

propriétés de fenêtres
activeWindow 62
drawRect 213
fileName 247
modal, propriété de fenêtre 404
name, propriété de fenêtre 442
sourceRect 653
title, propriété de fenêtre 711
titleVisible, propriété de fenêtre 712
visible, propriété de fenêtre 758
windowList 771
windowType, propriété de fenêtre 772
propriétés de la distribution
fileName 246
preLoadMode, propriété de distribution 531
selection, propriété de distribution 620
propriétés des éléments de menu
checkMark 137
enabled 223
name, propriété d'élément de menu 441
script, propriété d'élément de menu 610
propriétés globales, cpuHogTicks 169
propriétés système
activeCastLib 61
activeWindow 62
checkBoxType 137
deskTopRectList 199
digitalVideoTimeScale 201
emulateMultiButtonMouse 222
floatPrecision 259
frontWindow 273
idleLoadMode 318
idleLoadTag 318
keyPressed 352
Lingo pour 43
multiSound 439
platform 506
rightMouseDown 591
rightMouseUp 592
stage 660
proxy, définition des valeurs de serveurs 541
purge
acteurs de la mémoire 737
animations de la mémoire 739

Q

QuickTime 3, masques 387

R

rafraîchissement
de la scène 741
du contenu des pages web 120
rapport de l'état de lecture en flux continu 691
recherche
acteurs vides 252
animations 286
noms de fichiers 286, 616, 617
recouvrements
insertion 335
manipulation avec Lingo 36
rectangles
changement des dimensions 331
coordonnées 566, 567, 653
décalage 475
définition 564
identification 368
inflate, commande 331
inside, fonction 335
intersect, fonction 339
placement et dimensionnement 384
types 322
union, fonction de rectangle 737
récupération de fichiers 286
redimensionnement des images-objets 681
réflectivité 569
regroupement (),opérateur 49
rendu 526, 577
moteur 61
repères
affichage de la liste 387
avant et après les images 386
go loop, commande 299
loop, mot-clé 376
next, mot-clé 459
passage au repère précédent 300
passage au repère suivant 299
résolution 583, 584
ressources de modèle, Lingo pour 43
restes des divisions 403
retour à la ligne 773
Retour arrière (Macintosh) 90
rotation 598

S

Scénario
associé à l'animation courante 609
couleur de scénario affectée aux images-objets 609
enregistrement 91, 227

mise à jour 91, 740
pistes sélectionnées 609
Scène
centrage sur le moniteur 131
définition de la couleur 661
fixStageSize, propriété 255
impression 539
mise à jour 741
position des images-objets 370
position sur le bureau 661, 662, 664
rafraîchissement 741
taille après le chargement des animations 255
scripts
affectation aux clics de la souris 418, 427
affectation aux touches 350, 354
affectés aux acteurs 613
appel des gestionnaires dans les 121
associés aux images-objets 633
commentaires (--), séparateur 47
dans des animations liées 612
débogage 639
exécution d'un élément de menu 610
liés 367
numéro d'acteur des scripts d'images 269
numéro de script affecté aux images-objets 612
objets créés par des scripts parents 473
tests rollOver dans les scripts d'images 593
types de 613
scripts d'images
numéro d'acteur 269
tests de survol 593
scripts de comportement et images-objets 611
scripts parents
ancestor, propriété 75
me, mot-clé 390
objets créés par 473
sélection de texte 307
séparateurs
commentaires (--), séparateur 47
définition dans les éléments 346
séparation d'éléments 346
serveurs proxy 541
serveurs réseau, récupération de texte 284
session d'enregistrement de scénario 227
Shockwave Audio, acteurs
arrêt de la lecture 675
définition de l'URL 741
erreurs 278, 280
état 668
pause dans la lecture 499
percentPlayed, propriété d'acteur 501
percentStreamed, propriété d'acteur 501
play member, fonction 514
préchargement 533
preLoadBuffer member, commande 529
Shockwave, animations
lecture à partir d'Internet 301
nombre de paramètres externes 238
noms des paramètres externes 238
ouverture 301
valeurs des paramètres externes 239
signe égal à (=), opérateur 53
silhouettes 642
smoothness, propriété 644
son
activation ou désactivation 648
arrêt de la lecture 647, 652
augmentation progressive 649
contrôle du volume 651, 767, 768
diminution progressive 649
lecture 647, 652
niveaux 651, 767
sons, propriétés
channelCount 132
multiSound 439
sampleRate, propriété d'acteur 602
soundEnabled, propriété 648
soundLevel, propriété Macintosh 651
volume, propriété d'image-objet 768
volume, propriété système 767
sortie
Director 556, 641
gestionnaires 58, 234, 303
projections 235
soustraction (-), opérateur 46
spéculaire (propriété 3D)
propriété de lumière 654
propriété de matériau 655
spécularité 654
sphères, Lingo pour 43
sprite...intersects, opérateur 657
sprite...within, opérateur de comparaison 658
starts, opérateur de comparaison 666
supérieur à (>), opérateur 53
supérieur ou égal à (>=), opérateur 54
suppression
acteurs 231
animations de la mémoire 739
contenu des images-objets des images 141
de tous les éléments d'une liste 191

éléments dans les listes 191, 195
expressions de sous-chaînes 190
fenêtres 264
images dans une animation 192
valeurs dans les listes 195
symbole (#), opérateur 45, 551
symbole de continuation (\) 57
symbole de continuation (¬) 56
symboles
chaînes 686
expressions 686
symbole (#), opérateur 45, 551
système, éléments de bip sonore
alert, commande 69
beep, commande 90
systèmes de particules
distribution 207
gravity 297
Lingo pour 41

T

TAB, constante de caractère 687
Tab, touche 687
tabulation, ordre pour la propriété autoTab 86
taille
blocs de mémoire disponibles 272
chunkSize, propriété 139
de la scène 255
des acteurs 139, 642
des animations 435
des moniteurs 199
fixStageSize, propriété 255
idleReadChunkSize, propriété 319
lineSize, propriété d'acteur 366
size, propriété d'acteur 642
téléchargement des acteurs depuis Internet 392
temporisation, objet
détermination du nom 442
envoi d'événements aux objets enfants 689
renvoi 706
temporisations, Lingo 710
temps
conversion d'images en durée 270
conversion en battements 705
formats 705
time, fonction 705
unités de mesure 201, 711
tests de survol 357, 593
texte
ASCII, codes 135, 472
caractère final dans les sélections 621
caractère initial dans les sélections 621
chaînes dans les acteurs champ 692
chaînes de comparaison 162
char...of, mot-clé 133
charPosToLoc, fonction 134
chars, fonction 134
compte des caractères 470
compte des éléments 467
compte des lignes 467
compte des mots dans les expressions de sous-chaînes 469
conversion des expressions en chaînes 681
do, commande 208
EMPTY, constante de caractères 222
expressions sous forme de chaînes 681
field, mot-clé 244
item...of, mot-clé 345
last, fonction 356
length, fonction 360
line...of, mot-clé 363
offset, fonction de chaîne 474
opérateurs de concaténation (& ou &&) 47, 48
put...after, commande 548
put...before, commande 549
put...into, commande 549
récupération de fichiers présents sur des serveurs réseau 284
renvoyé par les opérations réseau 449
sélection 307, 619
sélection de mots 773
starts, opérateur de comparaison 666
suppression d'expressions de sous-chaînes 190
valeur numérique des chaînes 747
textures 693
coordonnées 400
Lingo pour 44
touches
affectation de scripts 354
Alt, touche (Windows) 478
Cmd, touche (Macintosh) 158
Ctrl, touche (Macintosh) 163
Ctrl, touche (Windows) 158, 163
dernière touche enfoncée 347, 348, 352
Entrée, touche 229
lastEvent, fonction 357
lastKey, fonction 358
on keyDown, gestionnaire d'événement 708
on keyUp, gestionnaire d'événement 353
Option, touche (Macintosh) 478
Retour arrière (Macintosh) 90

RETURN, constante de caractère 587
Suppression (Windows) 90
Tab, touche 687
Touche Maj 638
transformation d'identité 315
transformations
angles 86
inversion 340
Lingo pour 44
transitions
entre images 272, 546
puppetTransition, commande 546
transitionType, propriété d'acteur 725
types de 725
translations 725
tri des listes 644
TRUE, constante logique 729
tweenMode 730

U

UC, traitement des événements d'arrière-plan 169
unités de l'univers 484
unités de mesure du temps 201, 711
URL, acteurs Shockwave Audio 741
userData 745

V

valeurs, comparaison 127
variables
de propriété 549
de propriétés 540
variables globales 142, 295, 549, 639
variables locales 549, 640
voidP, propriété 766
vecteurs, Lingo pour 34, 749
vidéo
arrêt de la lecture 189, 303
pause dans les animations 163
préchargement d'animations 532
suppression d'images dans les animations 192
vidéo numérique
durée des 216
format 201
lecture des pistes 634
pause dans les animations 163
préchargement d'animations 532
suppression d'images dans les animations 192
vidéo numérique, propriétés d'acteur
center 130
controller 164
digitalVideoType 201
frameRate, propriété d'acteur 267
loop, propriété d'acteur 377
timeScale, propriété d'acteur 711
trackStartTime, propriété d'acteur 719
trackStopTime, propriété d'acteur 720
trackType, propriété d'acteur 721
video, propriété d'acteur 753, 757
volume, propriété d'image-objet 768
Vidéo pour Windows, logiciel 754
virgule flottante, nombres 258, 259
vitesse de lecture 437
VOID, constante 766

W

WAVE, fichiers 652
web, rafraîchissement du contenu des pages 120
Windows, ordinateurs
arrêt 641
bip sonore 90
platform, propriété système 506

X

XCMD et XFCN (Macintosh) 478
XCOD, ressources 478, 641
Xlibrary, fichiers
affichage d'Xtras et d'XObjects 641
fermeture 148
ouverture 478
XObjects
affichage dans les fichiers Xlibrary 641
ouverture 478
valeurs numériques des chaînes 747
Xtras
affichage dans les fichiers Xlibrary 641
création 450
disponibles pour l'animation 470
name, propriété d'Xtra 442
objets créés par 473
programmation 777
valeurs numériques des chaînes 747
xtra, fonction 777

Z

zones de texte des acteurs 113
zoom
effets 781
permission 73