Contents

For information on a command, click the name of the menu in which it appears:

Related topics:

Using Director Help
Keyboard shortcuts
Director Windows
Lingo

Using Director Help

Related topics: Acknowledgments
Copyright

This help system contains screens of information on using Macromedia Director. These screens are arranged in topics.

You can navigate through help topics by clicking underlined words (in green):

- Click words with a solid underline to jump to a related topic. (For example, to go to the contents screen click Contents).
- Click and hold words with a dotted underline to see pop-up graphics or text. For example: pop-up).

Buttons at the top of the help window help you find help topics:

- Click the **Contents** button of the help screen to see a list of help categories.
- Click the **Search** button of the help screen to see an index of the help system.
- Click the **Back** button to see the previously displayed help topic. (This button is not always active).
- Click the **History** button to see which help topics you have viewed. (This button is not always active).
- Click the **<<** and **>>** buttons to browse backward and forward through Director help. (These buttons are not always active).

You can also select commands from the Help window menus:

- Use the **Copy** command of the Edit menu to copy the text of a topic into the clipboard. This is especially useful for Lingo example topics.
- Use the **Annotate** command of the Edit menu to enter and save your own comments on a help topic.
- Use the **Bookmark** menu to assign a bookmark to the displayed help topic or to return to a topic that has a bookmark assigned.
- For more information, select the **How to Use Help** command of the Help menu of the Director Help window.

This is an example of pop-up help information.

Acknowlegments

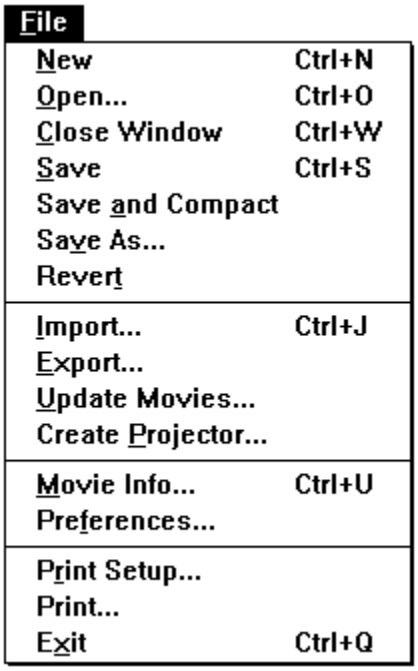
The Director 4.0 Help system was written by Jeff Swartz, based on sections from *Using Director* by Robin Foster and *Lingo Dictionary* by Joe Schmitz. Special thanks to Lalit Balchandani, Martin David, Steve Pasos, and Dan Sadowski.

Copyright

Macromedia, Inc.
600 Townsend St.
San Francisco, CA 94103

File menu

Click a menu command for more information:



The File menu contains commands for creating Director movies, opening and saving movies, importing and exporting files, creating projectors, and printing.

New command

The New command (Control-N) opens a new, untitled movie. Name the untitled movie by choosing Save from the File menu.

Open command

The Open command (Control-O) opens an existing Director movie. When you choose Open, the directory dialog box appears with a list of movie names.

By default, the dialog box lists only movies with the extension .DIR. You can change or delete the extension from the File Name box to list movies that don't have the .DIR extension.

Note: You can't open movies created with Director Player for Windows.

Close Window command

The Close Window command (Control-W) closes the current window, but doesn't close the stage.

Tip: To close all the windows except the stage, press Control-Alt-W.

Save command

The Save command (Control-S) saves the current movie to your disk. When you choose Save, the current version replaces the previous version with the same name. The first time you save a new Director movie or a movie created by a previous version of Director, the Save As dialog box is displayed so you can name the movie. It's a good practice to save often while you're working.

Any movie you save in Director 4.0 file format is compatible with Director 4.0 for Macintosh.

Save and Compact command

The Save and Compact (Control-Alt-S) command lets you save a movie under its original name so that it is optimized for playback. Since this operation reorders the cast and compacts the file, it takes longer than using the Save command, especially if you are saving a very large file. However, this command produces smaller and more efficient movies.

Use Save and Compact instead of the Save command to:

Reduce a movie to its minimum size: Save and Compact rewrites the file's entire contents so that any wasted space that might have accumulated in the original file is eliminated. This command differs from the Save command because the Save command only saves the cast members or other elements that have actually changed since you last saved the movie.

Gain optimum performance for playback on a CD-ROM drive or a slow hard disk: If you are working with a large movie, using the Save command will be faster. For large files, you might want to use Save and Compact when you are done authoring.

If you are working with a large movie, using the Save command will be faster. For a large movie, you might want to use Save and Compact when you are done authoring.

Save As command

Use the Save As (Control-Shift-S) command to name and save a new movie, to save an existing movie in a new file with a new name, or to save an existing movie on a different disk. Type the new name of the movie in the text box to name it, or click the Drive button to save to a different drive.

Revert command

The Revert command opens the last saved version of the current movie. This command is dimmed if you have not made any changes or if you are working on a new untitled movie that you have not yet saved.

Import command

Related topics: Linking to a file
Importing cast members with palettes

Use the Import command (Control-J) to import the following types of files into the cast (click an underlined file type for more information):

Windows bitmaps (.BMP)
Windows metafiles (.WMF)
Palettes (.PAL)
Sounds (.AIF and .WAV)
Director movies (.DIR)
Digital video (.AVI and .MOV)
MacPaint files (.PNT)
Macintosh PICT files (.PCT)
Encapsulated PostScript files (.EPS)
PC Paintbrush files (.PCX)
GIF files (.GIF)
Animator movies (.FLC and .FLI)
Photo CD files (.PCD)

When you choose Import, Director opens the cast window and displays a dialog box so you can choose the file you want to import.

Click here to see an illustration of the Import dialog box.

The import dialog box varies according to which type of file you wish to import.

The **Create** button is active for file types that can display a preview image. Click it to create a preview image of the selected file in the dialog box.

Choosing a file type from the List File of **Type** drop-down list displays only the files available in the chosen file format. When you select a file and click Import, the file is imported and appears in the first available cast position in the cast window. Clicking Import All causes all files of the selected type in the current folder to be imported into the cast.

Linking to a file

When you import a .WAV or .AIF sound file; PICT file; .DIB or .BMP bitmap; or a Director movie, you have the option of creating a link to the file in the cast rather than copying the contents of the file into your movie. This allows you to add a sound, bitmap, or Director movie to the current movie without increasing the size of the movie. It also allows you share the same linked file among several movies. If you change the linked file, the changes are not automatically reflected in the movie. To update the cast member that corresponds to a linked file, close the movie and then reopen it. Director looks for any linked cast members when you open the movie. If it cannot find a linked file, a dialog box appears so you can locate the file.

Director always imports digital video movies by linking to the file.

Note: A linked cast member's contents don't become part of a Director movie. If you make a copy of the Director movie, be sure to include copies of all the cast members that are linked to the movie.

Director cannot keep track of path names with more than 255 characters. Occasionally, files that are nested too many directories away from the movie file will fail to link correctly. For this reason, it's a good idea to keep linked files in a directory that's close to the original movie file.

Importing cast members with palettes

If you import a cast member that uses a palette that is different from the currently active palette, you will see the Palette Mismatch dialog box, in which you can select one of the following options:

- **Remap Colors**--Remaps the cast member's colors to the closest colors in the current palette.

- **Remap Colors and Dither**--Remaps the cast member's colors to the closest colors in the current palette and dithers the result.

- **Install Palette in Cast**--Imports the cast members palette. This ensures that the cast member looks exactly as it did when it was originally created.

Only one palette can be active at a time:

- While a movie is running, the active palette is determined by the position of the playback head in the score. The palette that's active in the frame where the playback head is located at any given moment is in control of the monitor during that frame.

- When a movie is stopped, the palette associated with the window that's currently active is in control. The most obvious example is the palette window: when it's active, the palette that's selected is in control. If the cast window is active, the palette associated with the cast member currently selected is in control. (Since a palette can itself be a cast member, you can switch control to a palette just by selecting its cast member in the cast window.) If the paint window is active, the palette associated with the cast member displayed there is in control. And so on.

Importing PICT files

If you import a PICT file, Director converts it to a bitmapped cast member and adds it to the cast. (If you don't want Director to convert the PICT file to a bitmapped graphic, check the As PICT checkbox when you import it.)

Linked File--If this box is checked, Director creates a link to the location of the PICT file on disk. Linked PICT files are always imported as bitmapped graphics (when you select the Linked File checkbox, the As PICT checkbox is dimmed). Linking to a PICT file is useful if you want to add large (16- or 24-bit) images to your movie without increasing the movie's size.

As PICT--Imports the PICT file in its original Macintosh PICT format rather than as a bitmapped graphic. If you import a PICT file in PICT format, you can't edit it in the paint window. The As PICT checkbox is dimmed when the Linked File checkbox is selected.

Importing MacPaint files

If you import a MacPaint file, Director adds it to the cast window.

Importing Sounds

Imported sounds are added to the cast. You can import .AIF and .WAV files.

Linked File--Check the checkbox to create a link to a .AIF or .WAV sound file on disk. When you link a sound file to a Director movie, you can use a sound editor to edit the sound. To update the cast member corresponding to a sound file you've edited, close the movie and reopen it.

Importing Director movies

Importing a Director movie converts the movie into a film loop, which Director adds to the cast window. Director imports all the movie's cast members, adds them to the cast window, and updates any references to them to reflect their new positions in the cast window. Director doesn't import the movie's tempos, transitions, or markers. Although it imports any scripts that are part of the movie, the movie's score scripts are no longer active. (That's because a score script can't be made part of a film loop.) If you want the movie's scripts to remain active, link the movie rather than importing it.

Linked File--Lets you create a link to a Director movie. Linking doesn't convert the movie into a film loop, nor does it add the movie's cast members to the cast window. Instead, it adds a single cast member--a linked movie cast member--to the cast window. One advantage in linking to a movie rather than importing it is that a linked movie doesn't increase the file size of the current movie. Another important advantage is that any scripts that are part of the linked movie remain functional. (To activate the scripts, select Enable Scripts in the linked movie's Cast Member Info dialog box.) Unlike scripts, however, tempo settings, palette settings, and transitions that are in a movie aren't functional when the movie is linked.

Note: You cannot link to a movie created with Director Player for Windows.

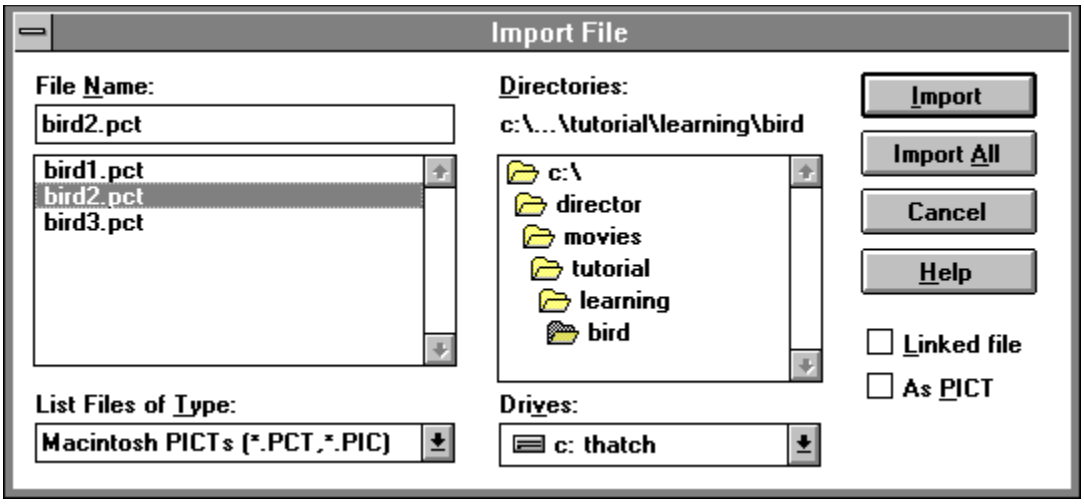
Importing digital video movies

Importing a digital video movie automatically links the file to the current movie and adds it to the cast window.

When you import a digital video movie, Director creates a link to the file that contains the movie rather than importing the movie in its entirety. If you use an application such as Premiere to make changes to the movie after you've imported it, you need to update the cast member that corresponds to the movie by closing and then reopening the Director movie that the digital video movie is linked to.

Note: The imported digital video movie's contents do not become part of the Director movie. If you make a copy of the Director movie, be sure to include all digital video movies that are part of the movie.

Related topic: Importing cast members with palettes


Import File
File Name:
bird2.pct
bird1.pct
bird2.pct
bird3.pct
Directories:
c:\...\tutorial\learning\bird
c:\
director
movies
tutorial
learning
bird
Import
Import All
Cancel
Help
Linked file
As PICT
List Files of Type:
Macintosh PICTs (*.PCT,*.PIC)
Drives:
c: thatch

Export command

Use the Export command (Control-Shift-E) to export frames of Director movies from Director to a file so you can record frames to videotape, save images as stills, or create digital video movies. You can choose to export frames of movies as a DIB file sequence in .BMP format or as a digital video movie in Video for Windows (.AVI) format.

Click here to see an illustration.

Click an element of the export dialog box for more information:

- Range of Frames
- Within Range of Frames
- Destination
- Video for Windows Options

Always save the movie you're working on before you export from it.

Related topic: Converting Director animations into digital video movies

Range of Frames (Export command)

These options determine which frames to export:

Current Frame--Exports just the current frame on the stage. This is the default.

Selected Frames--Exports the selected frames in the score window.

All--Exports all frames.

From/To--Exports only the range of frames that begin and end with the frame numbers you enter in the From: and To: boxes.

Within Range of Frames (Export command)

Further modifies your selection specified in the range of frames.

Every Frame--Exports all frames in the selected range.

Every Nth Frame--Exports only the frames at the interval you specify in the "N=" box.

Frames With Markers--Exports frames with markers set in the score window.

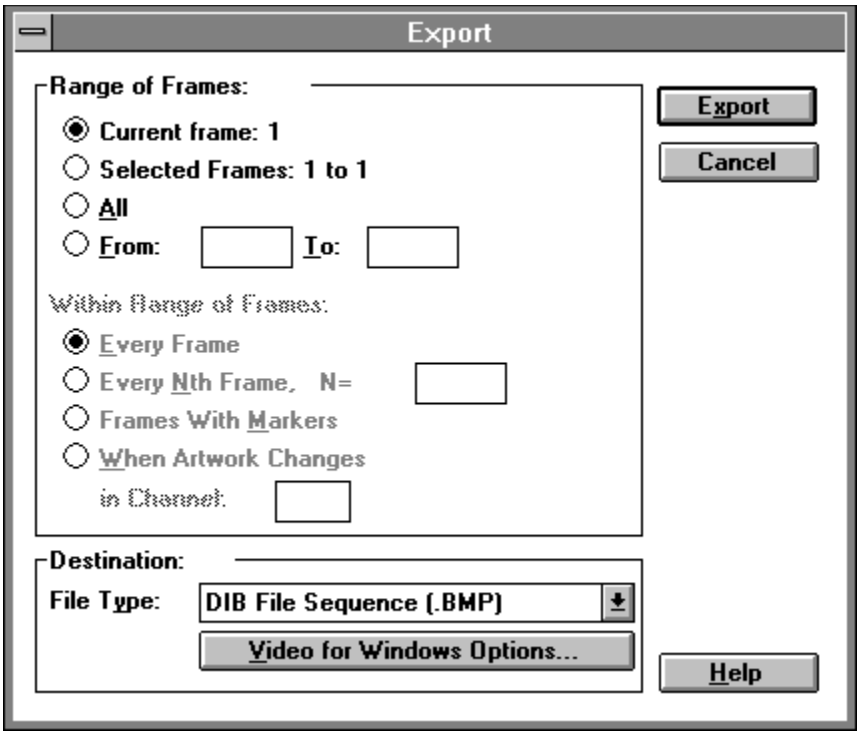
When Artwork Changes in Channel--Exports frames only when a cast member changes in the channel you specify in the box.

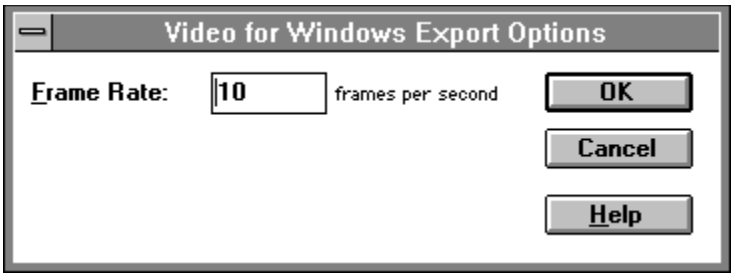
Destination (Export command)

File Type--Choose the exported file format from the File Type drop-down list. File formats you can export are DIB file sequence (.BMP) and Video for Windows (.AVI).

Video for Windows Options--This button is dimmed unless you choose Video for Windows (.AVI) from the File Type drop-down list. For a description of the options, see Exporting digital video.

Export--When you click Export, a dialog box appears, allowing you to name the file. If you're saving a DIB file sequence, Director automatically creates one file for each frame, attaching the corresponding frame number to each file. For example, if the name of the exported file is "MYFILE", Frame 1 will be exported to a file named "MYFI0001.BMP". Notice that if you enter a file name longer than four characters, Director drops any characters after the fourth and replaces them with four digits that correspond to the number of the frame.


Export
Range of Frames:
Current frame: 1
Selected Frames: 1 to 1
All
From:
To:
Within Range of Frames:
Every Frame
Every Nth Frame, N=
Frames With Markers
When Artwork Changes
in Channel:
Destination:
File Type:
DIB File Sequence (.BMP)
Video for Windows Options...
Export
Cancel
Help


Video for Windows Export Options
Frame Rate:
10
frames per second
OK
Cancel
Help

Video for Windows Export Options

The Video for Windows Options button in the Export dialog box opens the Video for Windows Export Options dialog box. This dialog box has controls for determining the frame rate.

Converting Director animations into digital video movies

You can export a Director animation as a digital video movie, and then import it into Director so that it becomes a single cast member. You might want to do this for the following reasons:

An imported digital video movie is linked to its source file on disk. If you edit the digital video movie in an application such as Premiere, the changes will be reflected in Director without you've having to delete the old version of the movie and import it again. (To update the digital video cast member in a Director movie, you do, however, need to close the Director movie and then reopen it.)

Director provides precise control over the digital video movie's playback, using the Digital Video Cast Member Info dialog box.

Other users can use and distribute the digital video movie for use in any application that supports digital video.

You might not want to convert a Director animation into a digital video movie if the animation includes interactivity. (Digital video movies do not allow interactivity.)

Update Movies command

Saves, compacts, and protects one or more movies. Saving and compacting a movie has the same effect as the Save and Compact command. Protecting a movie removes Lingo source code and cast thumbnail images from the movie. Making those changes prevents anyone from editing the movie--protected movies can no longer be opened in Director--but it doesn't change the way the movie plays.

Although protected movies can't be opened in Director, they can be played from a projector. To play a protected movie from a projector, you need to use a go to movie or play movie Lingo command. (For more information about projectors, see the next heading, "Create Projector"; for more information about Lingo, see Using Lingo.)

A protected movie has a .DXR extension rather than a .DIR extension.

Click here to see an illustration.

To save, compact, and protect a movie:

1. Choose Update Movie from the File menu.

   The Choose Movies to Update dialog box appears.

2. Select the movie you want to update.

   - To update all the movies in the directory, click the Select All button.
   - To select a range of movies, click the one at the beginning of the range and then Shift-click the one at the end of the range.
   - Movies that aren't in a continuous range, click the first one and then Control-click the others.

3. Click OK.

   The Save Updated Movies dialog box appears.

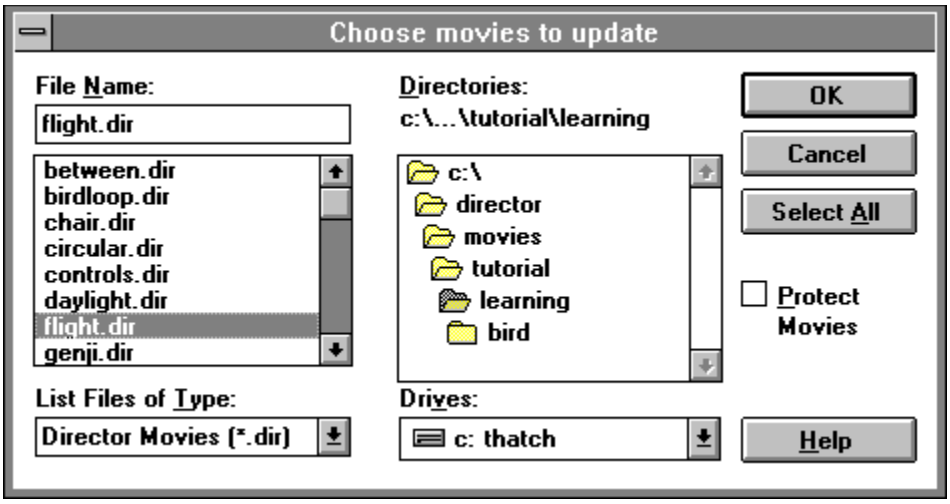
4. Select the directory where you want to save the movie.

   You can rename the movie if you want to, but it's not necessary: Director automatically changes a movie's extension from .DIR to .DXR when it updates the movie. If you want to both update and rename a group of movies, update them one by one (you can't rename movies when you update them as a group).

   Director prevents you from overwriting your existing movie or directory with the updated movie or directory.

5. Click OK.

   Director compacts, protects, and then saves the movie.


Choose movies to update
File Name:
flight.dir
between.dir
birdloop.dir
chair.dir
circular.dir
controls.dir
daylight.dir
flight.dir
genji.dir
List Files of Type:
Director Movies (*.dir)
Directories:
c:\...\tutorial\learning
c:\
director
movies
tutorial
learning
bird
Drives:
c: thatch
OK
Cancel
Select All
Protect Movies
Help

Create Projector command

Lets you create a play-only version of a Director movie, called a projector, that can be played on any computer running Windows 3.1. You can't play a projector you've created in Director for Macintosh on a Windows computer, and you can't play a projector you've created in Director for Windows on a Macintosh. (You can, however, open and play movies you've created with Director for Macintosh in Director for Windows--and vice versa.)

Click here to see an illustration.

To play a projector, either double-click its icon or run the .EXE file that contains it.

If a movie is open when you choose this command, Director closes the current movie (after asking you if you want to save your changes).

Click for more information:

- Add button
- Move Up/Move Down
- Create
- Options

Use this dialog box to add one or more movies from the Source Folder to the projector's play list.

Note: You can't create a projector from a movie created with Director Player for Windows.

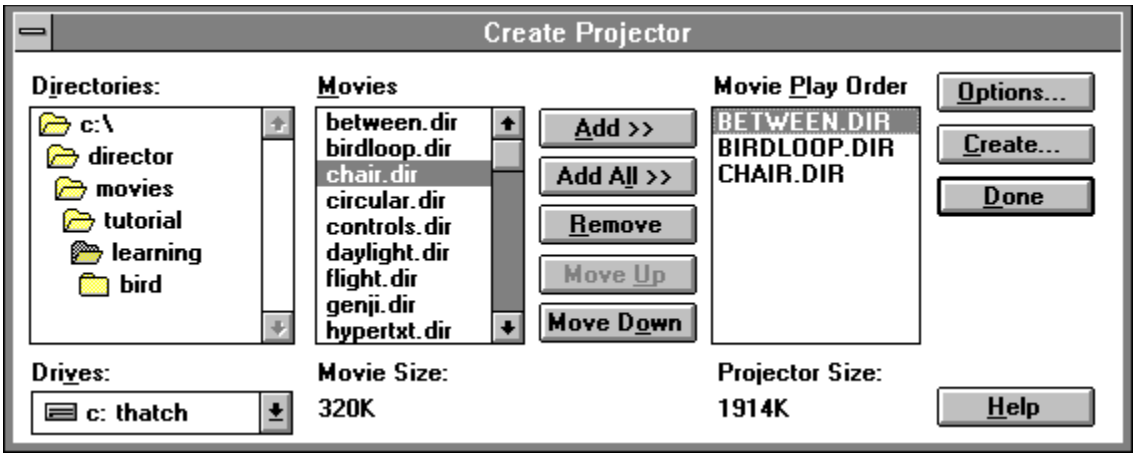
Clicking Add adds the selected source movie to the projector's play list. Clicking Add All adds all movies in the current directory to the play list. Clicking Remove removes a selected movie from the projector's play list.

Click Move Up to move a selected movie higher up in the play list. Click Move Down to move a selected movie further down in the play list.

You can include more than one interactive movie in the projector file list. You can also use Lingo to "go to" or "play" files while running a projector under Lingo control.

Click Create to create the projector file. Click Done to dismiss the dialog box.

Note: If your movies include links to external files, you must include the files when you distribute the projector.


Create Projector
Directories:
c:\
director
movies
tutorial
learning
bird
Movies
between.dir
birdloop.dir
chair.dir
circular.dir
controls.dir
daylight.dir
flight.dir
genji.dir
hypertxt.dir
Add >>
Add All >>
Remove
Move Up
Move Down
Movie Play Order
BETWEEN.DIR
BIRDLOOP.DIR
CHAIR.DIR
Options...
Create...
Done
Drives:
c: thatch
Movie Size:
320K
Projector Size:
1914K
Help

Add button (Create Projector command)

Clicking Add adds the selected source movie to the projector's play list. Clicking Add All adds all movies in the current folder to the play list. Clicking Remove removes a selected movie from the projector's play list.

Move Up and Move Down buttons (Create Projector command)

Click Move Up to move a selected movie higher up in the play list. Click Move Down to move a selected movie further down in the play list.

You can include more than one interactive movie in the projector file list. You can also use Lingo to "go to" or "play" files while running a projector under Lingo control.

Create button (Create Projector command)

Click Create to create the projector file. Click Done to dismiss the dialog box

Options button (Create Projector command)

The Options button displays a dialog box that specifies additional preferences for creating the projector.

Click here to see an illustration.

Note: These settings override the movie preferences you set using the Preferences command in the File menu.

These settings apply to all movies in the projector.

Click for more information:

- Play Every Movie
- Animate in Background
- Display Projector
- Resize Stage

Play Every Movie (Projector Options)

If checked, the projector plays all movies in the play list. Otherwise, the projector only plays the first movie in the play list unless other movies are called by Lingo from the first movie. In a projector with Play Every Movie checked, pressing Control-period will go to the next movie and Control-Q will quit.

Animate in Background (Projector Options)

If checked, if a user clicks outside the movie, on the desktop, for example, the movie continues playing. This is useful if you are using Apple Events. If not checked, the movie stops playing if the user clicks outside the movie.

Resize Stage (Projector Options)

If checked, the stage size changes when you open a movie that uses a stage size that is different from the current stage size. If not checked, the stage size remains unchanged.

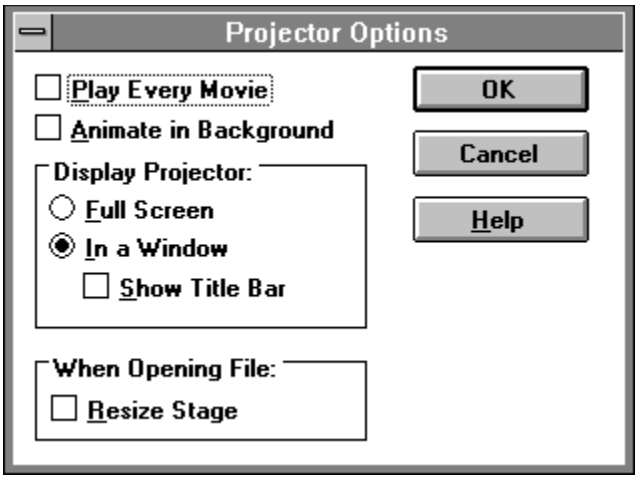
Display Projector (Projector Options)

The Display Projector options you choose set up the window in which the projector will appear.

Full Screen--If selected, displays the movie full screen, placing the menu bar (if there is one) at the top of the screen and hiding all of the Windows desktop. If there's a menu, it overlays the top of the stage. If a movie is the size of the screen or larger, Director automatically displays it in Full Screen mode and crops it to fit the screen if necessary.

In a Window--If selected, displays the movie in a normal window, without taking over the screen. The window is not resizeable, but it is moveable. If there's a menu, it overlays the top of the stage.

Show Title Bar--Available only if In a Window is selected. If checked, the window where the movie appears has a title bar; the name you give the projector file appears in the title bar.


Projector Options
Play Every Movie
Animate in Background
Display Projector:
Full Screen
In a Window
Show Title Bar
When Opening File:
Resize Stage
OK
Cancel
Help

Movie Info command

The Movie Info command (Control-U) lets you choose basic settings for the current movie.

Click here to see an illustration.

Click for more information:

- User Info
- Remap Palettes When Needed
- Allow Outdated Lingo
- Load Cast
- Default Palette
- Font Mapping Table

User Info (Movie Info)

Specifies the movie's creator and modifier.

Remap Palettes When Needed (Movie Info)

This option only affects the appearance of cast members on the stage. If this box is checked, whenever the cast member is displayed on the stage and the currently active palette (the palette in effect in a given frame of the movie) is different from the palette associated with the cast member, Director automatically creates a common palette and remaps all images on the stage that have a different palette to the common palette. The cast members themselves are not modified. The common palette determines how the cast member is remapped. For example, if a cast member uses a grayscale palette, it will be drawn on the stage using whatever grays are available in the common palette.

Allow Outdated Lingo (Movie Info)

Lets you include Lingo commands used by previous versions of Director.

Load Cast (Movie Info)

Provides a drop-down list that lists the options available for pre-loading cast members into memory.

When Needed--Choose this option to load cast members only when they are required and are not currently available in memory. Best for interactive movies. Don't use this option for movies that require high-speed animation.

After Frame One--Choose this option to display the first frame of animation and start playing any sounds in the first frame of the sound channels before loading the rest of the movie into memory. Best for large movies where you want to display an opening screen or play some initial sound while waiting for the rest of the movie to get loaded into memory. This is the default.

Before Frame One--Choose this option to load the entire movie into memory before playing the first frame of the movie. Best for small movies.

A new movie inherits the Load Cast setting from the movie that was previously open, as specified using the Movie Info command.

Default Palette (Movie Info)

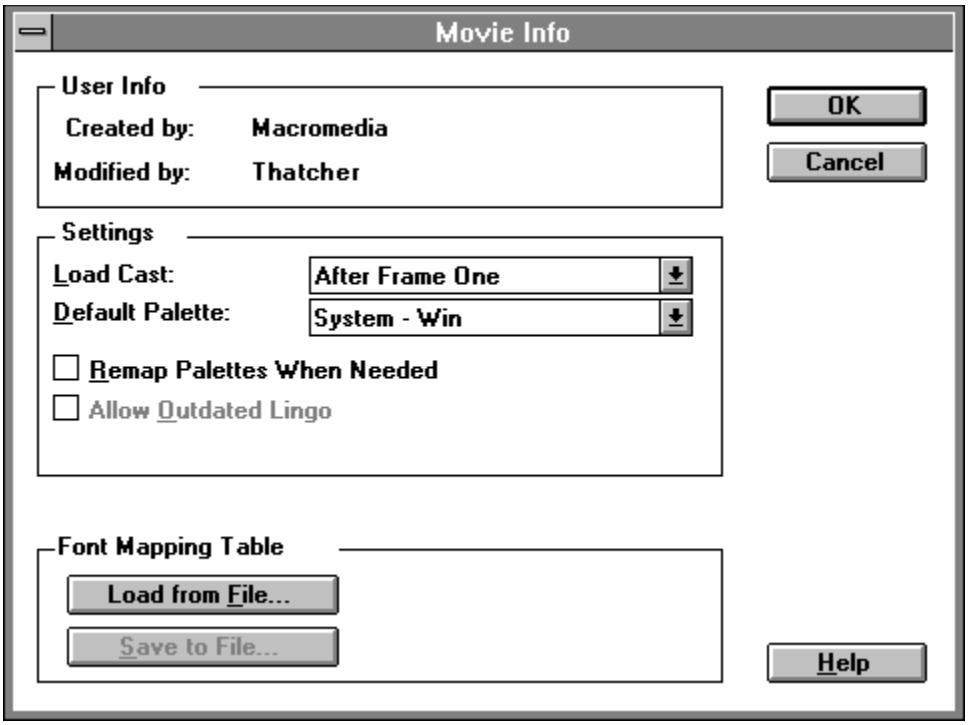
Lets you specify the movie's default palette. Director uses the default palette until it encounters a different palette setting in the palette channel.

Font Mapping Table (Movie Info)

Director uses a font map table to determine the appropriate substitute Macintosh fonts for text cast members created by Director for Windows. The font map table specifies how Director maps fonts between the Macintosh and Windows platforms. The font map table consists of a text file that contains the font mapping information.

Load from File--Click Load from File to have Director load the font mapping assignments specified in the chosen font map file.

Save to File--Click Save to File to save the current font map settings in a text file. You can then edit this file to include the names of appropriate substitute Windows fonts. (This button is dimmed unless the movie has a font map associated with it.)


Movie Info
User Info
Created by: Macromedia
Modified by: Thatcher
OK
Cancel
Settings
Load Cast:
After Frame One
Default Palette:
System - Win
Remap Palettes When Needed
Allow Outdated Lingo
Font Mapping Table
Load from File...
Save to File...
Help

Preferences command

The Preferences command (Control-Alt-U) allows you to modify some of Director's default settings.

Click here to see an illustration.

Click for more information:

- Stage Size
- Stage Location
- When Opening a Movie
- Save Settings
- Other Options

Director saves the cast window options and window positions in the Director 4.0 Preferences file (DIRECTOR.PRF). Other preferences are saved with, and are specific to, your current movie.

Stage size (Preferences)

These settings permit you to change the size of the stage window. Changing the stage size is useful if you want to display movies on a smaller or larger stage, or if you want to change the stage size to match the size of a digital video movie.

You can change the size of the stage by choosing a setting from the drop-down list, or by entering the width and height of the stage.

If you choose a setting from the drop-down list, the values in the Width and Height fields automatically update.

Menu selection	Example	Stage size (pixels)
VGA		640 x 480
SVGA		800 x 600
1024		1024 x 768
1280		1280 x 1024
QuickTime	QuickTime movie	160 x 120
Current screen	13-inch monitor	640 x 480
Custom		User defined

Note that if you enter a value for Width or Height that is not divisible by 16, Director uses the nearest smaller number that is divisible by 16.

Stage Location (Preferences)

Stage Location permits you to change the location of the stage.

Centered--Places the stage window in the center of your monitor. This option is useful if you play a movie that was created for a standard screen on a larger screen. You can also use this option if you are creating a movie on a larger screen that will be seen on smaller screens.

Left, Top--Alternatively, the values you type in the Left and Top boxes represent the number of pixels the stage is moved from the top left corner of the screen. These values only apply if the stage is smaller than the current monitor's screen size.

When Opening a Movie (Preferences)

Use Movie's Size/Location--If you open a movie that has a stage that is a different size than the current stage, the stage size changes to the movie's stage size and location.

Always Center--If you choose this option, the stage is centered on the screen when the movie opens.

Don't Change Size/Location--If you choose this option, your stage size and location remains the same as the current movie.

Save Settings (Preferences)

When Quitting--If chosen, Director saves the settings in the Preferences dialog box every time you quit.

Now--If checked, Director saves the settings when you click OK to leave the Preferences dialog box, but not when you quit.

When Quitting--If chosen, Director saves the settings in the Preferences dialog box every time you quit.

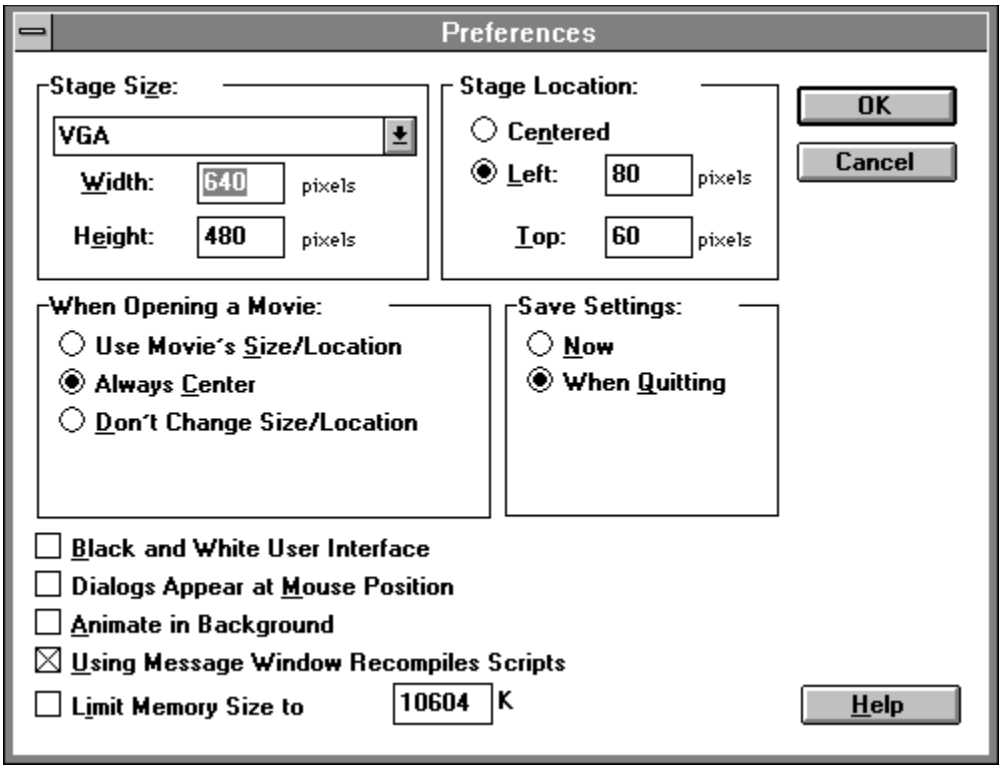
Other options (Preferences)

Black and White User Interface--If checked, Director switches to a black and white user interface. Using a black and white user interface improves performance if you switch color palettes, since Director doesn't have to update its color user interface to match the colors in the new palette each time you switch palettes. In addition, working with a black and white user interface may be less distracting as you work with the color palettes in your movie. For example, if you are working on an animation that uses multiple palettes and/or color cycling, using a black and white user interface may be less distracting.

If not checked, Director uses a color user interface. If you switch to a non-system color palette in your movie, Director switches the colors it uses in its user interface to match the palette in use.

Dialogs Appear At Mouse Position--If checked, dialog boxes are displayed at the mouse position. If this option is not checked, dialog boxes are centered on the monitor that contains the menu bar.

Animate In Background--If checked, your animation runs in the background while you are working with other applications in the Finder. When you are running animation in the background, the stage remains on the screen and the active application window appears in front of the stage.


Preferences
Stage Size:
VGA
Width:
640
pixels
Height:
480
pixels
Stage Location:
Centered
Left:
80
pixels
Top:
60
pixels
OK
Cancel
When Opening a Movie:
Use Movie's Size/Location
Always Center
Don't Change Size/Location
Save Settings:
Now
When Quitting
Black and White User Interface
Dialogs Appear at Mouse Position
Animate in Background
Using Message Window Recompiles Scripts
Limit Memory Size to
10604
K
Help

Print Setup command

The Print Setup command offers options for determining how a page is to be printed. The dialog box that you see depends on the type of printer you use. For more information about Print Setup options see your printer manual.

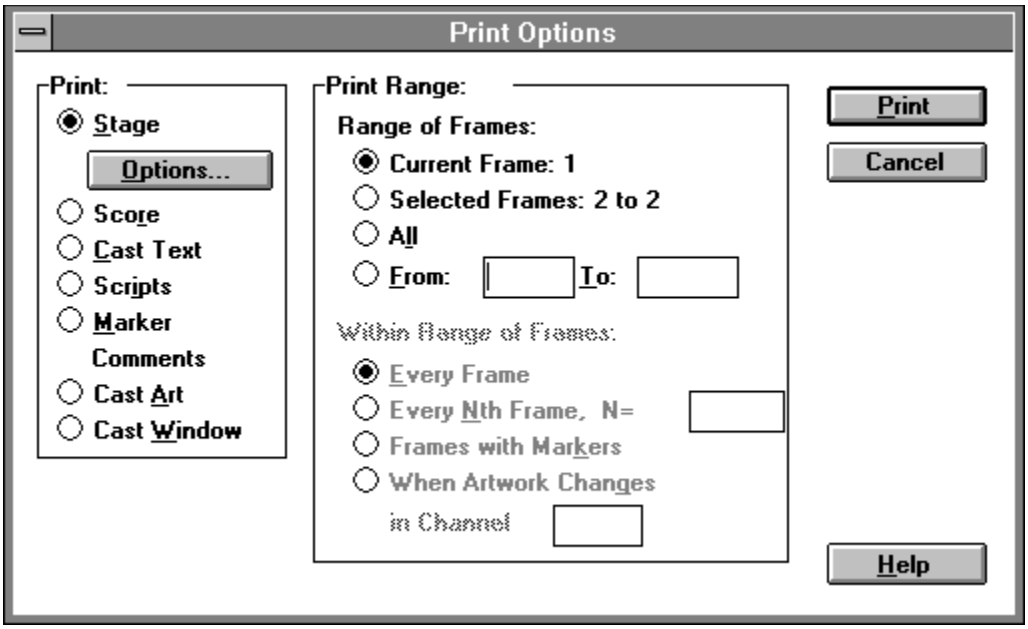
Print command

The Print command (Control-Alt-P) lets you print your movie in a variety of ways.

Click here to see an illustration.

Click for more information

- Print section of Print dialog box
- Print Range
- Stage Options


Print Options
Print:
Stage
Options...
Score
Cast Text
Scripts
Marker Comments
Cast Art
Cast Window
Print Range:
Range of Frames:
Current Frame: 1
Selected Frames: 2 to 2
All
From:
To:
Within Range of Frames:
Every Frame
Every Nth Frame, N=
Frames with Markers
When Artwork Changes in Channel
Print
Cancel
Help

Print section (Print command)

Lets you choose what part of your movie you want to print. You can print an image of the stage, the score, the cast member number and contents of text cast members in the cast window, all scripts or a range of scripts (movie, cast, score, and sprite scripts), the comments in the markers window, the cast window artwork, or all of the cast window.

When you use the Print command to print multiple frames per page, cast members in each frame are merged into a single image. If you are printing a single frame on a page, each cast member is imaged separately. Printing with this option causes draw objects (text, rectangles, lines, and circles) to print better.

If you are printing the stage, the Options button lets you specify additional printing options.

Print Range (Print command)

Controls which frames of your movie are printed.

Current Frame--Print the frame that is currently on the stage.

Selected Frames--Prints the frames that are selected in the score window.

All--Print all the frames in your movie.

From: To:--Enter a range of frames to print.

Within Range of Frames--Defines your selection within the range specified in Range of Frames.

Every Frame--This is the default and does not modify the choice in Range of Frames.

Every Nth Frame--Print frames at the interval you specify in the box. For example, if you type 10, Director prints every 10th frame.

Frames with Markers--Prints only the frames that have markers in the score window.

When Artwork Changes In Channel--Prints the frames in which cast members move or in which new cast members are introduced in the score, in the channel you specify in the box.

Stage Options (Print command)

Displays a dialog box that lets you adjust the layout of the items you choose to print. The image at the left of the dialog box previews the layout options.

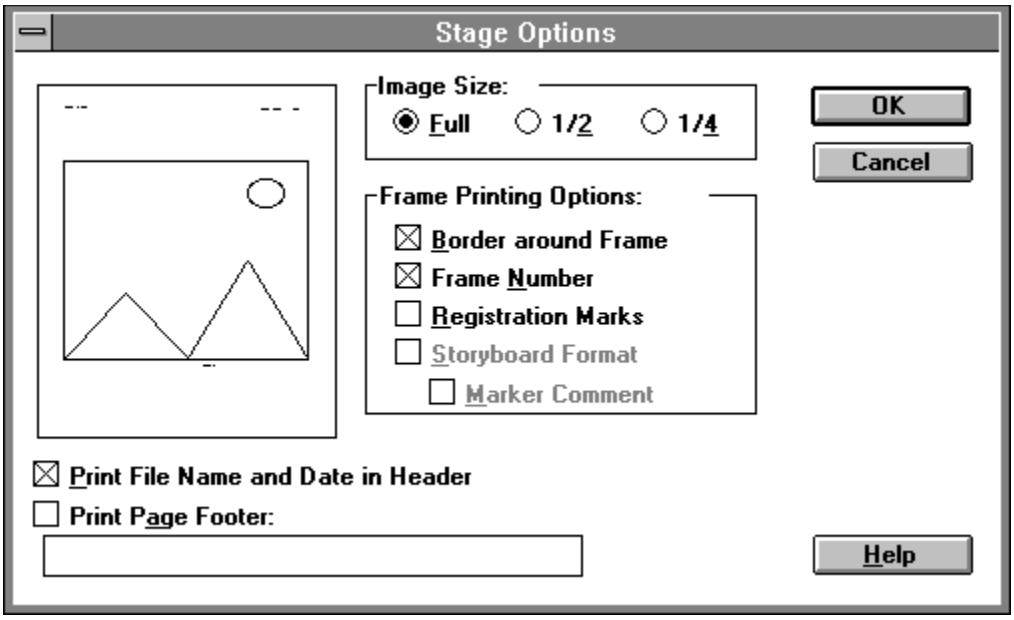
Click here to see an illustration.

Image Size--You can print your document at full size, or scale the images to 1/2 or 1/4 size, which is convenient for printing the images in storyboard format.

Frame Printing Options--Allows you to print a border around each frame, include the frame number, print registration marks, or print in storyboard format and include the comments associated with each frame. Storyboard format is only available if you are printing 1/2- or 1/4-size images.

Print File Name and Date in Header--If checked, prints a header on each page. The header consists of the name of the Director movie and the current date.

Print Page Footer--If checked, lets you print a footer on each page. Type the footer in the field.


Stage Options
Image Size:
Full
1/2
1/4
OK
Cancel
Frame Printing Options:
Border around Frame
Frame Number
Registration Marks
Storyboard Format
Marker Comment
Print File Name and Date in Header
Print Page Footer:
Help

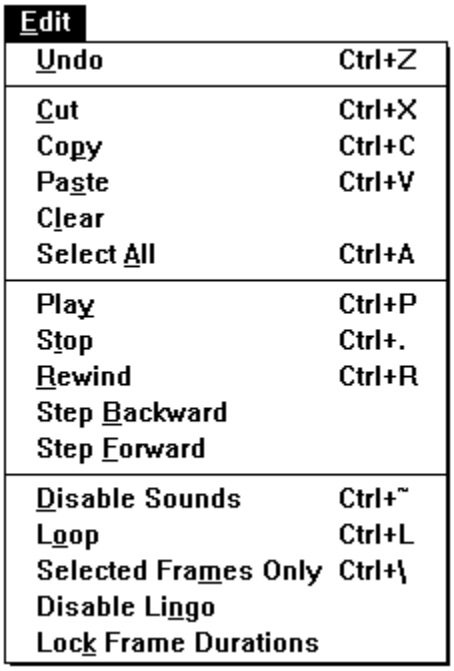
Exit command

The Exit command (Control-Q or Alt-F4) quits Director.

The clipboard is saved when you quit, and its contents are available for use in another application.

Edit menu

Click a menu command for more information:



The Edit menu contains standard commands for editing. It also includes control panel commands.

Undo command

Undo (Control-Z) reverses your last action. Since it works only on your last action, you must use it before making any other change to the animation. Undo works with most commands you use while writing, drawing, and animating.

Cut command

The Cut command (Control-X) removes the selected object from its current location and places it on the clipboard. It can then be pasted to another location.

Copy command

The Copy command (Control-C) makes a copy of the selected colors, text, art, or sequence of art and places that copy on the clipboard.

Paste command

The Paste command (Control-V) pastes the contents of the clipboard in a selected location.

Clear command

Clear removes the selected item or sequence of items, without saving it to the clipboard. Clear is used when you have no further use for a particular piece or sequence of art or text and when you do not want to affect what is already in the clipboard.

Delete is the keyboard shortcut for this command.

Select All command

Select All (Control-A) highlights all the selectable items in the active window.

To create a new animation with the same cast members, choose Select All in the score window, and then choose Clear. This clears all the score window notation while preserving the cast members. Your animation is erased, but the cast window is unchanged.

Play command

The Play command starts the movie. If you press the Shift key while choosing Play, the menu bar is hidden and the stage is cleared of all open windows as the movie plays; if the Director window is maximized, the menu bar is hidden as well.

Keyboard shortcut: Control-P or is the keyboard shortcut for this command. The Enter and plus (+) keys on the keypad toggle between Play and Stop.

Stop command

The Stop command halts the movie.

Keyboard shortcuts: Control-period (.) stops the movie. The keypad + key toggles between Play and Stop.

Rewind command

Rewind moves the playback head back to frame 1. If the animation is playing, it also stops.

Keyboard shortcut: Control-R or keypad 0 rewinds the movie.

Step Backward command

Step Backward steps the movie backward one frame at a time.

Keyboard shortcut: Control-left arrow or keypad 1 steps the movie backward.

Step Forward command

The Step Forward command advances the movie forward one frame. When using the step recording technique it can be used to record cast members to the next frame of animation.

Keyboard shortcut: Control-right arrow or keypad 3 steps the movie forward.

Disable Sounds command

If selected, turns your movie's sound off. By default, this option is off.

Keyboard shortcut: Control-~ or keypad 7 disables sounds.

Loop command

If selected, causes the movie to repeat continuously when played. When the movie reaches the last frame, it automatically starts again from frame 1. By default, this option is on.

Keyboard shortcut: Control-L or keypad 8 turns looping on or off.

Selected Frames Only command

If selected, lets you play a selected portion of a movie. This is convenient if you are working on just one part of a movie.

To play a portion of a movie, open the score and select the frames to be played. Choose Selected Frames Only and play the movie.

When a portion of the score has been marked as selected frames, a green bar appears at the top of the score over the selected frames.

Turn off Selected Frames Only when you want to return to normal play mode.

This command is dimmed if no frames are selected in the score.

Keyboard shortcut: Control-\ turns Selected Frames Only on or off.

Disable Lingo command

This command lets you ignore scripts during playback. If this command is not selected, Director executes all scripts during playback. This is the default. If this command is selected, Director ignores all scripts in the movie during playback.

This command is useful whenever you want to control whether interactivity is on or off during playback.

Lock Frame Durations command

When a movie's playback rate is locked, Director plays the movie using previously recorded frame durations. For frames without recorded durations, Director uses the current tempo. If checked, this command allows you to lock the movie's playback speed so that it will play back at the same speed on all types of computers.

Choosing this command is the same as clicking the Lock button in the Control Panel to lock frame durations.

To unlock the playback rate, remove the check from this command, click the Lock button in the control panel, or make an editing change to your movie, such as adjusting the tempo, or adding or deleting a frame in your movie. Unlocking the playback rate lets you record frame durations as you play the movie.

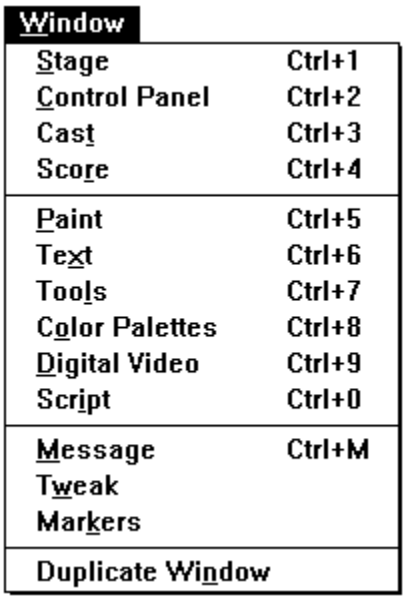
Locked movies will not play faster when played on a faster computer, but may still play slower if played on a slower computer.

Tip: You might want to clear all recorded frame durations from the movie if you want to only record frame durations for a section of the movie and lock them. To clear all recorded frame durations, press Alt while choosing this command or while clicking the Lock button. During playback, frames that don't have a recorded duration instead display "--" .

Window menu

The commands in the Window menu open and close Director's authoring windows. Open windows have checkmarks next to their names in the menu and the currently active window's name is underlined.

Click a window menu command for more information (or information on that window):



At the bottom of the menu is a list of all open text, script, and digital video windows, sorted by category, and sub-sorted by cast member number. Choosing a window opens it and brings it forward. If the window is open, but not forward, choosing it from the Window menu brings it to the front. If the window is already in front, Director hides it.

Director automatically hides any open windows if you choose the stage window from the Window menu.

Stage

The stage is the backdrop against which all Director animation appears. The size of the stage is set with the Preferences command on the File menu. In most cases, the edges of the stage window extend to the edges of your screen, so you can use all of the monitor for your movie. Movies continue to play on the stage when other windows are open and active. This permits you to study the movie in the score, for example, while keeping an eye on the stage.

The Stage command on the Window menu brings the stage to the front of the screen. It temporarily hides open windows; if the Director application window is maximized, it also hides the menu bar. Use the keyboard shortcut Control-1 to toggle back and forth between seeing the stage with the other open windows and the menu bar or just the stage.

Related topic: Resizing sprites on the stage

Resizing sprites on the stage

When a sprite is on the stage, clicking it displays a border around it, with a resize handle at the border's right edge. Drag the resize handle to change the sprite's size on the stage.

Control panel

Click to see an illustration.

Click the name of a control panel button for more information:

- Rewind
- Step Backward
- Stop
- Step Forward
- Play
- Disable Sounds
- Loop
- Selected Frames Only

Click the name of a control panel indicator for more information:

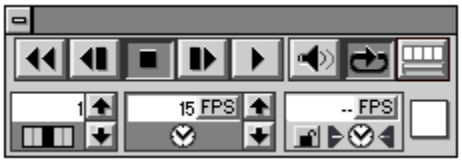
- Frame counter
- Tempo display
- Actual display
- Stage background

Open the control panel by choosing Control Panel from the Window menu, or by pressing Control-2.

The control panel has controls similar to a VCR. Use it to play, stop, step forward or backward, or rewind your movie. The control panel is also used to loop animation, set tempo, turn sound on and off, lock the movie's playback rate, and change the color of the background. The control panel indicates the current frame number, the current tempo, and the actual duration of the current frame.

To move the control panel, drag it by its title bar.

The control panel buttons have corresponding commands on the Edit menu.


1
15 FPS
-- FPS

Rewind button (control panel)

Rewind (Control-R) resets the movie to frame 1. If the movie is playing, clicking Rewind stops the movie and rewinds it to the first frame.

Step Backward button (control panel)

Step Backward (Control-left arrow) steps the movie backward one frame at a time. Press and hold down this button to step backward continuously.

Stop button (control panel)

Stop (Control-.) halts the movie.

Step Forward button (control panel)

Step Forward (Control-right arrow) advances the movie one frame at a time. Press and hold down this button to advance continuously.

If a score channel is in step-recording mode, stepping the movie forward also copies the contents of the current frame to the next frame.

Play button (control panel)

Play (Control-P) starts the current movie. If you press the Shift key while clicking Play, Director hides any windows that are open until you stop the movie. If the Director application window is maximized, Director also hides the menu bar.

Disable Sounds button (control panel)

Disable Sounds (Control-~) turns your movie's sound on and off. Click the button to alternate turning the sound on or off. When the button is pressed, sound is off and when it is not pressed, the sound is on.

Selected Frames Only (control panel)

Selected Frames Only (Control-\) lets you play a portion of a movie. First, open the score and select the frames to be played. Click Selected Frames Only and then play.

A green bar at the top of the score window in the area that contains the frame numbers indicates which frames have been marked for Selected Frames Only. Clicking Selected Frames Only again turns the selection off.

This button is dimmed if you haven't selected any frames in the score.

Loop (control panel)

Loop (Control-L) causes the movie to repeat continuously within the entire range of frames provided that the Selected Frames Only button is not pressed. When Selected Frames Only is pressed, the movie loops within the range of selected frames. When the movie reaches the last frame of the range, it starts again from the first frame. Loop is in effect when its button is pressed. Loop is on by default.

When you click the Loop button to turn it off, the loop symbol is replaced with the "once through" symbol.

Frame counter (control panel)

The frame counter displays the number of the frame currently on the stage. Click the up arrow to advance the movie or click the down arrow button to rewind the movie, frame by frame. Press and hold down an arrow button to rapidly advance or rewind the movie. To go to a specific frame number, click into the frame counter field, type a frame number, and press Enter.

Tempo display (control panel)

The tempo display shows the tempo of the current frame (which may be different from the default tempo). The number displayed indicates the upper limit of the speed of your movie in frames per second (FPS). Click the tempo mode to display the tempo in seconds per frame (SPF). Seconds per frame measures the duration, in milliseconds, of a frame. The display mode you choose is saved with the movie.

To change the tempo, click or press the up or down arrow button. To use a specific tempo, click into the tempo display area, type a tempo, and press Enter. If you are entering a tempo in frames per second (FPS), you must enter a whole number. If you are entering a tempo in seconds per frame (SPF), you must type a period (.) before entering a value.

It is recommended that you always enter a tempo setting in the first frame in the tempo channel.

If there are no tempo settings in the tempo channel, the control panel displays the default tempo. (If there is a tempo setting in the tempo channel, the control panel displays the tempo of the current frame.)

Director will attempt to play the movie at the specified tempo setting, but the movie may play slower on slower machines.

Actual display (control panel)

If a movie is playing, the Actual display shows the actual duration of the previous frame in frames per second (FPS), including the time necessary to execute any Lingo scripts (except `Exit Frame` scripts). If the movie is stopped, the Actual display shows the duration of the current frame. Frames that don't have a recorded tempo value display "--" instead of a value for the actual tempo.

If the movie is locked (using the Lock button in the control panel or using the Lock Frame Durations command in the Edit menu), the Actual display shows the previously recorded frame durations.

Since not all computers are equally fast, you can step through the movie frame-by-frame and compare the actual frame durations to the tempos you've set for the movie. This lets you see which frames your computer is not keeping up with. Comparing actual frame durations to tempos you've set is especially useful if you author on a fast computer and want to test playback on a slower computer.

Click the Actual mode button to change the way actual durations are displayed:

- FPS displayes the duration of each frame in frames per second.
- SPF displays the duration of each frame in seconds per frame.
- Sum provides a quick summary of elapsed seconds from the beginning of the movie to the current frame.
- The Est display provides a more accurate but slower calculation of elapsed time, and is useful if you want to include transitions and palette changes in determining frame durations. Computing estimated frame durations can reduce playback speed, so dont leave the Actual display in Est mode while playing the movie.

To lock the movie's playback rate so that it plays using previously recorded frame durations, click the Lock button.

Clicking the Lock button is the same as choosing Lock Frame Durations in the Edit menu.

Clear all recorded frame durations from the movie if you want to record frame durations for a section of the movie and lock them. To clear all recorded frame durations, press Alt while clicking the Lock button. During playback, frames that don't have a recorded duration instead display "--" .

Tips:

- Clear all recorded frame durations from the movie if you want to record frame durations for a section of the movie and lock them. To clear all recorded frame durations, press Option while clicking the Lock button. During playback, frames that don't have a recorded duration instead display "--" .
- To find frames that have longer durations than the current tempo, set both the Tempo and Actual displays to show seconds per frame (SPF); then step through the range of frames, and look for any frame whose actual duration is longer than the current tempo.

Stage background (control panel)

The stage background color chip lets you change the stage's background color. On a color computer you can use the background control to change the color of the background to any color in the current palette. To display the palette, position the pointer on the background color chip and click the mouse button. Then choose a background color from the pop-up palette.

Cast window

The cast window lets you view the cast members used in a movie. It is a database of graphics, sounds, color palettes, Lingo scripts, buttons, digital video movies, and text used in a Director movie.

Click to see an illustration.

Click a topic for more information:

Identifying cast members: shared & with scripts
Moving cast members within the cast window
Placing cast members on the stage
Placing cast members over time
Creating a film loop

Place button
Previous, Next arrows
Cast Member Info button
Open Script button
Cast member number
Cast member name

Open the cast window by choosing Cast from the Window menu or by pressing Control-3.

The cast window is organized by cast member positions. Each cast member position is identified by a number and, optionally, a name. For every occupied position in the cast window, a thumbnail image is displayed that represents the cast member's type.

Each cast member position is identified by a number, and optionally, a name.

A movie's cast can contain up to 32,000 cast members. You control the row width and the number of visible rows using the Cast Window Options command in the Cast menu.

Clicking any cast member selects it. Select a range of cast members by Shift-clicking. Control-click selects multiple non-adjacent cast members. Individual cast members can be cut, copied, pasted, or cleared from the cast window.

Tip: You can select several text cast members in the cast window and change their font, size, and style using the commands on the Text menu. You can change the color of a text cast member using the foreground and background color chips in the tools window.

For every occupied position in the cast window, an icon is displayed that represents the cast member's type. Film loop and movie cast members are represented by the same thumbnail icon.

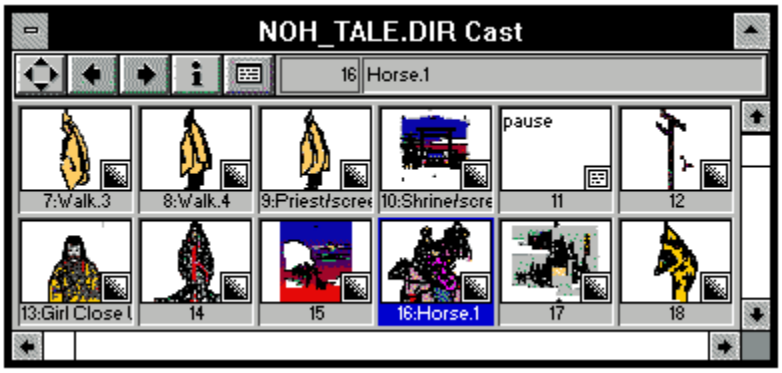
An embedded cast member is a physical part of a Director movie: the information about the cast member is saved in the same file as all the other information that makes up the movie. A linked cast member, on the other hand, is contained in a separate file; all that's saved with the Director movie is the information that identifies where the file that contains the cast member is located.

Thumbnails are redrawn only when you edit a cast member in Director or change the thumbnail size in the Cast Window Options dialog box. If the cast includes a linked file, and you edit the original file, Director doesn't redraw the thumbnail unless you change the thumbnail size.

Double-clicking a cast member is a shortcut for opening the paint window for a graphic cast member, the text window for a text cast member, the color palettes window for a palette cast member, the script

window for a script cast member, and the digital video window for a digital video cast member.

Related topic: Cast menu


NOH_TALE.DIR Cast
16
Horse.1
7:Walk.3
8:Walk.4
9:Priest/scree
10:Shrine/scre
pause
11
12
13:Girl Close
14
15
16:Horse.1
17
18

Identifying cast members: shared & with scripts

Cast members with scripts display a script indicator icon in the lower left corner. To control whether or not Director displays a script indicator icon in the cast, use the Indicate Cast Members with Scripts checkbox in the Cast Window Options dialog box.

The names of cast members that are part of a shared cast movie appear in italics. Director also italicizes the names of shared cast members in the digital video, paint, script, and text windows, and the names of thumbnail images in the upper-left corner of the score window.

You must name a shared cast movie "SHARED.DIR" to have Director recognize it as a movie that contains cast members that can be shared by other movies.

Place button (cast window)

Lets you move the selected cast member to the stage or score, or move it within the cast. When cast members are moved within the cast window, the score window is updated with their new position. When you press and hold down this button, the cursor changes to an open hand to let you drag one or more selected cast members. Using this button is the same as clicking and dragging a selected cast member in the cast window. Use this button to move selected cast members that may not currently be visible in the cast window, if you've scrolled to a new location.

To use the Place button to move a cast member to a new location within the cast window:

1. Select the cast member you want to move.

2. Scroll to the new location in the cast window where you want to insert the selected cast member.

3. Drag from the Place button to the new location in the cast window. As you drag within the cast window, a blinking insertion bar indicates the location where the cast member will be inserted.

4. Release the mouse to insert the selected cast member at the new location.

Previous, Next arrows (cast window)

These arrows let you navigate to the previous or next cast member, skipping over empty cast members.

Tip: The keyboard equivalents for Previous and Next are Control-Shift-left arrow and Control-Shift-right arrow.

Cast Member Info button (cast window)

Displays the Cast Member Info dialog box for the selected cast member. If the selection consists of more than one cast member, the dialog box displays the number of cast members selected, and their total size and purge priority.

Tip: Choosing Cast Member Info from the Cast menu or pressing Control-I are the same as clicking this button.

Open Script button (cast window)

Opens a new script window or makes an existing script window active for the selected cast member, or for the first selected cast member if more than one are selected. If the selected cast member has no script associated with it, clicking this button opens a new script window. This is the same as clicking the Script button in the Cast Member Info dialog box.

Tip: Choosing Open Script from the Cast menu or pressing Control-' (apostrophe) is the same as clicking this button.

Cast member number (cast window)

Displays the position of the selected cast member in the cast window or the position of the first selected cast member in a multiple selection.

Tip: When the cast window is front-most, typing the number of an existing cast member automatically scrolls the cast window to the cast member's location and selects it. After you've stopped typing for 1/2 second, the cast window will scroll to show the new selection. Alternatively, you can press Enter immediately after typing the number to scroll the cast window instantly. (You can also press Backspace to delete the last digit you typed.)

Cast member name (cast window)

Displays the name of the selected cast member, or the first selected cast member in a multiple selection. Click and type into the name area to enter or edit the name of a cast member. Press Return or Enter to confirm your changes.

Tip: Double-clicking a cast member is a shortcut for opening the paint window for a graphic cast member, the text window for a text cast member, the color palettes window for a palette cast member, the script window for a script cast member, and the digital video window for a digital video cast member. Holding the Alt key while double-clicking a text, script, or digital video cast member is a shortcut for opening a new text, script, or digital video window.

Moving cast members within the cast window

You can rearrange cast members within the cast window by dragging them to a new location.

To move one or more cast members:

1. In the cast window, click to select the first cast member.

   The pointer changes to a hand shape.

2. Shift-click to select additional adjacent cast members; Control-click to select additional non-adjacent cast members.

3. Drag one of the cast members you just selected.

   Dragging any selected cast member drags the entire selection.

   As you drag within the cast window, a blinking insertion bar indicates the location where the selection will be inserted.

4. Release the mouse to insert the selected cast members at the new location.

   The cast members are inserted to the right of the insertion bar. Cast members following the insertion bar shift over to make room for the inserted cast members.

Note: You cannot drag a selection if it includes cast members that are part of a shared cast. You must open the shared cast movie to rearrange its cast members.

Placing cast members on the stage

Placing cast members on the stage adds them to the score in the available channels in the current frame. If you place more cast members than you have available channels in the current frame, Director warns you that it will ignore the extra cast members.

When you place cast members on the stage (or in the score), Director adds them as sprites. A sprite is an image of the cast member on the stage or in the score that contains information about the cast member at one point in time.

To place one or more cast members on the stage:

1. In the cast window, click to select the first cast member. The pointer changes to a hand shape.

2. Shift-click to select additional adjacent cast members; Control-click to select additional non-adjacent cast members.

3. Drag one of the cast members you just selected to the stage. Dragging any selected cast member drags the entire selection.

   As you drag the cast members over the stage, Director displays an outline indicating the size of the area enclosing all the cast members. (If you drag a sound or a palette, it does not display an outline on the stage, since such cast members have no size dimensions.)

   When you drop the cast members on the stage, Director activates step-record mode in the channels where the cast members were dropped.

Tip: The keyboard shortcut for placing selected cast members in the center of the stage is Control-Shift-L.

Placing cast members over time

By default, Director places multiple cast members on the stage so that they all occupy the same frame in the score. To place cast members on the stage over time instead of in the same frame, press Alt while dragging the cast members from the cast window to the stage.

As you Alt-drag the cast members over the stage, Director displays an outline indicating the dimensions of the first visual cast member in the selection, which will be placed in the current frame. The rest of the cast members in the selection are placed into adjacent frames.

If the score window is open while you are dragging cast members to the stage, Director outlines the cells in the score where the cast member selection will be dropped. When you Alt-drag cast members to the stage, Director prevents you from dropping cast members into occupied cells in the score. (If you instead drag the cast members directly from the cast window into the score, Director does not prevent you from dropping the cast members into occupied cells and overwriting existing sprites.)

If you drag a sound cast member to the stage or score, Director places it in sound channel 1 of the current frame, or in sound channel 2 if sound channel 1 is occupied. If both channels are occupied, the sound is not dropped. If you drag a palette cast member to the stage or score, Director places it in the palette channel of the current frame, and assigns it these default properties: Speed=30 (instant) and Type=Normal.

When you drag a cast member to the stage, **Director uses the first empty cell in the current frame that is closest to the current score selection**. (If you drag the cast member directly into the score, you can more precisely specify the cell that will contain the cast member.) You can force Director to use a certain set of empty cells by first selecting them in the score before dragging in the cast members.

To cancel the drag of a cast member to the score, drag the cursor over the menu bar or desktop so that the cursor changes to indicate that the location under the cursor is an invalid drop location.

Tips:

- The keyboard shortcut for placing selected cast members over time in the center of the stage is Control-Alt-L.

- You can also place selected cast members across time in the score by choosing Cast to Time in the Cast menu.

Dragging cast members to the score

Dragging cast members into the score places them into a range of cells and activates step-record mode in the channels where the cast members are dropped.

To place one or more cast members into the score:

1. Click to select the first cast member in the cast window.

   The pointer changes to a hand shape.

2. Shift-click to select additional adjacent cast members; Control-click to select additional non-adjacent cast members.

3. Drag one of the cast members you just selected to the score. Dragging any selected cast member drags the entire selection.

   As you drag the cast members over the score, Director outlines the range of cells into which the cast members will be dropped.

Click for more information: More details on dragging cast members to the score

Details on dragging cast members to the score

By default, Director places cast members in channels in the score, so that they all occupy the same frame. To place cast members into the score across time instead of in the same frame, press and hold down Alt while dragging the cast members into the score.

As you drag cast members over the score, Director may restrict you to specific regions of the score depending on the type of cast members in the selection. For example, if you drag a sound cast member into the score, Director restricts you to the two sound channels. If you are dragging different types of cast members, Director lets you only drop them into cells of the appropriate type.

You can drag this type of cast member	**Into this part of the score**
Bitmap, PICT, Text, Button, Shape, Film loop, Movie, Digital Video	Sprite channels 1-48
Sound	Sound channel 1, sound channel 2
Palette	Palette
Script	None

When you drop cast members into the score, Director fills the highlighted cells in sequential order by cast number. The first cast member in the selection of a particular type will be dropped into the left-most or top-most cell in the channel.

Any occupied cells in the highlighted region are overwritten by the cast members you drag in.

Creating a film loop

When you move selected frames from the score into the cast, Director creates a new film loop cast member in the cast window. This a shortcut for copying and pasting a score selection into the cast window.

To create a film loop cast member in the cast window:

1. In the score window, select the cells that contain the sequence of cast members that you want to use to create the film loop.

2. Drag the selection into the cast window.

   As you drag within the cast window, a blinking insertion bar indicates the location where the selection will be inserted.

3. Release the mouse button to insert the selection to the right of the insertion bar.

   Director prompts you to name the film loop.

4. Enter a name for the film loop and click OK.

   The film loop is inserted in the cast to the right of the insertion bar. Cast members to the right of the insertion bar will shift over to make room.

The score

Related topic: Score menu

The score contains the notation that describes your movie and is the primary tool for creating and editing animation. The score window contains a record of everything that happens on the stage.

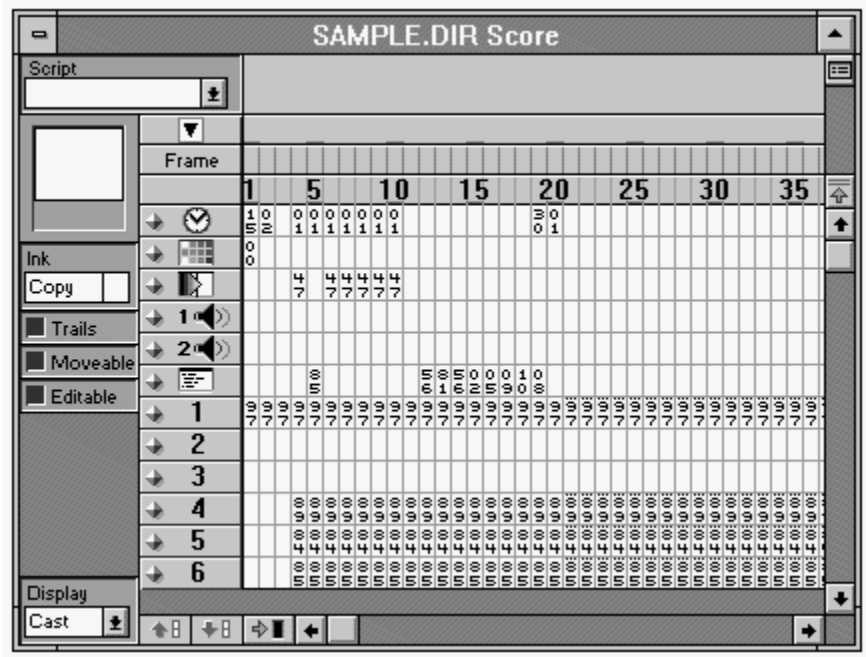
Click to see an illustration.

Click an element of the score window for more information:

- Script preview button
- Script pop-up menu
- Ink pop-up menu
- Display pop-up menu
- Trails checkbox
- Moveable checkbox
- Editable checkbox

Click a topic for more information:

- Score window basics
- Selecting cells
- Moving selections within the score window
- Moving around the score
- Applying color to cells
- Step-recording in a channel
- Real-time recording in a channel
- Turing a channel on and off


SAMPLE.DIR Score
Script
Frame
1
5
10
15
20
25
30
35
Ink
Copy
Trails
Moveable
Editable
Display
Cast
1
2
3
4
5
6

Score window basics

Open the score window by choosing Score from the Window menu, or by pressing Control-4.

The smallest unit in the score is a **cell**. Each cell contains information about one cast member at one point in time, called a **sprite**. When you select a cell, a small image of the sprite that occupies that cell appears in the upper left corner of the score window. Double-click the sprite's image to open a window in which you can edit the cast member.

The first five rows, called **channels**, keep track of special effects like tempos, palettes, transitions, or sounds. By setting effects in those channels, you can add sound to your movie, control the tempo, set and transition palettes, cycle colors, and add visual effects. The script channel is where you store scripts (instructions written in Lingo) that are executed when the movie reaches a particular location in the score.

The order of sprites in the 48 animation channels determines which sprites appear in the foreground and which appear in the background. Think of the stage as a pile of transparent sheets, 48 sheets thick. The sprite that occupies a channel closer to the top of the score is like the object drawn on the last sheet of acetate. It appears behind any sprite in a channel closer to the bottom of the score.

Sprites in channels closer to the bottom of the score appear in front of, or take priority over, those in channels closer to the top of the score.

Note: A sound in sound channel 2 overrides a sound in sound channel 1. A sound that's already playing in either sound channel overrides the sound in a digital video movie, and it also prevents the digital video movie's sound from playing even after the sound in the sound channel has stopped. Once the sound in a digital video movie has started, however, it overrides a sound in either sound channel.

A **frame** contains information about everything you see on the stage when you stop a movie at the moment that corresponds to the frame. Like a frame in a movie, a Director frame shows what each sprite is doing on the stage at one point in time. Each frame is numbered.

The score also contains small triangular **markers** to help you coordinate the comments in the markers window with specific frames of your movie or identify frames of your movie for printing. You can drag any number of markers from the left side of the score window and position them in any frame. Use this feature to add comments, speaker notes, or storyboard notes to specific frames of animation.

When you position a marker in the score window, an insertion point appears to the right of the marker so you can type a short comment for that marker. Use the markers window to review and edit marker comments.

Display pop-up menu (score window)

The Display pop-up menu lets you change the type of information displayed in each cell of the score. It is located in the lower left corner of the score window. Use it to view different types of notation in the score. By default, the score displays cast notation.

The **Extended** display enlarges the score and displays all of the cast member information at once.

- For the animation channels, the Extended display enlarges the score and displays all of the sprite information at once.
- Use the Score Window Options command to choose what information is contained in the Extended display.
- If you have chosen to view the cast members by name in the cast window (using the Cast Window Options command), the Extended display shows the first few letters of the cast member's name instead of the cast member's cast number, if a name exists for that cast member.
- For the tempo channel, the extended notation displays the settings specified in the Set Tempo dialog box.
- For the palette channel, the extended notation displays the settings specified in the Set Palette dialog box.
- For the transition channel, the extended notation displays the settings specified in the Set Transition dialog box.
- For the sound channel, the extended notation displays the cast member number.

The **Ink** display indicates which ink effect has been applied to each sprite.

The **Blend** notation shows the blend percentage that's applied to the sprite using the Set Sprite Blend command in the Score menu.

The **Cast** display depends on how you have chosen to view cast members in the cast window using the Cast Window Options command. If the cast window displays cast members by octal number, decimal number, or by number:name, the Cast display shows the last two digits of the cast member's position number. If the cast window displays cast members by name, the Cast display shows the first two letters of the cast member's name, if a name exists for that cast member. Cast display is the default.

The **Motion** display shows the direction of the cast member's movement and whether or not a new cast member appears in that cell. It is a useful way to view the score if you are tweaking a cast member to line up in a particular position relative to the cast member's previous position.

The **Script** display shows the location and number of the script associated with each cell in the score. Each script has an identifying number associated with it. Cells without scripts display 00. Sprites with cast scripts display a plus (+) sign in the cell.

Selecting cells

Drag across cells to select them. If you have Playback Head Follows Selection checked in the Score Window Options dialog box, the movie will advance or rewind as you select in the score. You can see the frames on the stage as you select them in the score.

Drag across cells or Shift-click to select multiple adjacent cells.

Double-click the channel number to select all the cells in a channel. Double-click the channel number and drag down to select all the cells in adjacent channels.

Drag across the frame numbers at the top of the score window to select all the cells in multiple frames.

Click a frame number at the top of the score window to select all the cells in a frame.

Double-click an occupied cell to select all adjacent cells that contain the same cast member.

Control-click for discontinuous selections.

The consequences of each type of selection are very important. For example, if only one cell in the score is selected, only one sprite in one frame of the movie is affected by the commands you choose. On the other hand, if you select a number of cells in a channel, you can select a sequence of sprites over a period of time. Then, any changes you make affects that sequence.

Whenever you select less than all the channels in a frame, from a single cell to a range of cells over one or more channels, you are editing and affecting space. Cutting, copying, pasting, or clearing either adds or removes elements within a time sequence but does not affect the length of the sequence.

Moving selections within the score window

You can move one or more selected cells to a new location within the score or to the cast. (You cannot drag score selections to the stage.)

To move selected cells to a new location within the score:

1. Select the cells. The cursor changes to a hand.

2. To select additional frames, Shift-click them.

3. Drag the frames to the new location. As you drag, an outline of your selection appears under the cursor. Dragging to the edge of window will auto-scroll the window. To instead move a copy of the selection, press the Alt key while dragging the selection.

4. Release the cursor to place the selection at the new location.

   The contents of the frames you move replace the contents of existing frames at the new location.

   If Playback Head Follows Selection is checked (using the Score Window Options command in the Score menu), the selected frames will be displayed on the stage as you drag the selection in the score.

   To cancel the drag, move the cursor over the menu bar or choose Undo.

Moving around the score

The **playback head** at the top of the score indicates which frame of the movie is currently on the stage. The playback head travels in the scratch bar as the movie plays. You can drag the playback head or click to move to a specific frame. If Playback Head Follows Selection is checked in the Score Window Options dialog box, the playback head travels with any selection you make in the score.

Another way to move your view of the score to the current frame of your movie is the **jump** button, located in the lower left corner of the score window. Whenever you click this button, Director brings the playback head into view and your view of the score jumps to the frame that is currently on the stage.

The **jump to top** button at the top right side of the score window scrolls the score window to the script channel when you click it. If the score channel is already visible, clicking the jump to top button scrolls the score window to the top of the effects channels.

An additional aid to navigation is the **frame counter** that pops up in the middle of the horizontal scroll bar when you drag the horizontal scroll box. It indicates how far you have advanced or rewound the view of the score as you drag the scroll bar.

The **Shuffle Forward** and **Shuffle Backward** buttons let you move a selection of sprites up or down in the score, to change their foreground priority on the stage. Shuffle Forward switches a range of cells you've selected in a channel with the cells in the channel below it. The sprites associated with the selected cells move in front of the other sprites. Shuffle Backward switches a range of cells you've selected in a channel with the cells in the channel above it. The sprites associated with the selected cells move behind the other sprites.

Tip: The keyboard shortcut for Shuffle Forward is Control-Shift-down arrow; the keyboard shortcut for Shuffle Backward is Control-Shift-up arrow.

Applying color to cells

The cell color selector lets you apply color to score cells. The color only affects the cells in the score; it does not affect how sprites appear on the stage.

To apply color to cells, select them and click a color from the cell color selector. You can apply color to empty or occupied cells. The color is carried with the cell if you move it or copy it. Applying color to cells can help you identify different sections in a large score.

You show or hide cell colors using the Colored Cells option in the Score Window Options dialog box. If you've already applied color to cells, hiding cell colors doesn't remove their color. Score window scrolling performance is faster if you hide cell colors.

Script preview button (score window)

The Script Preview button displays the first two lines of the script associated with one or more selected cells in the script channel. If the selection has no script associated with it, the Script Preview button is blank. Clicking the Script Preview button opens a new script window. (If a script window is already open, the new script window replaces the existing one; to open a new script window in addition to the existing one, press Alt as you click the Script Preview button.)

To hide the script pop-up menu and the Script Preview button, click the Show/Hide button.

Script pop-up menu (score window)

The script pop-up menu lists all the score scripts used in the current movie. The script pop-up menu displays the script number associated with a selected frame in the script channel. If the selected cell has no script associated with it, the pop-up menu is blank. You can use this pop-up menu to apply existing scripts to areas of the score. Select the cells you want to apply the script to; then choose the script you want from the script pop-up menu.

Choose the New option in the script pop-up menu to create a new score script. Choose the 0 option to remove the script from the selected cell.

For a complete description of how to create and use scripts, refer to the *Using Lingo* manual.

Related topic: Script window

Ink pop-up menu (score window)

The Ink pop-up menu lists the inks you can apply to a sprite. When you place a cast member on the stage, usually the cast member's sprite (the image of the sprite that appears on the stage) looks the same as the cast member in the cast window. You can use inks to change the way a sprite looks in a variety of ways. Probably the most common use, for example, is to make the opaque bounding box that appears around some sprites transparent. (An opaque bounding box appears around a non-rectangular sprite when you place it on a background that isn't white.) You can also use inks to make one color in sprite transparent, to make the entire sprite semi-transparent, to change black pixels to white and white to black, and to combine the color of the sprite with the color of the background to create a new color.

Click for more information on an ink effect:

- Copy
- Matte
- Bkgnd Trans.
- Transparent
- Reverse
- Ghost
- Not Copy
- Not Transp
- Not Reverse
- Not Ghost
- Mask
- Blend
- Darkest
- Lightest
- Add
- Add Pin
- Subtract
- Subtract Pin

Related topic: About score window inks

About score window inks

An ink modifies the way a sprite looks pixel by pixel. For example, if two pixels next to each other are two different colors, an ink might leave one pixel as it is and make the other transparent. There are four things that determine how an ink changes a particular pixel in a sprite:

- The pixel's source color (the original color of the pixel in the cast member).
- The foreground color that's selected in the tools window. (The foreground color in the paint window has no effect.)
- The background color that's selected in the tools window. (The background color in the paint window has no effect.)
- The pixel's destination color (the color currently displayed on the stage at the spot where the pixel will appear).

To apply an ink:

1. In the score, select the cells that contain the sprite (or sprites) you want to apply the ink to.

   When you drag a cast member to the stage, the default ink is the last ink you chose in the Ink pop-up menu. The preselected default ink is Copy.

2. If you want to change the foreground or background color of the sprite, use the foreground or background color chip in the tools window (not the paint window) to choose the color you want.

   The effect that changing the foreground or background color produces differs depending on the ink you use. For example, in one case changing the foreground color might simply change the color of the sprite on the stage. In another case, changing the background color might identify the color you want to make transparent.

3. Select the ink you want from the Ink pop-up menu drop-down list.

   Choose an ink carefully, since some inks (such as Matte, Background Transparent, and Blend) can result in decreased performance when animating a cast member.

Copy (Ink pop-up menu of the score window)

Copy is the default ink.

When the foreground color that's selected in the tools window is black and the background color that's selected there is white, a sprite looks the same as the cast member it's based on.

When you select a foreground color other than black, Director changes any pixel that's black to the foreground color. (Remember that it's the sprite's appearance that's changing, not the cast member's: when Director displays the sprite on the stage, it replaces any pixel in the cast member that's black with the foreground color.) If you select a background color other than white, Director replaces any pixel that's white with the background color. (Copy wasn't designed to allow you to change the color of pixels that are a color other than black or white. If a cast member contains colors other than black and white, the effect of selecting a foreground color other than black and a background color other than white may vary from system to system.)

No matter what foreground and background colors are selected, if the cast member isn't rectangular, an opaque bounding box appears around the sprite when it's in front of another sprite or a background that isn't white. If you want to make the bounding box transparent, use Matte rather than Copy.

Sprites with the Copy ink animate faster than sprites with any of the other inks.

Matte (Ink pop-up menu of the score window)

Matte works just like Copy except that it makes the bounding box around a sprite transparent. Matte uses more memory then Copy because Director has to create a duplicate of the cast member. Sprites youve applied Matte to also animate more slowly than other sprites.

Bkgnd Trans. (Ink pop-up menu of the score window)

Background Transparent makes pixels that match the background color you've selected in the tools window transparent. (Changing the foreground color in the tools window has no effect.) Background Transparent, like Matte, makes the bounding box around a non-rectangular sprite transparent.

Sprites you've applied Background Transparent to may animate more slowly than other sprites.

Transparent (Ink pop-up menu of the score window)

Transparent makes white pixels transparent.

If you select a foreground color other than black in the tools window, Director changes pixels that are black to the foreground color.

Transparent was designed to work with black-and-white cast members. Consequently, only black pixels--or the foreground color you assign to them--are completely opaque and only white ones are completely transparent. (Changing the background color in the tools window to a color other than white has no effect.) Transparent produces interesting effects with cast members that contain other colors, but those effects may vary from system to system.

Reverse (Ink pop-up menu of the score window)

Reverse turns black pixels white when they're displayed on a black background (otherwise, they're black). It makes white pixels transparent. (Changing the foreground and background colors in the tools window has no effect.)

Reverse was designed to work with black-and-white cast members on a black-and-white background. It produces interesting effects with cast members and backgrounds that contain other colors, but those effects may vary from system to system.

Ghost (Ink pop-up menu of the score window)

Ghost changes black pixels to the background color selected in the tools window and makes white pixels transparent.

Ghost was designed to work with black-and-white cast members on a black-and-white background. If you use Ghost in a black-and-white environment (in which the foreground color is always black and the background color is always white), black pixels are invisible until they're in front of a black background. When they move in front of a black background, they appear white. White pixels are transparent, so they never show up at all. The effect is of a sprite that appears when it moves in front of a black background, and then vanishes when it passes over a white background. If you change the background color in the tools window to a color other than white, you can produce the same effect on a background that matches the color you've chosen. Ghost produces interesting effects with cast members and backgrounds that contain other colors, but those effects may vary from system to system.

Not Copy/Transp./Reverse/Ghost (score Ink pop-up)

Not Copy works the same as Copy except that it changes black pixels to the background color and white pixels to the foreground color.

Not Transparent is the mirror image of Transparent:

- It makes black pixels--not whites ones--transparent.

- If you select a foreground color other than black in the tools window, Director changes pixels that are white--not black--to the foreground color.

Not Reverse is the mirror image of Reverse: it turns white pixels black when they're displayed on a white background (otherwise, they're white). It makes black pixels transparent.

Not Ghost is the mirror image of Ghost: it changes white pixels to the background color selected in the tools window and makes black pixels transparent.

Mask (Ink pop-up menu of the score window)

Mask is similar to Matte: Matte produces an opaque sprite inside a transparent bounding box by producing a mask that's the exact shape of the original cast member. Mask, like Matte, makes the bounding box around the sprite transparent, and it also lets you make any other part of the sprite transparent as well.

For Mask to work properly you must create a duplicate of the cast member to be masked and place it in the cast window in the cast member position to the immediate right of the cast member being masked. The mask must be drawn in 1-bit black and white. Make any part of the mask you want to be opaque black and any part you want to be transparent white.

Mask, like Matte, uses more memory then Copy because it requires a duplicate of the cast member. Sprites youve applied Mask to also animate more slowly than other sprites.

Blend (Ink pop-up menu of the score window)

Blend produces a semi-transparent sprite.

Use the Set Sprite Blend command on the Score menu to set the level of transparency anywhere from 0% (completely transparent) to 100% (completely opaque).

Sprites youve applied a blend to animate more slowly than other sprites.

Darkest (Ink pop-up menu of the score window)

Darkest compares the color of a pixel with the color of the spot on the stage where the pixel will appear and then displays whichever color is darker.

Lightest (Ink pop-up menu of the score window)

Lightest compares the color of a pixel with the color of the spot on the stage where the pixel will appear and then displays whichever color is lighter.

Add (Ink pop-up menu of the score window)

Add creates a new color by adding the color value of a pixel to the color value of the spot on the stage where the pixel will appear. If the sum exceeds the highest possible color value, Director wraps the remainder around the color scale to determine the color it will display.

Add Pin (Ink pop-up menu of the score window)

Add Pin is similar to Add: Director adds the color value of a pixel to the color value of the spot on the stage where the pixel will appear. However, if the sum exceeds the highest possible color value, Director drops the remainder.

Subtract (Ink pop-up menu of the score window)

Subtract creates a new color by subtracting the color value of a pixel from the color value of the spot on the stage where the pixel will appear. If the difference is less than zero, Director subtracts the remainder from the highest possible color value to determine the color it will display.

Subtract Pin (Ink pop-up menu of the score window)

Subtract Pin is similar to Subtract: Director subtracts the color value of a pixel from the color value of the spot on the stage where the pixel will appear. However, if the difference is less than zero, Director uses the lowest possible color value.

The best way to see how each ink effect works is to play the sample movie Ink Effects. To find it, go to the Director directory, open the Tutorial directory, and then open the Learning directory. Double-click the icon labeled Ink_FX. When you're finished watching the movie, press Control-period to stop it.

Trails checkbox (score window)

If Trails is checked, the selected sprite remains on the stage, leaving a trail of images along its path as the movie plays. If Trails is unchecked, the selected sprite is erased from previous frames as the movie plays. The checkbox also reflects the current selection. If the current selection includes sprites that don't all have the same setting, the Trails checkbox is shaded gray.

Tip: The dashed line in the center of the cells that appears when you view the score using the Ink or Extended display indicates that those cells have the Trails effect applied to them. You can also see if a cast member has trails by selecting the cast member on the stage or in the score and looking at the Trails checkbox.

Moveable checkbox (score window)

This option is only available if you select one or more cells in the sprite channels in the score. If Moveable is checked, users can move the selected sprite(s) around on the stage during playback. If checked, the "Moveable" setting is in effect only when the playback head is executing those frames that contain the moveable sprites. The checkbox also reflects the current selection. If the current selection includes sprites that don't all have the same setting, the Moveable checkbox is shaded gray.

Editable checkbox (score window)

This option is only available if you select one or more text sprites. If Editable is checked, users can edit the selected text sprites on the stage during playback. This option is convenient for making a text sprite editable in some frames, and noneditable in others. You can turn this setting off when it is no longer required. If checked, the "Editable" setting is in effect only when the playback head is executing those frames that contain the editable sprites. The checkbox also reflects the current selection. If the current selection includes sprites that don't all have the same setting, the Editable checkbox is shaded gray.

You can set a text cast member to always be editable using the Text Cast Member Info dialog box. If you set a text cast member to be editable in the cast, it is always editable when used in the score. Director ignores the score's Editable checkbox setting for the cast member.

Using the Lingo command `set the editableText of sprite to true` is the same as checking the Editable checkbox in the score.

Step-recording in a channel

Step-recording lets you create frame-by-frame animation in a channel. To activate step-recording in a channel, Alt-click the channel number in the score where you want recording to occur. Select additional channels by Alt-clicking their channel numbers.

Note: Step-record mode is automatically invoked if you drag cast members from the cast window to the score or the stage.

A step-recording indicator appears in the channels that are in step record mode.

When you choose Step Forward from the Edit menu or control panel, the sprite in the channel with the step-recording indicator is copied into the next frame of your movie.

Step-recording remains in effect until you Alt-click the channel number again, click out of the channel, drag the playback head, or click Rewind or Back Step.

Real-time recording in a channel (score window)

Real-time recording lets you move a sprite directly on the stage while recording its motion in a score channel. To activate real-time recording in a channel, press Control-Spacebar while clicking a cell in the desired channel. To activate real-time recording in the first empty score channel, just press Control-Spacebar.

An indicator appears next to the channel number to indicate that real-time recording will occur in that channel. Only one channel at a time can be used for real-time recording.

To record a cast member's path in the score, first select it in the cast window. Then hold down Control-Spacebar and click anywhere on the stage to begin recording the sprite's path. Director records the motion of the sprite in the score as you move it across the stage. Information in any of the other channels is copied as you record.

To stop real-time recording, release the mouse button.

Turning a channel on and off (score window)

Turning a channel off tells Director to ignore the channel during playback. By default, no channels are ignored (i.e., all channels are active). Clicking the button next to a channel turns the channel off, causing Director to ignore that channel when you play the movie.

If you turn the script channel off, Director ignores all scripts during playback. (This is the same as checking the Disable Scripts command in the Edit menu.)

Channel on/off settings are not saved with the movie.

Paint window

Related topics: Paint Window Options, Paint menu

Click to see an illustration.

Click a topic for more information:

- The paint window tools
- The Ink pop-up menu

- Destination color chip
- Foreground color chip
- Background color chip
- The pattern chip
- The line width selector
- The color resolution indicator

- Place button
- Add button

- Creating a new bitmap cast member
- Using rulers
- Zooming in and out
- Selecting colors and patterns
- Drawing with the shape tools

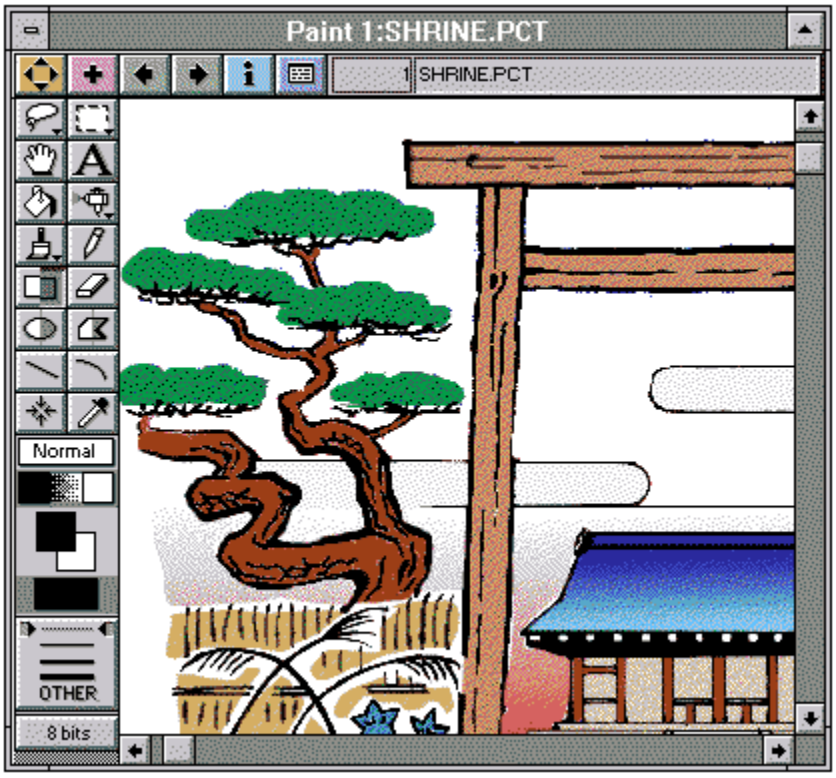
The paint window has a complete set of paint tools and inks you can use to create cast members for your movies. The paint and cast windows share a dynamic link. Anything you draw in the paint window becomes a cast member automatically and is displayed in the cast window. When you make a change to a cast member in the paint window, the image in the cast window is instantly updated--as is the cast member wherever it appears on the stage. Conversely, when you paste or import a new bitmapped cast member into the cast window, it appears in the paint window.

Open the paint window by choosing Paint from the Window menu, or by pressing Control-5. When you open the paint window, the Paint and the Effects menu appear on the menu bar.

Tip: A shortcut for opening the paint window is to double-click a bitmapped cast member on the stage or in the score or cast windows. The paint window opens with that cast member showing.

The paint window contains a complete paint program so you can create or edit cast members without leaving Director. The paint tools are contained in the tool palette at the left side of the window.

If you have a cast member set to a color depth other than 1 bit (black and white) or 8 bits (256 colors) or if you have Windows display driver set to something other than 256 colors (SuperVGA mode), you can duplicate, delete, or change the color depth of a cast member in the paint window, but you can't edit it in any other way.


Paint 1:SHRINE.PCT
1
SHRINE.PCT
Normal
OTHER
8 bits

Creating a new bitmap cast member

To create a new cast member, click the **Add** button. (The keyboard shortcut for Add is Control-Shift-A.)

Director creates a new, empty easel in which you can create the cast member. Whatever you draw is placed in the first available cast member position.

The previous and next buttons control which cast member is displayed.

Using rulers

The paint window has a set of vertical and horizontal rulers to help you align and size your artwork.

To use the rulers, choose Show Rulers from the Paint menu or press Control-Shift-K. The rulers appear at the top and left side of the paint window. The default unit of measurement is inches.

To change the unit of measurement, click the corner where the rulers meet. With each click the unit of measurement changes, first to centimeters, then to pixels, then to picas, then back to inches.

To change the location of the zero point, drag right or left along the ruler at the top of the window or up or down along the ruler at the side.

To remove the rulers from the paint window, choose Hide Rulers from the Paint menu or press Control-Shift-K.

Zooming in and out (paint window)

The Zoom commands on the Paint menu permit you to zoom in or out at four levels of magnification.

To zoom in on the image in the paint window, choose Zoom In (Control-+) from the Paint menu.

The image size increases to the first level of magnification. To increase the magnification, choose Zoom In again. The zoom centers on the last part of the image clicked in the paint window. To zoom in on a particular feature of the image, Command-click the image, or position the pointer over the feature before choosing Zoom In.

You can see a normal-size view of the image in the box in the upper right corner of the paint window. You can go back to the normal-size view any time by clicking inside the box.

Zoom Out reverses the direction of the zoom. To reduce a magnified view, choose Zoom Out (Control-minus) from the Paint menu.

Tip: Switch back and forth between normal size and the last level of magnification you chose by double-clicking the pencil tool or by pressing the Control key and clicking the art with any tool.

Selecting colors and patterns (paint window)

Colors and patterns are similar in the way you choose them and in the way you make use of them. You choose both colors and patterns from pop-up palettes (the pop-up colors palette isn't available if you have your monitor set to black and white), and you make use of both with the same set of tools.

The colors that appear on the pop-up palette come from a set of colors known as a palette. If you're working with an 8-bit video card, you'll be able to work with palettes of 256 colors.

Director has nine standard palettes: two System palettes (for Macintosh and Windows systems), Rainbow, Grayscale, Pastels, Vivid, NTSC, Metallic, and VGA palettes. (The standard palettes aren't all available all the time; which palettes are available depend on the number of colors your monitor is set to display.) The System palette for your platform is the default palette. You can create as many additional palettes as you want.

There are three types of colors you can choose from the current palette: The foreground color is the color that you paint with when the pattern is solid and the ink is Normal. The background color is the secondary color in a pattern. The destination color is the color you want to replace the foreground color with when you use the Switch ink or the Switch Colors command; when you use the Cycle ink, the destination color is the end of the range of colors (beginning with the foreground color) that you want to cycle through; and when you use the Gradient ink, the destination color and the foreground color define the two extremes of the range that make up the color gradient.

Director has three standard palettes of patterns--Grays, Standard, and QuickDraw--as well as a custom palette. The custom palette contains a set of default patterns; you can create new patterns by editing the default patterns in the Patterns dialog box. You can also create a tile--a multicolored pattern that's a duplicate of a small rectangular section of an existing cast member.

Place button (paint window)

Lets you drag the selected bitmap cast member to the stage, score, or cast window. Press and hold down this button to drag the selected cast member.

Add button (paint window)

The Add button creates a new bitmap cast member. Director creates the new cast member in the first empty slot in the cast window that follows the location of the current cast member. If there are no empty slots following the current cast member, Director continues searching for an empty slot from the beginning of the cast. The new cast member inherits the stage monitor's color depth.

Tip: The keyboard shortcut for Add is Control-Shift-A.

The paint window tools

Click to see an illustration.

The table below shows whether clicking and/or dragging makes the tool work. Click the name of paint window tool for more information:

Tool (Click for info)	**Click or Drag**	**Result**
Lasso	Drag	Selects irregular shapes
Selection rectangle	Drag	Selects rectangular areas
Hand	Drag	Moves artwork within easel
Paint bucket	Click	Fills with foreground color or current pattern
Air brush	Both	Sprays foreground color or current pattern
Paintbrush	Both	Paints foreground color or current pattern
Text	Both	Selects text
Pencil	Both	Toggles pixels between foreground and background color
Rectangle	Drag	Draws hollow or filled rectangles and squares
Eraser	Both	Erases artwork
Ellipse	Drag	Draws hollow or filled ellipses and circles
Polygon	Click	Draws hollow or filled polygons
Line	Drag	Draws straight lines
Arc	Drag	Draws arcs (one quarter of an ellipse or circle)
Registration	Click	Sets registration point
Eyedropper	Click	Picks foreground color

Tips: If you press the Control key and click the image in the paint window, your view of the artwork will zoom in to a magnified view. In most cases, pressing the Shift key while dragging a tool constrains it to horizontal or vertical. The ellipse and rectangle tools are constrained to a perfect circle or square when Shift-dragging.

Drawing with the shape tools

The shape tools are the line, rectangle, circle, and polygon tools. The rectangle, circle, and polygon tools have a left, or hollow side, and a right, or shaded side. When you click the hollow side, the shape you draw is only an outline; it is not filled with a pattern or color. When you click the filled side, the shape you draw is filled with the currently selected foreground and background colors and patterns.

If you press the Alt key while drawing with one of the shape tools, the border of the shape is drawn with the current pattern.

If you press the Shift key while drawing with the line tool, the line is constrained to horizontal, vertical, or 45-degree angles.

Double-clicking the shaded side of the rectangle, circle, or polygon tool opens the Gradients dialog box.

Lasso paint tool

Related topic: Lasso pop-up menu

The lasso can be used to select an area. Once selected, drag the artwork, cut, copy, or clear it. You can also use the following commands from the Effects menu: Invert Colors, Trace Edges, Fill, Darken, Lighten, Smooth, and Switch Colors.

Once selected, drag the artwork, cut, copy, or clear it. You can also use the following commands from the Effects menu: Invert Colors, Trace Edges, Fill, Darken, Lighten, Smooth, and Switch Colors (the last four commands are available only on a color Macintosh).

Tips:

- When artwork is selected with the lasso, hold down the Alt key while dragging the artwork to make a copy of it.

- If you press the Alt key while dragging the lasso, the lasso draws straight lines to select a polygon shape. Click the lasso to anchor a point and draw another straight line. Double-click when you reach the end of your selection.

- Once an object is selected with the Lasso, pressing the Alt key while dragging the object makes a copy of your selection.

- Pressing the Shift key while dragging the object constrains its movement to a horizontal or vertical line. To move the cast member in one-pixel increments, select it on the stage and use the arrow keys on your keyboard.

Lasso pop-up menu (paint window)

Pressing and holding the mouse button while the pointer is on the lasso tool causes the lasso pop-up menu to appear.

Choosing a command from the lasso pop-up menu modifies how the lasso works.

- **Shrink** causes the lasso to tighten around the selected object so that only the object is selected.

- **No Shrink** permits you to select the entire area you drag around. The lasso selects whatever is inside the selected area.

- **See Thru** causes your selection to become transparent, as if the Transparent ink effect were applied.

Selection rectangle (paint window)

Related topic: Selection rectangle pop-up menu

The selection rectangle can be used to select artwork in the paint window. When selected with the selection rectangle, artwork can be dragged, cut, copied, and cleared. It can also be modified with the commands in the Effects menu.

To select the entire graphic in the paint window, double-click the selection rectangle tool.

To move the artwork in the drawing area once you've selected it, position the crosshair inside the selected area and then drag the selected area to a different location.

Several key combinations affect the selected area when you drag it.

Effect	Mouse or key combination
Copy	Alt-drag
Stretch	Control-drag
Stretch proportionally	Control-Shift-drag
Constrain to horizontal or vertical	Shift-drag
Clear	Backspace
Scale	Control-Alt-drag

Use the keyboard arrow keys to horizontally or vertically nudge an object selected with the lasso or the selection rectangle.

Selection rectangle pop-up menu (paint window)

If you press and hold the mouse button when the pointer is positioned on the selection rectangle tool, the selection rectangle pop-up menu appears with commands to modify the action of the tool.

- **Shrink** causes the rectangle to shrink around the selected artwork.

- **No Shrink** permits you to select everything within the selection rectangle.

- **Lasso** makes the selection rectangle tighten around your selection like the lasso tool. The selection tightens around the object and selectively selects the pixels according to the color of the pixel beneath the crosshair when you started your drag.

- **See Thru Lasso** makes the selection rectangle tighten around your selection like the lasso and applies the Transparent ink.

Hand tool (paint window)

The hand tool moves the image within the paint window, changing your position in the window relative to the artwork. Click the hand tool to select it, then drag the artwork to pan the view. If you press the Alt key while using the hand tool, its effect is the same as scrolling the view with the horizontal and vertical scroll bars.

Tip: A shortcut for using the hand is pressing the Spacebar. This turns any tool (except the text tool) into the hand tool when the mouse button is pressed.

Text tool (paint window)

The text tool lets you type in any font, size, or style in the paint window. Use it to set the font, size, and style of the text.

The text you create in the paint window is bitmapped. It can be dragged around the paint window before you deselect the text. However, once you click outside the text box after creating the text, you cannot edit its font, size, or style. To change the font, size, or style after clicking, you must erase the text and replace it with new bitmapped text with the attributes you prefer. However, you can modify selected text with ink effects from the Ink pop-up menu, patterns from the patterns pop-up palette, or foreground and background colors with the foreground and background color chips in the paint window.

Paint bucket (paint window)

The paint bucket fills any enclosed area with the currently selected color and pattern. The fill can be further modified with the ink effects in the Ink pop-up menu in the paint window. If there is a break in the outlined area you are filling, the paint will leak out and fill the surrounding area. If this happens, immediately choose Undo Bitmap from the Edit menu. Then Zoom In from the Paint menu to get a magnified view and inspect the outline for breaks.

Tip: Double-clicking the paint bucket tool opens the Gradients dialog box.

Air brush (paint window)

Related topic: Air brush pop-up menu

The air brush sprays the currently selected color and pattern. The spray can be further modified by choosing the ink effects from the Ink pop-up menu in the paint window. The longer you hold the airbrush in one spot, the darker it fills in the area.

Tip: Double-clicking the air brush in the tool palette opens the Air Brushes dialog box. Use this dialog box to set the size of the air brush's spray, the size of the dots of paint it sprays, and how fast it sprays paint.

Air brush pop-up menu (paint window)

If you press and hold the mouse button when the pointer is positioned on the air brush, the air brush pop-up menu appears. Each of the five settings in the pop-up menu can be defined so you can have several types of spray available without opening the Air Brushes dialog box.

To define a setting:

1. Choose the menu item you want to define from the Air Brush pop-up menu.

2. Choose Air Brushes from the Paint menu.

3. Select the type of spray you want in the Air Brushes dialog box.

4. Click Set.

The choices you make in the Air Brushes dialog box are assigned to the menu item and remain until you change them.

Paintbrush (paint window)

Related topic: Paintbrush pop-up menu

The paintbrush draws with the currently selected colors, ink effect, or fill pattern. Double-click the paintbrush to change the size and shape of the brush. When the Brush Shapes dialog box appears, click the brush shape you need, and then click Set.

Paintbrush pop-up menu (paint window)

The paintbrush pop-up menu is similar to the air brush pop-up menu. Press and hold the mouse button while the pointer is positioned on the paintbrush tool to open the pop-up menu.

Each of the five settings in the pop-up menu can be defined so you can have several brush shapes available without opening the Brush Shapes dialog box.

To define a setting:

1. Choose the menu item you want to define from the Paintbrush pop-up menu.
2. Choose Brush Shapes from the Paint menu.
3. Select the brush shape you want in the Brush Shapes dialog box.
4. Click Set.

The choices you make in the Brush Shapes dialog box are assigned to the menu item in the pop-up menu and remain until you change them.

Pencil (paint window)

The pencil creates a one-pixel-wide line. On a black and white monitor, the pencil draws black pixels on a white background and white pixels on a black background. On a color monitor, the pencil draws with the currently selected foreground color unless you are drawing on pixels that are the foreground color. In that case, the pencil draws in the background color.

Double-clicking the pencil tool magnifies your view of the artwork in the paint window. It magnifies the current easel at the point last clicked with any of the paint tools. You can also zoom in while using the pencil or any other tool by pressing the Control key and clicking.

Once you are in a magnified view you can edit the cast member pixel by pixel. You can use any of the paint tools while in a magnified view. Clicking the reduced view in the upper right corner of the paint window returns you to a 100% view. Double-clicking the pencil in the tool palette while in magnified view also returns you to a 100% view.

Move around the magnified view using the paint window's scroll bars or by using the hand tool.

Rectangle tool (paint window)

The rectangle tool draws rectangles of any shape and size. When you click the left or hollow side of the rectangle tool, it draws an outline in the current foreground color as you drag the crosshair diagonally. When you click the right or shaded side of the rectangle tool, the rectangle you draw is filled with the current foreground and background colors in the current ink and pattern. The thickness of the rectangle's border is controlled with the line width selector at the bottom of the tool palette.

Tip: Double-clicking the shaded side of the rectangle tool opens the Gradients dialog box.

To constrain the rectangle to a square, press the Shift key as you drag with the crosshair pointer. If you press the Alt key while drawing a rectangle, the border is drawn with the current pattern.

Eraser (paint window)

The eraser clears the portion of the cast member you drag across. The eraser always clears to white. Double-clicking the eraser tool erases everything in the paint window's visible area.

Ellipse tool (paint window)

The ellipse tool creates circles and ovals. Like the rectangle tool, the ellipse is an outline if you click the tool on the left or hollow side, and is filled with the current foreground and background colors, ink, and pattern when you click the tool on the right or shaded side. The thickness of the line is controlled by clicking the line width selector at the bottom of the tool palette.

Tip: Double-clicking the shaded side of the circle tool opens the Gradients dialog box.

When you hold down the Shift key as you draw, the circle tool draws perfect circles. If you use the circle tool while pressing the Alt key, the border of the circle is drawn with the currently selected pattern.

Polygon tool (paint window)

The polygon tool draws polygons with as many sides as you want. As with the rest of the shape tools, the left or hollow side of the tool creates an outline, and the right or shaded side of the tool draws an area filled with the current foreground and background colors, ink, and pattern. When you click the tool, the pointer becomes a crosshair. Click in the paint window to start drawing the side of the polygon. Each time you click, a line is drawn from the spot you clicked previously. When you are ready to finish off your shape, double-click and a line is drawn connecting the spot you just clicked and the point of your first click.

The thickness of the lines drawn with the polygon tool is controlled with the line width selector at the bottom of the tool palette. If you press the Alt key while drawing a polygon, the border is drawn with the currently selected pattern.

Tip: Double-clicking the shaded side of the polygon tool opens the Gradients dialog box.

Line tool (paint window)

The line tool draws straight lines at any angle. When you hold down the Shift key, the line tool draws vertical, horizontal, or 45-degree lines, depending upon the direction you begin to drag. Change the line width by clicking the line width selector in the tool palette.

The line is drawn with the currently selected foreground color and ink effect.

Tip: Pressing the Alt key while drawing a line causes the line to be drawn in the currently selected pattern.

Arc tool (paint window)

The arc tool draws one quarter of an ellipse or circle. When the tool is active, the pointer becomes a crosshair. Drag the crosshair from the starting point of the line and move the pointer to see the curve. Experiment with dragging the tool until it produces the line you need. The thickness of the arc is controlled with the line width selector at the bottom of the tool palette.

The line is drawn with the currently selected foreground color and ink unless you press the Alt key while dragging, in which case the arc is drawn with the current pattern.

Registration tool (paint window)

When you create a cast member in the paint window, it is automatically assigned a registration point that centers on the artwork. You can see this by creating a simple cast member and clicking the registration tool. When you click the tool, dotted lines appear in the paint window. The intersection of these dotted lines is the **registration point** of the cast member. Using registration points speeds your ability to quickly and accurately put cast members on the stage and have them all line up with the same point.

The registration tool is used to line up cast members for frame-by-frame animation. When you have a series of cast members, you can align their registration points so you have a fixed reference point for animation. A simple example might be the hands of a clock. The hands are made up of different cast members at various positions around the face of the clock, but they must be anchored to the center of the clock. Setting a registration point on the hands at the point about which they rotate enables them to line up in the proper position on the stage.

Another example of using registration points is a running figure. You can set a registration point at the same spot on the ground so when the series of running figures is animated, they bounce up and down relative to the same point on the ground.

Registration points are most useful when animating with the Switch Cast Members command in the score menu, with film loops, and when previewing a sequence of cast members in the paint window with the Align Bitmaps command in the Cast menu. When you use the Align Bitmaps command, the cast members in the paint window are all lined up on each other's registration points, permitting you to preview the animation by quickly flipping through the cast members using the left and right arrows at the top of the paint window.

The default registration point for a bitmap image is the center of the cast member in the paint window. However, the registration point for shapes, buttons, or text is always the upper left corner of the image. Clicking a point in the paint window sets the registration point at that location.

To set a registration point:

1. In the paint window, select the cast member that you want to register.

2. Click the registration tool.

   The dotted lines in the paint window intersect at the registration point. The default registration point is the center of the cast member.

   The pointer changes to a crosshair when you move it to the paint window.

3. Click a location in the paint window to set the registration point.

   You can also drag the dotted lines around the window to reposition the registration point.

Tip: To reset the default registration point at the center of the cast member, double-click the registration tool.

Eyedropper (paint window)

The eyedropper is used to match colors. When you select the eyedropper, any color you click in the paint window becomes the foreground color. Use it to match colors without opening the color palette. Clicking a tool with the right mouse button turns the tool into the eyedropper.

The Ink pop-up menu (paint window)

Below the tool palette is a pop-up menu you can use in the paint window. Ink effects extend the results you can achieve with the paint tools. They modify either how a tool behaves, how colors are displayed, or--in the case of the Reveal ink--how two cast members interact graphically. Some inks work better when painting with patterns, and others work better when painting with solid colors. Some inks are available only with certain tools, and some only when the color depth of both the monitor and the cast member is set to 8 bits (256 colors).

Ink (click for info)	**Works best with** (Solids, patterns, tools)
Normal	Solids or Patterns. Paintbrush, paint bucket, air brush, pencil, shape tools, text tool
Transparent	Patterns. Paintbrush, air brush, shape tools, text tool
Reverse	Solids or patterns. Paintbrush, air brush, pencil, shape tools, text tool
Ghost	B&W solids or color patterns. Paintbrush, air brush, shape tools, text tool
Gradient	Solids. Paintbrush, paint bucket, shape tools
Reveal	Paintbrush, paint bucket, air brush, shape tools (except line and arc), text tool
Cycle	Paintbrush, air brush in 256 colors only
Switch	Solids. Paintbrush in 256 colors only
Blend	Solids or patterns. Paintbrush, air brush, shape tools, text tool in 256 colors only
Darkest	Patterns. Paintbrush, air brush, shape tools, text tool in 256 colors only
Lightest	Patterns. Paintbrush, air brush, shape tools, text tool in 256 colors only
Darken	Paintbrush in 256 colors only
Lighten	Paintbrush in 256 colors only
Smooth	Paintbrush in 256 colors only
Smear	Paintbrush in 256 colors only
Smudge	Paintbrush in 256 colors only
Spread	Paintbrush
Clipboard	Paintbrush

Normal (Ink pop-up menu of the paint window)

Normal is the default ink. It is opaque and maintains the color of the current foreground color and pattern.

Transparent (Ink pop-up menu of the paint window)

Transparent ink makes the background color of patterns transparent so you can see artwork drawn previously in the current cast member through the pattern.

Reverse (Ink pop-up menu of the paint window)

Reverse ink makes overlapping colors reverse. Any pixel in the foreground art that was originally white becomes transparent. Any pixel that was black reverses the color of the background art.

Ghost (Ink pop-up menu of the paint window)

Ghost in black and white creates an image than can only be seen when placed over a black background. In color, Ghost draws with the current background color.

Gradient (Ink pop-up menu of the paint window)

Gradient lets you paint with the gradient fill selected in the Gradients dialog box. A gradient fill is one that progresses from one color, the foreground, to another color called the destination color. You can paint with Gradient ink with the paintbrush, paint bucket, or shape tools.

Reveal (Ink pop-up menu of the paint window)

Reveal works indirectly with the art in the previous cast slot. Imagine the previous cast member's artwork covered with a white area. Reveal erases the white area to show the artwork in the previous easel. Reveal can be used to create specific shapes from shades created with the air brush. Since it is impossible to mask certain shapes for the air brush, spray an area with the air brush first; then in the next cast member, paint the shapes you need with a Reveal ink. As you paint your object, you will expose the air brush pattern in the previous easel.

Cycle (Ink pop-up menu of the paint window)

Cycle is a color ink. As you draw with a cycling ink, the colors change as the ink progresses through the palette. The beginning and ending points of the color cycle are determined by the foreground and destination colors. If you want to cycle through the whole palette, choose white as the foreground color and black as the destination color.

Switch (Ink pop-up menu of the paint window)

Switch changes any pixel that is the current foreground color to the current gradient destination color as you paint over that color.

Blend (Ink pop-up menu of the paint window)

Blend creates a translucent color ink. You can see the background object, but its color is blended with the foreground object's color. You can choose the percentage of blend in the Paint Window Options dialog box.

Darkest (Ink pop-up menu of the paint window)

Darkest is a useful ink for colorizing black and white artwork. For example, if you are painting yellow over black and white, black will remain black since it is darker than yellow, and white will become yellow because yellow is darker than white.

Lightest (Ink pop-up menu of the paint window)

Lightest is another useful ink for colorizing black and white artwork. For example, if you are painting yellow over black and white, black objects become yellow when painted with the Lightest ink effect, and white remains white because it is lighter than yellow.

Darken (Ink pop-up menu of the paint window)

Darken makes colors darker. The more the paintbrush passes over an area, the darker it becomes. The color of the foreground, background, or destination inks have no effect on Darken. Darken creates an effect that is the same as reducing a color's brightness with the controls in the color palettes window. You can vary the rate of this ink effect in the Paint Window Options dialog box.

Lighten (Ink pop-up menu of the paint window)

Lighten makes existing artwork lighter. The more times you pass over the artwork with the paintbrush, the lighter it becomes. The color of the foreground, background, or destination inks have no effect on Lighten. Lighten creates an effect that is the same as increasing a color's brightness with the controls in the color palettes window. You can vary the lightness of this ink effect in the Paint Window Options dialog box.

Smooth (Ink pop-up menu of the paint window)

Smooth that blurs existing artwork when painted with the paintbrush. It is not directional as are Smear and Smudge. The color of the foreground, background, or destination inks have no effect on Smooth. Smooth only works with art already in the paint window. Use it to smooth out jagged edges.

Smear (Ink pop-up menu of the paint window)

Smear works with the paintbrush. It is similar to mixing paint. Any area you drag across with a Smear ink is spread in the direction of the brush and fades as it gets farther from its source. The color of the foreground, background, or destination inks have no effect on Smear. Smear only works with art already in the paint window.

Smudge (Ink pop-up menu of the paint window)

Smudge is a color ink for the paintbrush that is similar to Smear. It is also like mixing paint. The colors fade faster as they are spread. The color of the foreground, background, or destination inks have no effect on Smudge. Smudge only works with art already in the paint window.

Spread (Ink pop-up menu of the paint window)

Spread works with the paintbrush in color. Whatever is under the paintbrush when you start to drag is picked up as the ink for the brush. Copies of what is beneath the brush are pushed across the window as you draw.

Clipboard (Ink pop-up menu of the paint window)

Clipboard uses the current contents of the Clipboard as a pattern to paint with.

Destination color chip (paint window)

The destination color chip lets you select the second color that you want to use--along with the foreground color--with the Gradient, Cycle, and Switch inks.

With the Gradient ink, the foreground color and the destination color mark the beginning and end of a range of colors that define a gradient. A gradient is a blend of colors that can be used for shading, highlights, backgrounds, and special effects. (When you're working in black and white, gradients are created with a pattern of black and white pixels that fade from black to white or vice versa.)

With the Cycle ink, the foreground color and the destination color define the beginning and end of the series of colors you cycle through as you paint.

With the Switch ink, the destination color is the color you want to use to replace the current foreground color.

To set a destination color, click the destination color chip to display the pop-up palette and then choose the color you want to use.

Tip: Hold down the Alt key while pressing the up or down arrow key to cycle through the colors in the destination color chip.

Foreground color chip (paint window)

The foreground color is the color displayed in the foreground color chip. It's also displayed in the color chip on the left side of the gradient color selector. The foreground color is the color you work with when you're using the solid pattern solid and the Normal ink effect.

Press the up or down arrow key to cycle through the colors in the color palette associated with the Foreground color chip.

Click the foreground, background, or destination color chip to display the palette, and then double-click the chip again to open the color palettes window.

Background color chip (paint window)

The background color is the color displayed in the lower right color chip. The background color is the secondary color that appears in a pattern. When used with the Transparent ink, the background color in a pattern is drawn so you can see through the background color to artwork beneath. The background becomes transparent.

Tip: Hold down the Shift key while pressing the up or down arrow keys to cycle through the colors in the color palette associated with the Background color chip.

Pattern chip (paint window)

The current pattern is displayed in the pattern chip below the two color chips in the tool palette.

Click the pattern chip to select a new pattern from the pop-up palette.

Tip: Pressing the Alt key before displaying the pattern palette permanently changes the patterns to shades ranging from the foreground color to the background color rather than the set of patterns you see without pressing the Alt key. Press the Alt key again to return to the default set of patterns.

Click the pattern chip to select a new pattern from the pop-up palette.

Line width selector (paint window)

The line width selector controls the thickness of the line drawn by the line or arc tool and the thickness of the borders drawn by the shape tools.

The width of the line drawn by the line, arc, rectangle, ellipse, and polygon tools can be changed with the line width palette. The line width palette has several settings for line width ranging from no line (the dotted line in the line selector) to Other. Use the dotted line setting when you want to draw filled shapes without borders. If you choose Other, the line width is determined by the Other Line Width setting in the Paint Window Options dialog box.

Tip: Double-click the line width selector to open the Paint Window Options dialog box. Use it to set the Other Line Width.

Color resolution indicator (paint window)

The color resolution indicator displays the color resolution of the current cast member in the paint window.

Double-clicking the color resolution indicator opens the Transform Bitmap dialog box. Use the Transform Bitmap dialog box to change the color resolution of the current cast member in the paint window. Changing the color resolution from color to black and white saves disk space. You can still make a selected 1-bit cast member a color other than black by selecting colors with the foreground and background color chips in the tools window after it has been reduced to 1-bit. (This colorizes the sprite on the stage, but does not affect the original cast member, which remains black and white.)

If you import black and white cast members, changing their color resolution to multiple colors permits you to colorize them with any color in the current palette.

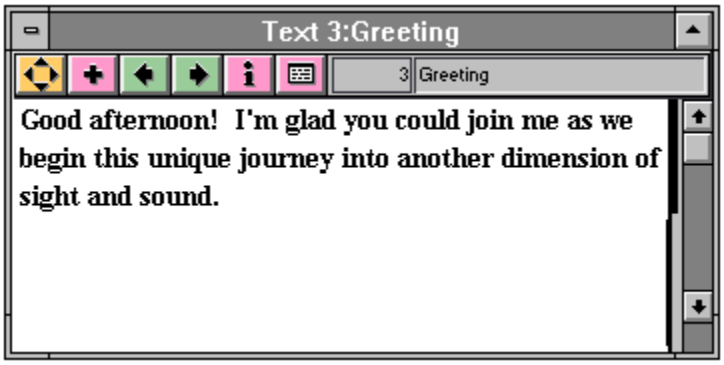
Text window

Use the text window to create and edit a text cast member.

Click to see an illustration.

Click a topic for more information:

- Text window basics
- Add button
- Selecting and editing text
- Using the resize handle
- Formatting text
- Applying color to text


Text 3:Greeting
3
Greeting
Good afternoon! I'm glad you could join me as we begin this unique journey into another dimension of sight and sound.

Text window basics

The commands on the Text menu set text attributes and find and replace text. You can also use the tools window to set the foreground and background colors of the text you create.

Text created in a text window prints well with a laser printer. When printing frames from your movie you can avoid the jagged lines of bitmapped text by using the text window rather than the paint window to create text.

To open a new text window, choose Text from the Window menu. Director adds the name of the text window to the bottom of the Window menu. The window's title displays the position of the first empty cast member where the text will be entered in the cast window.

Alternatively, double-clicking a text cast member in the cast window or on the stage will either open the cast member's text window or bring it to the front if it is already open.

Changes you make in the text window are automatically reflected in the cast when you stop typing, switch to a different window, click outside the text window, or close it. All other windows displaying the current text cast member are automatically updated.

The Place, Previous, Next, Info, and Script buttons work the same as they do in the cast window.

Add button (text window)

Click Add to create a new text cast member. (If there's a text window already open, the window with the new cast member replaces the existing one; if you want to leave the existing text window open when you create the new cast member, hold down Alt as you click the Add button.) Director creates the new text cast member in the first empty slot in the cast window that follows the location of the current cast member.

Tip: The keyboard shortcut for Add is Control-Shift-A. To create a new empty cast member and in a new window, use Control-Alt-Shift-A.

Double-click the cast number (to the right of the Script button) to open the cast window and select the current script's thumbnail.

Double-click a text cast member in the cast window or on the stage either to open the cast member's text window or to bring it to the front if it's already open.

Selecting and editing text

Click in the text window to set an insertion point. The text you type is inserted into the current cast member and automatically appears in the cast window. You can enter up to 32,000 characters, including spaces.

You select text in the text window or on the stage by dragging across the text, or by double-clicking to select a whole word. Triple-clicking selects all text in the cast member, which is the same as choosing Select All from the Edit menu.

You can cut, copy, and paste text between text windows.

If there's more text in the cast member than can be displayed in the window at one time, use the Home, End, Pg Up, and Pg Dn keys to display the part of the text you want to see.

Using the resize handle (text window)

When text is on the stage, or in the text window, a resize handle at the right of the text controls the width of the text and the shape of any border set with the Text menu. Drag the resize handle in the text window or on the stage to change the text's width.

The handle's thickness in the text window indicates the current vertical extent of the text. The handle appears thicker in the area occupied by the text, and narrower where there is no text.

Tip: To create another view of the same text window, hold down the Alt key while choosing the name of the active text window from the Window menu. A second view of the text window is useful if you're editing a large text cast member, since you can display different sections of the text in each view, and cut and paste between them.

Formatting text

To apply a format to text, select it and choose a command from the Text menu. You can apply a font, size, and style to text on a per-character basis.

Applying color to text

To apply a color to text:

1. Select the text in the text window.
2. Choose the Tools command from the Window menu.

   The tools window appears.
3. In the tools window, choose a text color using the foreground color chip.

   The color you choose is applied to selected text in the text window.
4. Choose a color for the text window's background using the background color chip.

   The color you choose is applied to the text window background.

Tools window

Click to see an illustration.

Click a tool name for more information:

- Text tool
- Line tool
- Shape tools
- Button tools
- Foreground, Background color chips
- Pattern chip
- Line width selector

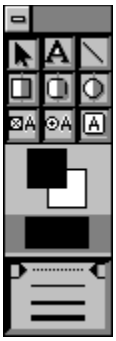
Click a topic for more information:

- Creating shapes
- Making changes to a shape

The tools window has tools for creating text objects and draw objects directly on the stage. Text objects and draw objects make excellent backgrounds and print well with a laser printer. They are easy to create and are resizable. They also use less memory than bitmapped images you create in the paint window. If you are producing handouts for a presentation, use text created with the text tool to avoid the jagged lines of bitmapped text.

The shapes you create with the shape tools appear as cast members in the cast window and the score window. The text you create with the text tool appears in the cast, score, and text windows. Text objects and draw objects do not appear in the paint window.

Using a text object for animation does have a drawback. It tends to animate more slowly than bitmapped text, so it's best to use bitmapped text if animation speed is an issue.



Text tool

The text tool is an alternative to the text window for creating text. When you create text with the text tool on the stage it can also be seen in the text and cast windows. Click the arrow to select text that is already on the stage. You can change the color of a selected text object using the foreground and background color chips in the tools window.

Line tool

Click the QuickDraw line tool to select it and drag it across the stage to draw. You can choose the color for the line with the foreground and background palettes that appear when you press on the color chips. The width of the line tool is controlled with the line width selector at the bottom of the tools window.

Shape tools

Click the shape tools on the left or hollow side to draw an outline of the shape. Click the right or shaded side of the tools to draw with a solid color or pattern. You can choose the color for the shape with the foreground and background palettes that appear when you click the color chips. Use the pattern chip to select the current pattern. The thickness of the borders of the shape tools is controlled with the line width selector at the bottom of the tools window. You can change the shape using the Shape drop-down list in the Cast Member Info dialog box.

Button tools

Director provides three tools for creating buttons, checkboxes, and radio buttons. Click the checkbox tool, button tool, or radio button tool and drag a rectangle on the stage to create the button. Then type the text that you want to appear on or next to the button. If necessary, set the font, style, and size. The button is placed in the cast as a button cast member. You can edit the button's text on the stage or in a text window.

Foreground, Background color chips

You can use the foreground and background color chips in the tools window to:

- Set the color of text you type in the script window
- Set the color of text you type in the text window or on the stage
- Set the color of draw objects
- Set the color of 1-bit sprites
- Identify the foreground and background colors of sprites you want to apply a score ink to.

To set the color of text you type in the text or script windows, make the text or script window active, and then choose a text color using the foreground color chip. The color is subsequently applied to text you type, but not to existing text. To set the window's background color, choose a color using the background color chip.

To set the color for a sprite, select the sprite in the score or on the stage and choose a new foreground color using the foreground color chip, or a new background color using the background color chip.

If you change the color of a 1-bit cast member, Director changes the color of the sprite on the stage but does not change the color of the actual cast member, which remains black and white.

Pattern chip

Lets you select the current pattern for a shape tool.

Line width selector

Lets you select the current width of the line tool, or the border for a shape tool.

Creating shapes

You create shapes with the line, rectangle, rounded rectangle, and circle tools. As with the paint tools, clicking the left side of the tool produces a hollow shape. Clicking the right side of the tool produces a shape filled with the current color or pattern in the pattern chip.

Tips:

- Pressing the Shift key while dragging the crosshair constrains the tool to drawing a perfect square or circle. The line tool is constrained to horizontal, vertical, or 45-degree lines with the Shift key.

- To create a shape that is the full size of the screen, draw one as large as you can, drag it up and to the left, then extend it by dragging the handle in the lower right corner. You can then reposition it slightly lower and to the right to hide the edge along the right and the bottom. You can also use the Sprite Info command on the Score menu to enter the exact dimensions and position of the shape, to make it the same size as the screen.

Making changes to a shape

To make changes to a shape:

1. Click the pointer in the tools window.

   The pointer allows you select shapes.

2. Click a shape on the stage to select it.

3. In the tools window, select a new foreground color, background color, pattern, or line thickness.

   The shape changes to match your selection.

Color palettes window

Related Topics:

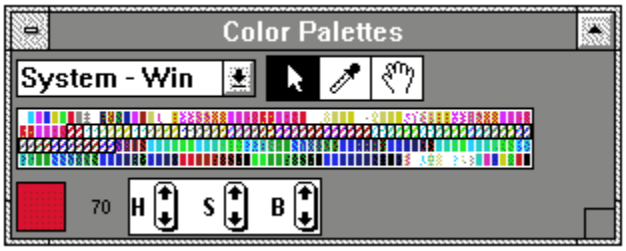
More about color palettes
Editing colors

Click to see an illustration.

The color palettes window lets you determine which palette to use to color the cast members on the stage. When you open the color palettes window, the Palette menu appears in the menu bar and the color palettes window displays the current palette (either 16 or 256 colors). You can edit colors or switch palettes entirely. To use a palette in the score, select a cell in the palette channel and choose Set Palette from the Score menu.

Director has nine built-in palettes:

- System -- Mac (the standard 256-color Macintosh system palette)
- System -- Win (the standard 256-color Windows system palette)
- Rainbow palette
- Grayscale palette
- Pastels palette
- Vivid palette
- NTSC palette
- Metallic palette
- VGA palette (Video Graphics Array), a special palette for VGA 4-bit displays. It provides consistent results when playing Director movies under Windows in 4-bit mode.


Color Palettes
System - Win
70
H
S
B

More about color palettes

If you add new palettes to your movie from other graphics applications, those palettes also appear in the pop-up menu and in the cast window. Duplicated palettes that you modify also appear in the color palettes window's pop-up menu.

When you switch to a new palette, the pixels change color in sprites on the stage based on the position number of their colors in the original palette.

Sprites always take on the colors of the palette that is currently active. The active palette is determined by which cast member is selected in the cast, score, or paint windows or which palette is selected in the color palettes window.

Use the hand tool to drag colors in the palette to reposition them.

Use the eyedropper to match the color of any pixel you click with the tool.

Clicking the arrow changes your pointer back to the arrow pointer.

Editing colors

The easiest way to edit colors is with the Windows Color dialog box. To edit a color using the Color dialog box, double-click a color in the color palettes window. The Color dialog box uses both the HSL (Hue, Saturation, Luminosity) system and the RGB (Red, Green, Blue) system to measure color.

The color palettes window uses a variant of the HSL system called HSB (Hue, Saturation, Brightness). Brightness and Luminosity are identical. To edit selected colors in the color palettes window using the HSB system, click the arrows at the bottom of the window to increase or decrease the value of hue, saturation, or brightness. The index number of the selected color appears in the lower left corner of the color palettes window along with a sample of the color.

Hue is the primary or secondary color created by mixing two primaries.

Saturation is a measure of how much white is mixed in with the color. A fully saturated color is vivid; a less saturated color is a washed out pastel or even a shade of gray.

Brightness or luminosity controls how much black is mixed in with a color. Colors that are very bright have little or no black. As the brightness/luminosity is reduced, the color gets darker as if more black were added. If brightness/luminosity is reduced to 0, then no matter what the values for Hue or Saturation, the color will be black. If brightness/luminosity is increased to its maximum value, then no matter what the values for Hue or Saturation, the color will be white.

Digital video window

Double-clicking a digital video cast member in the cast window or on the stage displays the cast member's digital video window.

The digital video window lets you play the digital video movie. Use the controls at the bottom of the window to play, stop, advance, or rewind the movie. When the movie is stopped, you can cut, copy, and paste frames from the movie into another digital video window.

Click to see an illustration.

Note: On a Macintosh, the term "digital video" refers only to QuickTime animations. However, on a PC, Director for Windows supports two digital video formats: Microsoft's Video for Windows (.AVI) and QuickTime for Windows (.MOV).

You can use the digital video window to play and edit a digital video movie, to place the movie on the stage, to display the movie's Cast Info dialog box, and to attach a script to the movie.If the window contains a Video for Windows (.AVI) movie, the movie starts playing as soon as you open the window (or whenever you switch to the window if it's already on the screen). To step the movie, click it.

If the window contains a QuickTime for Windows (.MOV) movie, use the controls at the bottom of the window to play, stop, advance, or rewind the movie.

The **Place**, **Previous**, **Next**, **Cast Member Info**, and **Open Script** buttons work the same as they do in the cast window.

Tip: The keyboard equivalent for Add is Control-Shift-A. To switch to a new empty cast member and open a new window for it, use Alt-Control-Shift-A.


Video 5:GLOBEREV.AVI
5
GLOBEREV.AVI

Script window

You use the script window to enter and edit Lingo script. A script window can contain up to 32,000 characters.

Click to see an illustration.

For a description of the buttons at the top of the script window, see the Cast window topic.

Tip: Many Director Help topics for Lingo commands include examples that you can paste into your scripts and use.

More than one script window can be open at the same time. The Window menu lists all open script windows. The active script window appears underlined.

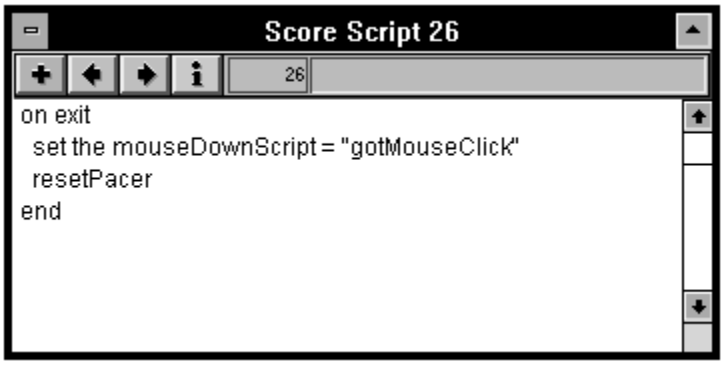
Director saves the changes you make in the script window when you stop typing, switch to a different window, click outside the script window, or close it.

Director recompiles the script in the script window when you:

- Close it (by double-clicking the System box, or by pressing Enter on the keypad or Control-W)
- Choose Recompile Script or Recompile All Scripts from the Text menu
- Play the movie.

See "Writing scripts" in Chapter 4 for information about creating scripts. For complete information about using scripts, see Using Lingo.

Tip: Double-click the cast number (to the right of the Info button) to open the cast window and select the current script's thumbnail. Double-click a cell in the script channel to open a script window; the help window includes Lingo examples that you can paste into your scripts and use.


Score Script 26
26
on exit
set the mouseDownScript = "gotMouseClick"
resetPacer
end

Message window

Click to see an illustration.

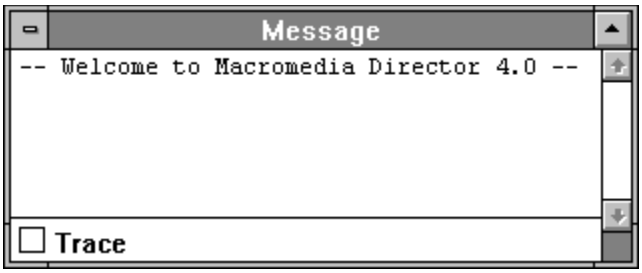
The message window is a convenient place to experiment with and test Lingo scripts. Actions occur immediately when you press the Enter key, so you can see the results before you insert your scripts into a movie. This allows you to see the results of any script, including whether it is a valid script.

To move around the message window, use the arrow keys or scroll the window. Press Control-up arrow to move the insertion point to the top of the window. Press Control-down arrow to move the insertion point to the bottom of the window.

Use Cut or Clear to remove text from the message window. To clear all the text from the insertion point to the bottom of the window, press Control-Shift-Delete.

The message window has a Trace feature that can help you find problems in scripts. If Trace is checked, you can play a movie and all the Lingo commands will appear in the message window as they are executed. Using the Trace feature slows down animation, so turn it off when you're not using it.

See the *Using Lingo* book for more information about working with the message window.


Message
-- Welcome to Macromedia Director 4.0 --
Trace

Tweak window

Click to see an illustration.

The tweak window allows you to move one or more selected sprites in any direction with precision. Drag the point in the left side of the window to set the number of pixels. The value in the boxes will continue to increase or decrease as you drag the line beyond the boundary of tweak window.

After you set the number of pixels, click the Tweak button to move the selected sprites. Continue clicking the Tweak button to repeatedly move the selected sprites the same distance.

Tip: You can use the arrow keys on the keyboard to move the tip of the line in the tweak window one pixel at a time. When the tweak window is closed, you can also use the arrow keys to move selected sprites one pixel at a time.


ΔX
16
ΔY
-22
Tweak

Markers window

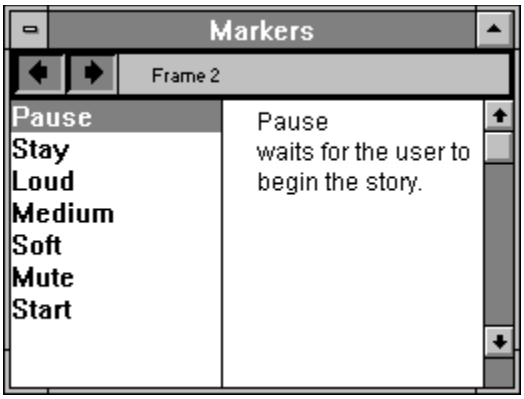
Click to see an illustration.

The markers window lets you write comments associated with markers you set in the score. For example, a Director animation can have staging or acting directions, storyboard scripts, or speaker's notes written in the markers window and tied to specific frames in the score. A storyboard, transparencies, or handouts can be printed that include pictures of selected frames of your movie along with the comments written in the markers window. Double-clicking one of the triangular markers in the score window opens the markers window to the comment associated with that frame.

Once you've labeled a frame in the score, you can use the marker name in your scripts. This is important because references to frame numbers may become invalid if you insert or delete frames in the score. Marker names remain constant no matter how much you edit the score.

The left column of the markers window displays the marker names from the score window. Clicking a marker name or clicking the left or right arrow in the markers window moves the playback head to the selected marker location in the score window, and displays the comments associated with the marker in the markers window.

To enter a comment, click a marker name to select it. You can then enter your comments beginning at the insertion point that appears in the right column of the markers window. By default, the marker name appears as the first line of text in the right column. If you edit the marker name, your changes are also reflected in the score window. If you don't want to edit the marker name, press the Return key to start a new line, and enter your comments on the new line in the right column of the markers window.


Markers
Frame 2
Pause
Stay
Loud
Medium
Soft
Mute
Start
Pause
waits for the user to
begin the story.

Duplicate Window command

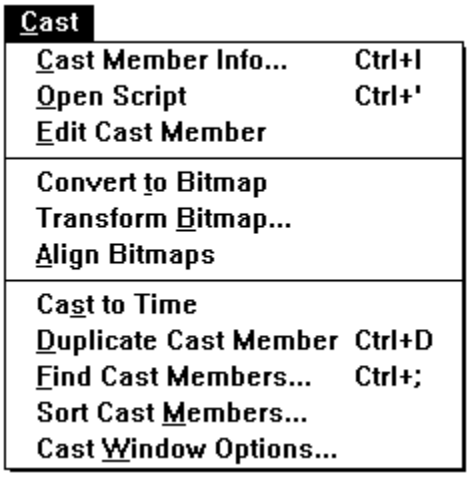
This command duplicates the front-most window and its contents, creating another view of it. Director positions the duplicate on top of the original window, so you must move the duplicate to uncover the original window. This command only works if a text, digital video, or script window is the front-most window.

Tip: Pressing Alt while choosing a text, digital video, or script window from the Window menu is a keyboard shortcut for choosing this command.

Duplicating a window is useful if the window's contents are large and you want to look at or edit different sections of the window simultaneously. Changes you make in the window are automatically reflected in all other views of the same window.

Cast menu

Click a menu command for more information:



The Cast menu contains commands for managing cast members.

Cast Member Info

Cast Member Info displays a dialog box containing information about the selected cast member: its name, cast position, its type, and its size in kilobytes. The Cast Member Info dialog box also displays additional information and options for each cast member type. Use the options in the dialog box to define the behavior and appearance of the selected cast members.

The type of information in the Cast Member Info dialog box depends on the type of cast member. Click for more information on a type of Cast Member Info dialog box:

Bitmap Cast Member Info
Button Cast Member Info
Digital Video Cast Member Info
Film Loop Cast Member Info
Linked Director Movie Cast Member Info
Palette Cast Member Info
PICT Cast Member Info
Script Cast Member Info
Shape Cast Member Info
Sound Cast Member Info
Text Cast Member Info

Getting information for multiple cast members

In the cast window, select a cast member and click the Cast Member Info button as a shortcut for choosing this command.

If you are editing the cast member in the paint, text, digital video, or script window, you can click the window's Cast Member Info button as a shortcut for choosing this command.

Tip: Press Control-I or click a cast member in the cast window or on the stage with the right mouse button to open the Cast Member Info dialog box.

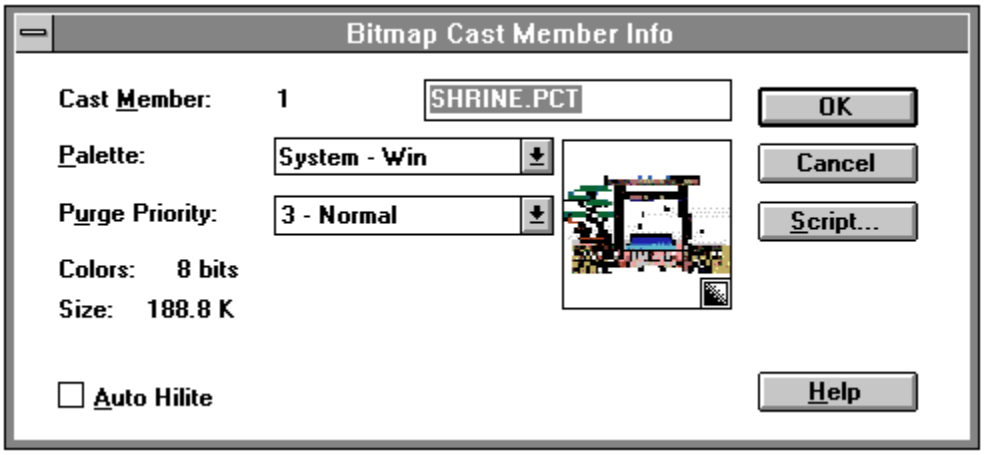
Using the Cast Member Info dialog box is particularly useful when switching back and forth between creating new color cast members and editing old color cast members. You can reset the palettes of each cast member manually to avoid the confusion of working with multiple palettes.

Bitmap Cast Member Info

Click here to see an illustration.

Click for more information:

- Cast Member field
- Palette
- Purge Priority
- Colors
- Size
- File Name
- Auto Hilite
- Script


Bitmap Cast Member Info
Cast Member:
1
SHRINE.PCT
OK
Palette:
System - Win
Cancel
Purge Priority:
3 - Normal
Script...
Colors:
8 bits
Size:
188.8 K
Auto Hilite
Help

Cast Member field (Bitmap Cast Member Info)

Enter or edit the name in the field. The name remains attached to the cast member if it is cut or copied and pasted. Director does not prevent you from creating duplicate cast member names, but you should avoid using them. If more than one cast member with the same name has a script attached to it, Lingo uses the script attached to the cast member with the lowest cast number. Use cast names instead of cast numbers to address cast members in a Lingo script, so that you don't have to update your scripts if your cast members get renumbered or sorted.

Palette (Bitmap Cast Member Info)

Lets you temporarily assign a different palette to the cast member, while maintaining the cast member's original palette. You can change the palette assignment at any time by choosing another palette from the pop-up menu.

Purge Priority (Bitmap Cast Member Info)

Controls how Director removes the bitmap cast member from memory if memory is low.

3-Normal: The selected cast member will be removed from memory as necessary. This is the default

2-Next: The selected cast member will be among the next to be removed from memory.

1-Last: The selected cast member will be among the last to be removed from memory.

0-Never: The selected cast member remains in memory and is never purged.

Note: The more cast members you give a purge priority of Last or Never, the more likely a movie is to run out of memory. Be especially cautious using Never.

Colors (Bitmap Cast Member Info)

Displays the color depth of the cast member.

Size (Bitmap Cast Member Info)

Displays the size of the cast member, in kilobytes.

File Name (Bitmap Cast Member Info)

If a cast member is an external .BMP or .DIB file that is linked to its source file on disk displays the location of the linked file. Clicking the file name lets you choose a different file to link to.

Auto Highlite (Bitmap Cast Member Info)

For cast members that you use as buttons, use the Auto Hilite option to have the cast member automatically highlight when the user clicks it using the mouse. Even if Auto Hilite is checked, the cast member is not affected by clicking the mouse unless it is controlled by a Lingo script.

Script (Bitmap Cast Member Info)

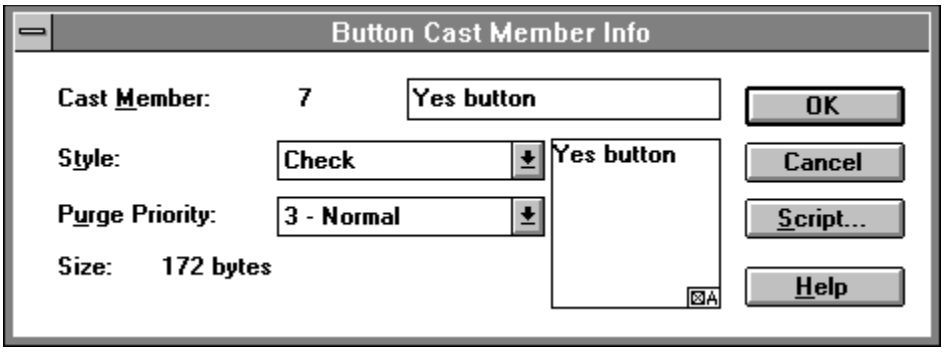
Director opens a script window for the cast member, and displays the Lingo menu in the menu bar. For more information on creating and using cast member scripts, see the *Using Lingo* manual. The script remains attached to the cast member if the cast member is cut or copied and pasted.

Button Cast Member Info

Click here to see an illustration.

Click for more information:

- Style
- Purge Priority
- Size
- Script


Button Cast Member Info
Cast Member:
7
Yes button
OK
Style:
Check
Yes button
Cancel
Purge Priority:
3 - Normal
Script...
Size:
172 bytes
Help

Style (Button Cast Member Info)

The Style drop-down list shows the button style that's currently in effect. You can display the entire list to choose a different button style.

Purge Priority (Button Cast Member Info)

Controls how Director removes the cast member from memory if memory is low.

3-Normal: The selected cast member will be removed from memory as necessary. This is the default

2-Next: The selected cast member will be among the next to be removed from memory.

1-Last: The selected cast member will be among the last to be removed from memory.

0-Never: The selected cast member remains in memory and is never purged.

Note: The more cast members you give a purge priority of Last or Never, the more likely a movie is to run out of memory. Be especially cautious using Never.

Size (Button Cast Member Info)

Displays the size of the cast member, in kilobytes.

Script (Button Cast Member Info)

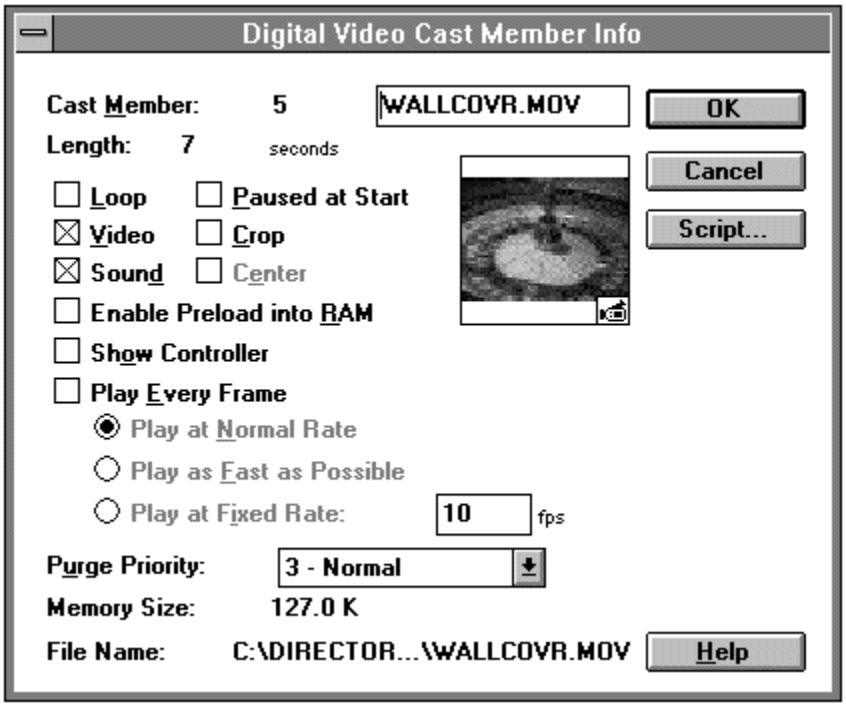
Director opens a script window for the cast member, and displays the Lingo menu in the menu bar. For more information on creating and using cast member scripts, see the Using Lingo manual. The script remains attached to the cast member if the cast member is cut or copied and pasted.

Digital Video Cast Member Info

Click here to see an illustration.

Click for more information:

- Length
- Loop
- Paused at Start
- Video
- Sound
- Crop
- Center
- Enable Preload into RAM
- Show Controller
- Play Every Frame
- Purge Priority
- Memory Size
- File Name


Digital Video Cast Member Info
Cast Member: 5
WALLCOVR.MOV
OK
Length: 7 seconds
Cancel
Loop
Paused at Start
Video
Crop
Sound
Center
Script...
Enable Preload into RAM
Show Controller
Play Every Frame
Play at Normal Rate
Play as Fast as Possible
Play at Fixed Rate:
10
fps
Purge Priority:
3 - Normal
Memory Size: 127.0 K
File Name: C:\DIRECTOR...\WALLCOVR.MOV
Help

Length (Digital Video Cast Member Info)

The length of the digital video movie, in seconds.

Loop (Digital Video Cast Member Info)

If Loop is checked, the digital video movie loops from the end back to the beginning and continues to play.

Paused at Start (Digital Video Cast Member Info)

If checked, the digital video movie is paused when it first appears on the stage (while playing the Director movie).

By default, a digital video movie starts playing the moment it first appears. If you check Paused at Start, you can later start the movie using the Lingo command:

```
set the movieRate of sprite n to R
```

where:

- `n` is the sprite number (within the current frame)
- `R` is a number representing the rate. For example, 0 = stop, 1 = normal speed, 2 = 2x speed, and -1 = reverse.

Video (Digital Video Cast Member Info)

If the Video option is checked, the digital video movie's video portion plays. If unchecked, the video portion does not play. Uncheck this option and check the Sound option if you want to play the audio-only portion of a movie.

Sound (Digital Video Cast Member Info)

If Sound is checked, the digital video movie's sound track plays. If you don't have enough memory, you can leave this option unchecked so that Director does not load the movie's sound into memory during playback.

A sound that's already playing in either sound channel overrides the sound in a digital video movie, and it also prevents the digital video movie's sound from playing even after the sound in the sound channel has stopped. Once the sound in a digital video movie has started, however, it overrides a sound in either sound channel.

Crop (Digital Video Cast Member Info)

If Crop is checked, the movie retains its original size if you resize its bounding rectangle, and its edges may get clipped. If Crop is not checked, the movie is scaled if you resize its bounding rectangle.

Center (Digital Video Cast Member Info)

Center is only available if Crop is checked. Use Center to automatically center the movie if you resize its bounding rectangle. If Center is not checked, the movie maintains its original position, relative to the upper left corner, if you resize its bounding rectangle.

Enable Preload into RAM (Digital Video Cast Member Info)

If checked, Director lets you preload the entire movie (or as much of the movie as can fit into available memory) using the `preLoad` or `preLoadCast` Lingo commands. If there is not enough memory to load the entire movie, Director loads only what can fit into memory. If this option is unchecked, Director does not load the movie into memory and instead plays it from disk. This results in slower animation speeds, since each frame must be retrieved from disk before it is played.

Show Controller (Digital Video Cast Member Info)

If this option is checked, a controller bar appears below the movie to allow the user to start, stop, and step through the movie.

Play Every Frame (Digital Video Cast Member Info)

If checked, every frame of the digital video movie plays. The digital video movie's soundtrack will not play, since the movie can't play the sound track asynchronously while the video portion plays frame-by-frame. If not checked, the movie skips frames as necessary to keep up with a constant tempo.

If Play Every Frame is checked, choose one of the following options:

- **Play at Normal Rate--**Each frame plays at its normal rate, and no frames are skipped.

- **Play as Fast as Possible--**The movie plays as fast as possible while still displaying each frame.

- **Play at Fixed Rate--**Lets you play the movie using a specific frame rate. Use this option only for digital video movies that use the same frame rate for each frame of the movie.

Purge Priority (Digital Video Cast Member Info)

Controls how Director removes the cast member from memory if memory is low.

3-Normal: The selected cast member will be removed from memory as necessary. This is the default

2-Next: The selected cast member will be among the next to be removed from memory.

1-Last: The selected cast member will be among the last to be removed from memory.

0-Never: The selected cast member remains in memory and is never purged.

Note: The more cast members you give a purge priority of Last or Never, the more likely a movie is to run out of memory. Be especially cautious using Never.

Memory Size (Digital Video Cast Member Info)

Size displays the size, in bytes, of the digital video movie's description in memory. It does not indicate the file size of the digital video movie.

File Name (Digital Video Cast Member Info)

Displays the name of the linked external file for the imported digital video cast member. Clicking the file name lets you choose a different file to link to.

Film Loop Cast Member Info

Click here to see an illustration.

Displays information about the selected film loop cast member.

Click for more information:

- Crop
- Center
- Sound
- Loop
- Purge Priority
- Size
- Script

Crop (Film Loop Cast Member Info)

If checked, the film loop retains its original size if you resize its bounding rectangle, and its edges may get clipped. If Crop is not checked, the film loop is scaled if you resize its bounding rectangle.

Center (Film Loop Cast Member Info)

This option is only available if the Crop option is checked. If checked, the film loop is automatically centered if you resize its bounding rectangle. If Center is not checked, the film loop maintains its original position, relative to the upper left corner, if you resize its bounding rectangle.

Sound (Film Loop Cast Member Info)

If checked, sound is enabled during playback. If not checked, sound is disabled during playback.

Purge Priority (Film Loop Cast Member Info)

Controls how Director removes the cast member from memory if memory is low.

3-Normal: The selected cast member will be removed from memory as necessary. This is the default

2-Next: The selected cast member will be among the next to be removed from memory.

1-Last: The selected cast member will be among the last to be removed from memory.

0-Never: The selected cast member remains in memory and is never purged.

Note: The more cast members you give a purge priority of Last or Never, the more likely a movie is to run out of memory. Be especially cautious using Never.

Loop (Film Loop Cast Member Info)

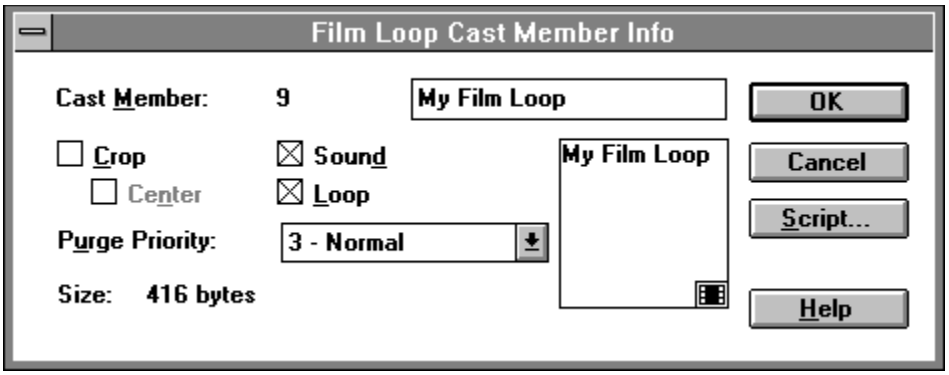
If checked, the film loop returns from the last frame back to the beginning and continues to play. If this option is not checked, the film loop doesn't loop, and the last frame remains on the Stage when the film loop finishes playing.

Size (Film Loop Cast Member Info)

Displays the size of the cast member, in bytes.

Script (Film Loop Cast Member Info)

Director opens a script window for the cast member, and displays the Lingo menu in the menu bar. For more information on creating and using cast member scripts, see the *Using Lingo* manual. The script remains attached to the cast member if the cast member is cut or copied and pasted.


Film Loop Cast Member Info
Cast Member:
9
My Film Loop
OK
Crop
Sound
Center
Loop
My Film Loop
Cancel
Script...
Purge Priority:
3 - Normal
Size:
416 bytes
Help

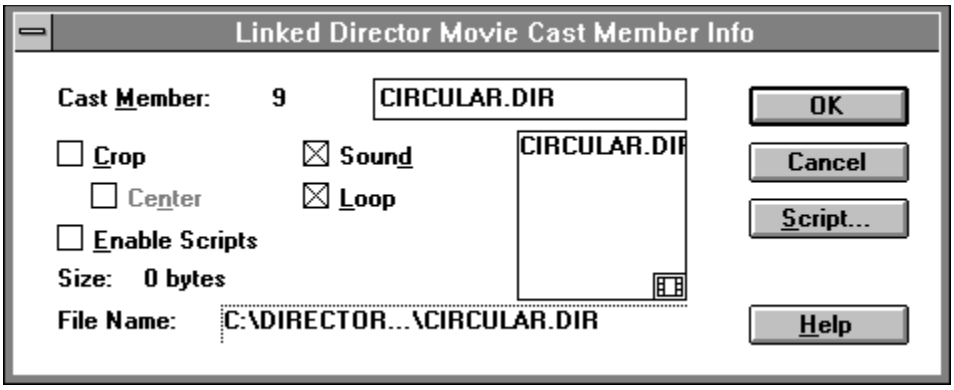
Linked Director Movie Cast Member Info

Click here to see an illustration.

Displays information about an imported movie that is linked to its source file on disk.

Click for more information:

- Crop
- Center
- Sound
- Loop
- Enable Scripts
- Size
- File Name
- Script


Linked Director Movie Cast Member Info
Cast Member: 9
CIRCULAR.DIR
OK
Crop
Sound
CIRCULAR.DIR
Cancel
Center
Loop
Script...
Enable Scripts
Size: 0 bytes
File Name: C:\DIRECTOR...\CIRCULAR.DIR
Help

Crop (Linked Director Movie Cast Member Info)

If checked, the movie retains its original size if you resize its bounding rectangle, and its edges may get clipped. If Crop is not checked, the movie is scaled if you resize its bounding rectangle.

Center (Linked Director Movie Cast Member Info)

This option is only available if Crop is checked. If checked, the movie is automatically centered if you resize its bounding rectangle. If Center is not checked, the movie maintains its original position, relative to the upper left corner, if you resize its bounding rectangle.

Sound (Linked Director Movie Cast Member Info)

If checked, sound is enabled during playback. If not checked, sound is disabled during playback.

Loop (Linked Director Movie Cast Member Info)

If checked, the movie returns from the last frame back to the beginning and continues to play. If this option is not checked, the movie doesn't loop, and the last frame remains on the stage when the movie finishes playing.

Enable Scripts (Linked Director Movie Cast Member Info)

If checked, Director activates the movie's scripts when the movie is used in the score. If this option is not checked, Director ignores the movie's scripts.

Size (Linked Director Movie Cast Member Info)

Displays the size of the cast member, in bytes.

File Name (Linked Director Movie Cast Member Info)

Displays the location of the external file associated with the linked movie. Clicking the filename lets you choose a different file to link to.

Script (Linked Director Movie Cast Member Info)

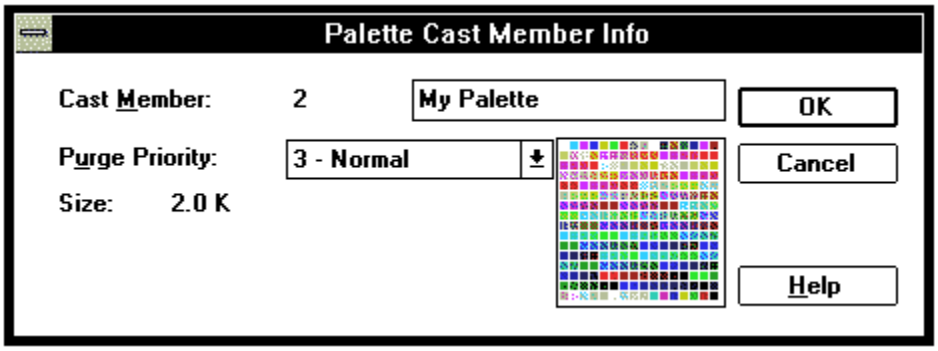
Director opens a script window for the cast member, and displays the Lingo menu in the menu bar. For more information on creating and using cast member scripts, see the *Using Lingo* manual. The script remains attached to the cast member if the cast member is cut or copied and pasted.

Palette Cast Member Info

Click here to see an illustration.

Click for more information:

- Cast Member
- Purge Priority
- Size


Palette Cast Member Info
Cast Member:
2
My Palette
OK
Purge Priority:
3 - Normal
Cancel
Size:
2.0 K
Help

Cast Member (Palette Cast Member Info)

Enter or edit the name in the entry field. The name remains attached to the cast member if it is cut or copied and pasted. Director does not prevent you from creating duplicate cast member names, but you should avoid using them. If more than one cast member with the same name has a script attached to it, Lingo uses the script attached to the cast member with the lowest cast number. Use cast names instead of cast numbers to address cast members in a Lingo script, so that you don't have to update your scripts if your cast members get renumbered or sorted.

Purge Priority (Palette Cast Member Info)

Controls how Director removes the cast member from memory if memory is low.

3-Normal: The selected cast member will be removed from memory as necessary. This is the default

2-Next: The selected cast member will be among the next to be removed from memory.

1-Last: The selected cast member will be among the last to be removed from memory.

0-Never: The selected cast member remains in memory and is never purged.

Note: The more cast members you give a purge priority of Last or Never, the more likely a movie is to run out of memory. Be especially cautious using Never.

Size (Palette Cast Member Info)

Displays the size of the cast member.

PICT Cast Member Info

Click here to see an illustration.

Click for more information:

- Cast Member
- Purge Priority
- Size
- Script

Cast Member (PICT Cast Member Info)

Enter or edit the name in the field. The name remains attached to the cast member if it is cut or copied and pasted. Director does not prevent you from creating duplicate cast member names, but you should avoid using them. If more than one cast member with the same name has a script attached to it, Lingo uses the script attached to the cast member with the lowest cast number. Use cast names instead of cast numbers to address cast members in a Lingo script, so that you don't have to update your scripts if your cast members get renumbered.

Purge Priority (PICT Cast Member Info)

Controls how Director removes the cast member from memory if memory is low.

3-Normal: The selected cast member will be removed from memory as necessary. This is the default

2-Next: The selected cast member will be among the next to be removed from memory.

1-Last: The selected cast member will be among the last to be removed from memory.

0-Never: The selected cast member remains in memory and is never purged.

Note: The more cast members you give a purge priority of Last or Never, the more likely a movie is to run out of memory. Be especially cautious using Never.

Size (PICT Cast Member Info)

Displays the size of the cast member, in bytes.

Script (PICT Cast Member Info)

PICT cast members can have a Lingo script associated with them. To create or review the script, click the Script button in the dialog box. Director opens a script editor window for the cast member, and displays the Lingo menu in the menu bar. For more information on creating and using cast member scripts, see the *Using Lingo* manual. The script remains attached to the cast member if it is cut or copied and pasted.

PICT Cast Member Info

Cast Member: 2 SHRINE.PCT

Purge Priority: 3 - Normal

Size: 101.8 K

OK

Cancel

Script...

Help

Script Cast Member Info

Click here to see an illustration.

Click for more information:

- Cast Member
- Size
- Type

Cast Member (Script Cast Member Info)

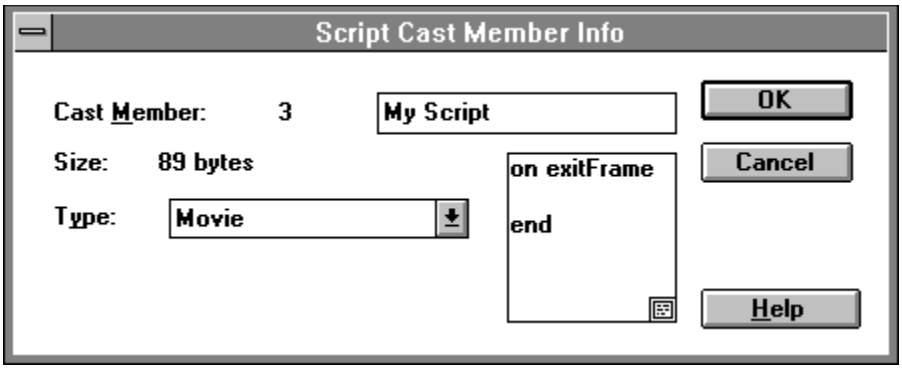
Enter or edit the name in the field. The name remains attached to the cast member if it is cut or copied and pasted. Director does not prevent you from creating duplicate cast member names, but you should avoid using them. If more than one cast member with the same name has a script attached to it, Lingo uses the script attached to the cast member with the lowest cast number. Use cast names instead of cast numbers to address cast members in a Lingo script, so that you don't have to update your scripts if your cast members get renumbered.

Size (Script Cast Member Info)

Displays the size of the cast member, in bytes.

Type (Script Cast Member Info)

The Type drop-down list displays the script type (movie or score) for the selected cast member, and lets you change it. A movie script's handlers are global, and can be called from other scripts. A score script's handlers are local, and cannot be called from other scripts. If you change a movie script into a score script, it appears in the script drop-down list in the score.


Script Cast Member Info
Cast Member: 3
My Script
OK
Size: 89 bytes
Cancel
Type: Movie
on exitFrame
end
Help

Shape Cast Member Info

Click here to see an illustration.

Click for more information:

- Shape
- Filled
- Script

Shape drop-down list (Shape Cast Member Info)

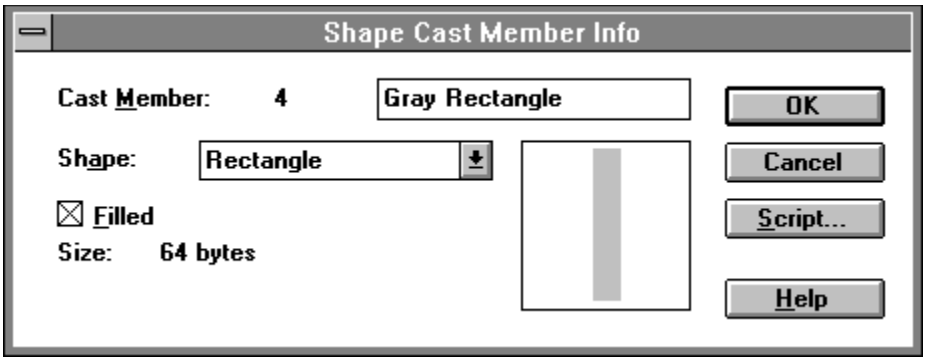
The Shape drop-down list shows the shape that's currently in effect. You can display the entire list to choose a different shape.

Filled (Shape Cast Member Info)

If Filled is checked, the shape will be filled using the current fill pattern and colors as specified in the tools window.

Script (Shape Cast Member Info)

Director opens a script window for the cast member, and displays the Lingo menu in the menu bar. For more information on creating and using cast member scripts, see the *Using Lingo* manual. The script remains attached to the cast member if the cast member is cut or copied and pasted.


Shape Cast Member Info
Cast Member: 4
Gray Rectangle
OK
Shape:
Rectangle
Cancel
Filled
Script...
Size: 64 bytes
Help

Sound Cast Member Info

Click here to see an illustration.

Click for more information:

- Looped
- Purge Priority
- Size
- File
- Play

Looped (Sound Cast Member Info)

If checked, the sound plays continuously. If not checked, the sound plays once, even if the movie loops.

Note: Director can loop mono or stereo sounds.

Purge Priority (Sound Cast Member Info)

Controls how Director removes the cast member from memory if memory is low.

3-Normal: The selected cast member will be removed from memory as necessary. This is the default

2-Next: The selected cast member will be among the next to be removed from memory.

1-Last: The selected cast member will be among the last to be removed from memory.

0-Never: The selected cast member remains in memory and is never purged.

Note: The more cast members you give a purge priority of Last or Never, the more likely a movie is to run out of memory. Be especially cautious using Never.

Size (Sound Cast Member Info)

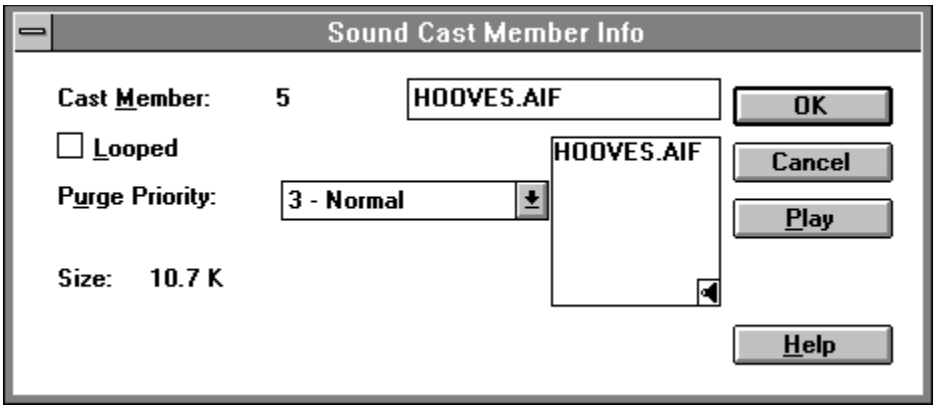
Displays the size of the cast member, in kilobytes.

File (Sound Cast Member Info)

If a cast member is an external sound file that is linked to its source file on disk, the Cast Member Info dialog box displays the location of the linked file. Clicking the file name lets you choose a different file to link to.

Play (Sound Cast Member Info)

Click the Play button to preview the sound at its pre-recorded sampling rate.


Sound Cast Member Info
Cast Member:
5
HOOVES.AIF
OK
Looped
HOOVES.AIF
Cancel
Purge Priority:
3 - Normal
Play
Size:
10.7 K
Help

Text Cast Member Info

Click here to see an illustration.

Click for more information:

- Style
- Editable Text
- Auto Tab
- Don't Wrap
- Purge Priority
- Size
- Script

Style (Text Cast Member Info)

Use the Style drop-down list to choose a text display option for the text cast member.

Adjust to Fit--Causes the text box to expand vertically when text is entered that extends beyond the current size of the box.

Scrolling--Attaches a scroll bar to the right side of the text box. Useful for a large amount of text.

Fixed--Causes the box to retain its original size. If text is entered that extends beyond the limits of the box, the text is not displayed.

Limit to Field Size--Sets the field's width to be fixed to the size of the field. Characters that don't fit are ignored.

Editable Text (Text Cast Member Info)

Allows the text cast member to be edited during movie playback. You can use this option instead of using the Lingo command set `the editableText of sprite` to true.

Note: If you set a text cast member to be editable in the Cast, it is always editable when used in the Score. Director ignores the Score's Editable checkbox setting for the cast member.

Style (Text Cast Member Info)

Causes the Tab key to advance to the next editable text field on the stage during playback. Note that the Editable Text checkbox must be checked, or the Lingo command `set the editableText of sprite to true` must be specified for this option to have any effect.

Auto Tab (Text Cast Member Info)

If checked, prevents text from moving to the next line. Text that extends beyond the right edge is truncated. You must use the Enter key to generate a new line.

Purge Priority (Text Cast Member Info)

Controls how Director removes the cast member from memory if memory is low.

3-Normal: The selected cast member will be removed from memory as necessary. This is the default

2-Next: The selected cast member will be among the next to be removed from memory.

1-Last: The selected cast member will be among the last to be removed from memory.

0-Never: The selected cast member remains in memory and is never purged.

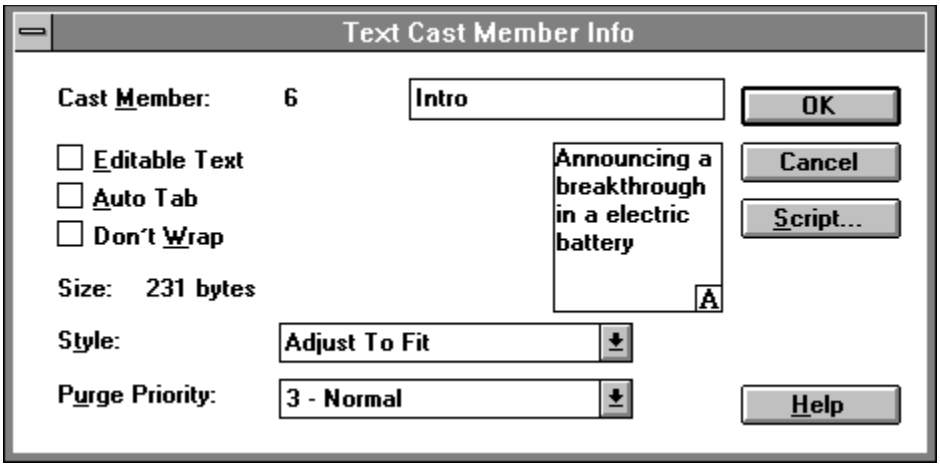
Note: The more cast members you give a purge priority of Last or Never, the more likely a movie is to run out of memory. Be especially cautious using Never.

Size (Text Cast Member Info)

Displays the size of the cast member, in bytes.

Script (Text Cast Member Info)

Director opens a script window for the cast member, and displays the Lingo menu in the menu bar. For more information on creating and using cast member scripts, see the Using Lingo manual. The script remains attached to the cast member if the cast member is cut or copied and pasted.


Text Cast Member Info
Cast Member:
6
Intro
OK
Editable Text
Auto Tab
Don't Wrap
Announcing a breakthrough in a electric battery
Cancel
Script...
Size:
231 bytes
Style:
Adjust To Fit
Purge Priority:
3 - Normal
Help

Cast Member Info for Multiple Items

You can select multiple cast members with Shift-click or Control-click to get information about the number, type, and total size of your selection, and to set the purge priority for the selected cast members when memory is low.

Click here to see an illustration.

Click for more information:

- Cast Member
- Type
- Size
- Purge Priority

Cast Member (Cast Info for Multiple Items)

Displays the number of cast members in the selection.

Type (Cast Info for Multiple Items)

When several cast members are selected, this is set to "Multiple." If all the selected cast members are of the same type, this field instead displays their type.

Size (Cast Info for Multiple Items)

Displays the total size, in kilobytes, of the selected cast members.

Purge Priority (Cast Info for Multiple Items)

Controls how Director removes the cast members from memory if memory is low.

3-Normal: The selected cast member will be removed from memory as necessary. This is the default

2-Next: The selected cast member will be among the next to be removed from memory.

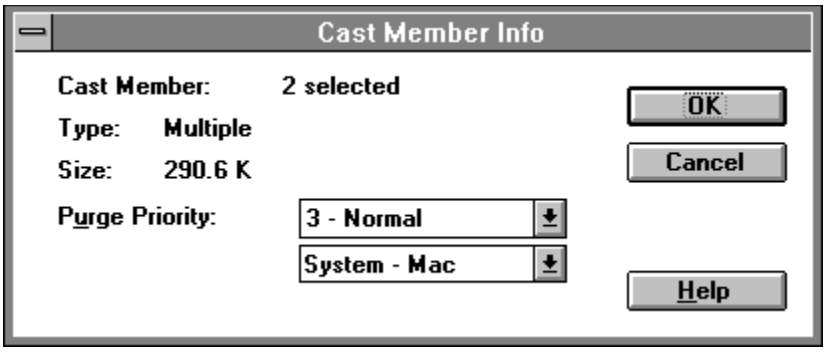
1-Last: The selected cast member will be among the last to be removed from memory.

0-Never: The selected cast member remains in memory and is never purged.

Note: The more cast members you give a purge priority of Last or Never, the more likely a movie is to run out of memory. Be especially cautious using Never.

If all cast members have the same purge priority, the drop-down list is blank.

Note: If you set many cast members to have a purge priority of Last or Never, your movie may run out of memory.


Cast Member Info
Cast Member:
2 selected
Type:
Multiple
Size:
290.6 K
Purge Priority:
3 - Normal
System - Mac
OK
Cancel
Help

Open Script command

Opens the script associated with the selected cast member. This is the same as clicking the Script button in the Cast Member Info dialog box.

In the cast window, select a cast member and click the Open Script button as a shortcut for choosing this command.

If you are editing the cast member in a paint, text, or digital video window, you can click the window's Script button as a shortcut for choosing this command.

Keyboard shortcut: Control-' is the keyboard shortcut for this command. Press Alt while choosing Open Script to open the selected cast member's script in a new script window.

Edit Cast Member command

This command is only available if there is a selection in the cast window. It displays the appropriate editing window for the selected cast member. For example, if you select a bitmap cast member and choose this command, Director opens the paint window for the selected cast member. For cast members that don't have editing windows (e.g., Shape, PICT, Sound, Movie, and Film Loop cast members) Director displays the cast member's Cast Member Info dialog box.

Tip: Double-click a cast member in the cast window as shortcut for choosing this command.

Convert to Bitmap command

This command converts text objects created in the text window or with the text tool in the tool window to bitmapped cast members. The converted text can then be edited in the paint window. Once you convert text objects to bitmapped graphics, you cannot undo the change.

Using bitmapped text gives you more choices of inks to use on the text. Another advantage to using bitmapped text is that once created, the text is not dependent on the available fonts to be displayed. On the screen, text objects and bitmapped text look identical.

There is a trade-off in converting to bitmap. Bitmapped text and graphics animate faster, but they may take up more memory and disk space. They also have jagged edges when printed on a laser printer.

Transform Bitmap command

This command lets you change the size, color depth, and palette of one or more selected cast members. The palette you choose is applied to the cast members and cannot be undone. If you want to maintain the cast member's original palette while temporarily applying a different palette, use the Cast Member Info command instead.

Click to see an illustration of the dialog box.

Click for more information:

- Size
- Colors
- Remapping
- Importing cast members with different palettes

Note: If you only want to change the size of the cast member on the stage, use the Sprite Info command in the Score menu.

The dialog box displays the values for the current cast member selection. If more than one cast member is selected, a blank value indicates that some cast members in the selection have different values. To maintain a cast member's original value, leave that value blank in the dialog box.

Size (Transform Bitmap)

Scale--Use the Scale option to reduce or enlarge the cast member by a percentage. If multiple cast members are selected, you can scale them all relative to their original size.

Width, Height--Use the Width and Height fields to specify the horizontal and vertical dimensions of the cast member in pixels. If multiple cast members are selected, you can resize all the cast members to the dimensions you enter.

Colors (Transform Bitmap)

Color Depth--You may want to reduce a cast member's color depth to save memory and disk space. On the other hand, you may want to increase a black-and-white cast member's color depth so you can use the paint tools to colorize it.

A cast member's color depth setting can be 1-bit color (black and white), 2-bit color (4 colors), 4-bit color (16 colors), or 8-bit color (256 colors). Larger color depths require more memory and may also increase the time it takes Director to animate a cast member. Black and white cast members require the least amount of memory.

Director never displays more than 256 colors on the stage at one time regardless of a cast member's color depth. That means that if you use a 16- or 24-bit cast member, Director displays it using the colors from its 256-color palette that most closely match the cast member's original colors. Furthermore, there's a performance penalty if the cast member's color depth is either larger or smaller than the color depth you've selected in Windows display driver: when the color depth of a cast member and the display driver are different, Director must remap the cast member every time it draws the cast member. That's a good reason to change a cast member's depth to match the display driver's.

If you reduce a cast member's color depth to 1 bit, you can still select colors for a sprite based on the cast member. Just select the sprite and choose a color from the tools window foreground or background color pop-up palette. (Don't use the paint window's color palette.) Keep in mind that when you change the color, you're changing only the color of the sprite on the stage. The cast member remains black and white.

Palette--The palettes listed in the pop-up menu are the default Director palettes, plus any additional palettes found in the cast window. You can create a common palette that contains most of the colors your cast member needs.

When you use the Transform Bitmap command, Director matches the colors of the cast member with similar colors in the new palette, regardless of the position of the original color in the palette. For example, if the original artwork is red and the only red available in the new palette is pink, the red is changed to pink.

If the movie is playing, the active palette is the one that is currently in use at any given time, as specified in the score. The active palette may be different from the palette used by the movie.

Remapping (Transform Bitmap)

Remapping the cast member to the current palette causes the cast member to appear in the colors closest to those in the current palette. Importing the cast member's palette ensures that it looks as it did when originally created.

Dither--Use Dither to improve the color or shading of the cast member. Director examines the palette and creates a blend from one color to another using the nearest matching colors.

Remap to Closest Colors--When you create a cast member in a color paint application, it is created with a specific palette. If you want to use that cast member in a movie, but a different palette is set in the palette channel of the score, your cast member will not appear in the colors you intended. Use this option to remap the cast member to the colors in the current palette that are closest to the colors the cast member was originally created with.

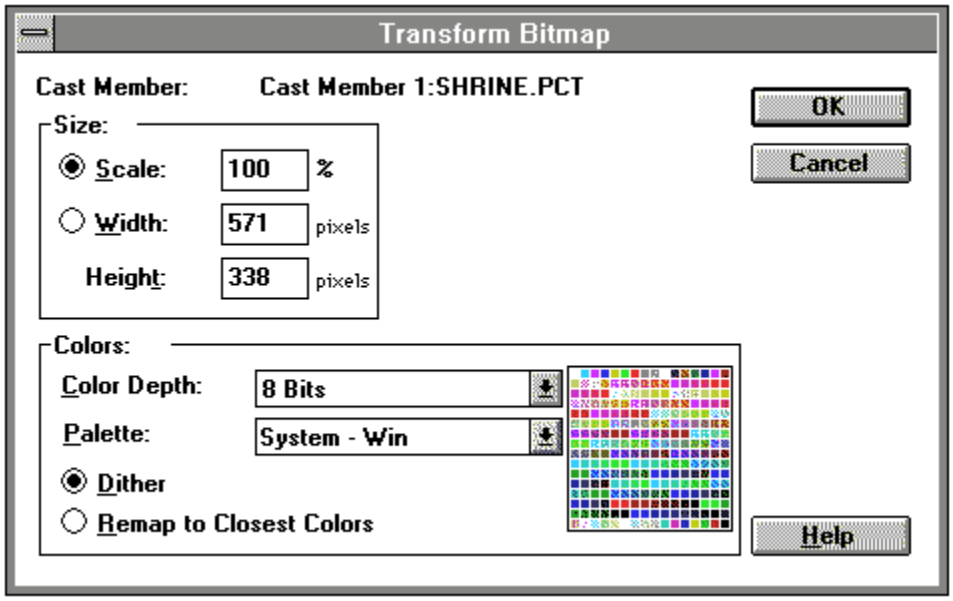
Importing cast members with different palettes

When importing cast members with palettes that are different from the currently active palette, a dialog box appears asking you if you want to remap the cast member to the current palette, or import the cast member and the cast member's palette.

If you import the cast member and the palette, they will both be placed in the next available cast member positions in the cast window.

Remapping the cast member to the current palette causes the cast member to appear in the colors closest to those in the current palette. Importing the cast member's palette ensures that it looks as it did when originally created.

Note: Remapping cast members is a one-way street. If you remap a colorful cast member to a grayscale palette, then remap again to the cast member's original color palette, the cast member may remain gray. The cast member always maps to the colors in the new palette that most closely resemble its current colors.


Transform Bitmap
Cast Member:
Cast Member 1:SHRINE.PCT
OK
Cancel
Size:
Scale:
100
%
Width:
571
pixels
Height:
338
pixels
Colors:
Color Depth:
8 Bits
Palette:
System - Win
Dither
Remap to Closest Colors
Help

Align Bitmaps command

Align Bitmaps arranges cast members so that their registration points line up in the paint window. A registration point lets you align a series of cast members using a fixed reference point. When the registration points are aligned, you can use the left and right arrows in the paint window to flip through the cast members and preview a sequence of animated cast members. Align Bitmaps has no effect on sprite positions in the score.

Cast to Time command

Cast to Time is a quick way to create a cast member sequence in your movie. This command places selected cast members in the cast window sequentially into the score.

If you select a single cell in the score before choosing this command, the selected cast members are added to the score beginning at the selected cell. Any existing score data is replaced by the Cast to Time sequence. If you set an insertion point in the score before choosing this command, the Cast to Time sequence is inserted in Channel 1, beginning at the insertion point. If you select a range of score cells before choosing this command, the Cast to Time sequence that is inserted will only be as long as the number of selected cells in the score.

Tip: You can also place selected cast members across time in the score by holding down the Alt key while dragging them from the cast.

Duplicate Cast Member command

The Duplicate Cast Member command (Control-D) duplicates the selected cast member and pastes the duplicate into the next available position in the cast window.

If the paint, text, digital video, or script window is open when you choose this command, the contents of the window changes to the new cast member. The name and registration point of the duplicate is the same as the name and registration point of the original cast member.

This command cannot be used with a cast member that is part of a shared cast.

This is a quick way to create a series of cast members for frame-by-frame animation. Duplicate a cast member, change it slightly, duplicate the changed cast member, alter it some more, duplicate it again, and so on.

Tip: If the cast window is front-most, you can select multiple cast members and use this command to duplicate the entire selection at the same time.

Find Cast Members command

Find Cast Members permits you to quickly identify those cast members that are either unused in the current movie or are created with one of the palettes in the palette pop-up menu. You can also select cast members by name or type. Use this command to identify cast members to clear from your movie or candidates for remapping to another palette.

Click to see an illustration of the dialog box.

Tip: Control-; (semicolon) is the keyboard shortcut for this command.

The dialog box lists cast members in the current movie, including shared cast members.

Find Cast members:

Whose type is--Choose a cast member type from the menu if you only want to find cast members of a specific type. (The default is All.)

Whose name begins with--Finds cast members whose name begins with the characters you enter.

That use the palette--Finds cast members that use the chosen palette.

That are not used in the score--Finds cast members that are not used in the score. Keep in mind, however, that a cast member that is not used in the score may still be used in a Lingo command.

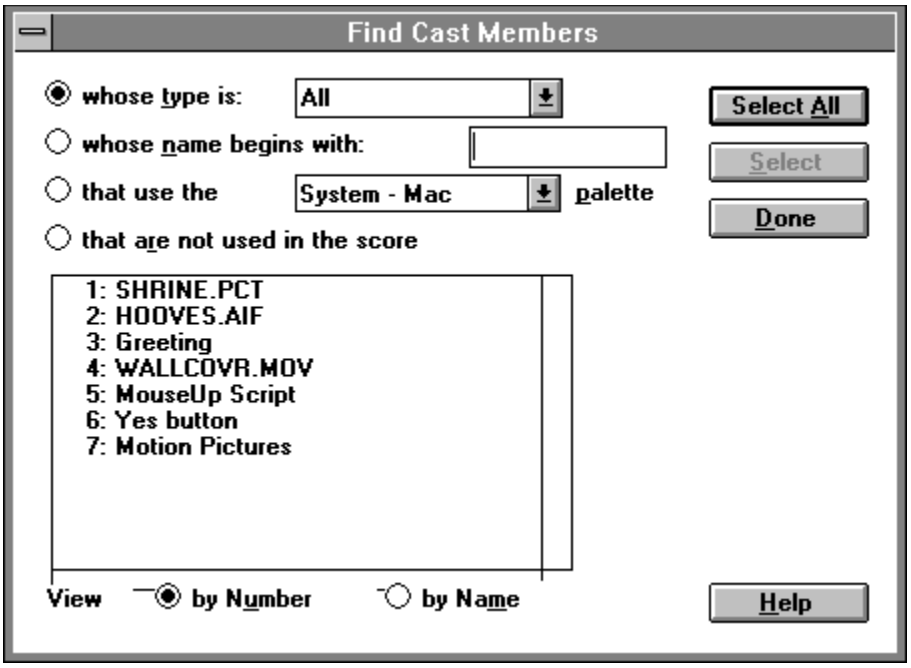
View by Number--Sorts cast members by cast member number.

View by Name--Sorts cast members alphabetically by name.

Selecting a cast member in the list and clicking **Select** (or double-clicking a cast member) closes the dialog box and selects the cast member in the cast. Clicking **Select All** closes the dialog box and selects all matching cast members in the cast.

Clicking **Done** closes the dialog box without affecting the selection in the cast window.

Tip: To quickly select cast members by name, begin typing the first few letters of the name, and the dialog box will automatically display a list of cast members whose name begins with the letters you type.


Find Cast Members
whose type is:
All
whose name begins with:
that use the
System - Mac
palette
that are not used in the score
Select All
Select
Done
1: SHRINE.PCT
2: HOOVES.AIF
3: Greeting
4: WALLCOVR.MOV
5: MouseUp Script
6: Yes button
7: Motion Pictures
View
by Number
by Name
Help

Sort Cast Members command

This command lets you rearrange selected cast members in the cast and eliminate empty cast member positions in between non-empty ones. To rearrange the entire cast, first choose Select All from the Edit menu before choosing this command.

Click to see an illustration of the dialog box.

Director automatically updates the score with the new frame number for each repositioned cast member and the animation sequence is unaltered.

This command is particularly useful when you want to reorganize a selection of cast members or the entire cast in a compact format.

Sort Selection:

By Name--Sorts selected cast members alphabetically by name.

By Type--Sorts selected cast members by type (in the following order: bitmap, palette, button, text, sound, shape, pict, digital video, film loop, movie, script).

By Size--Sorts selected cast members by file size, in decreasing size order.

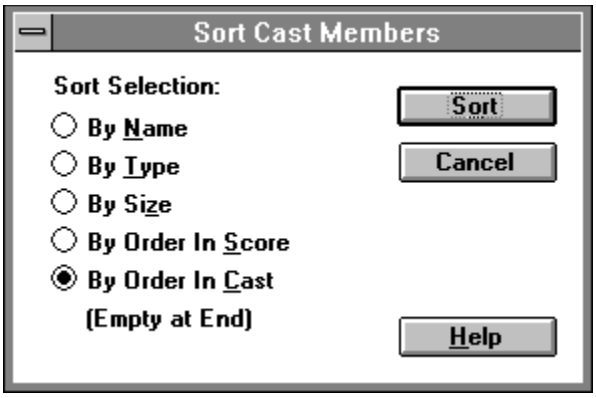
By Order in Score--Sorts selected cast members in the order in which they appear in the Score. If a cast member does not appear in the Score it is placed after all the cast members that are referenced from the Score.

By Order in Cast--Sorts selected cast members in the order in which they appear in the Cast. Empty cast members are placed at the end.

Notes:

- Using this command will not affect shared cast members, since these cast members cannot be rearranged. To rearrange shared cast members, you must open the shared cast movie, called SHARED.DIR.

- Because cast numbers may change when you use this command, cast number references in scripts may become invalid. If you use Sort Cast Members, you may have to go through your scripts to update them with the new cast numbers. Use cast names instead of cast numbers to address cast members in a Lingo script so that you don't have to worry if your cast members get re-numbered.


Sort Cast Members
Sort Selection:
By Name
By Type
By Size
By Order In Score
By Order In Cast
(Empty at End)
Sort
Cancel
Help

Cast Window Options command

Displays a dialog box that lets you control cast window appearance. These preferences are stored in the Director 4.0 Preferences file (DIRECTOR.PRF).

Click to see an illustration of the dialog box.

Click for more information:

- Maximum Visible
- Row Width
- Thumbnail Size
- Cast ID Style
- Cast Type Icons
- Indicate Cast Members With Scripts

Maximum Visible (Cast Window Options)

Specifies the maximum number of cast members displayed in the Cast window. Note that this option does not limit the actual number of cast members that can exist in the cast. If you have a small number of cast members, you can hide the remaining unused cast slots and make better use of the vertical scrollbar. The default is 1000.

Row Width (Cast Window Options)

Determines how many thumbnails are displayed in each row in the cast window. Eight, Ten, and Twenty specify fixed-row widths that are independent of the window size; if the cast window is smaller horizontally than the width of the cast row, you must use the horizontal scrollbar to reveal the rest of the cast. The Fit to Window option automatically adjusts the number of cast members per row to fit the current width of the cast window. In this mode, the horizontal scrollbar is disabled, since the entire width of the cast is always in view. The default is Fit to Window.

Thumbnail Size (Cast Window Options)

The size of each cast thumbnail image displayed in the Cast window. Thumbnails always maintain the standard 4:3 aspect ratio. The default is Medium.

Cast ID Style (Cast Window Options)

Selects the display format of the cast member ID displayed below each cast thumbnail image in the cast window. The chosen format is also used in other windows, whenever a cast ID is displayed. The default is Number:Name.

Number--Displays cast number in decimal format.

Name--Displays cast name, if one exists; otherwise displays cast number in decimal format.

A11 (Octal)--Displays cast number in octal format, as used in previous versions of Director. Octal format is no longer supported in Lingo and should only be used when working with movies created using an earlier version of Director.

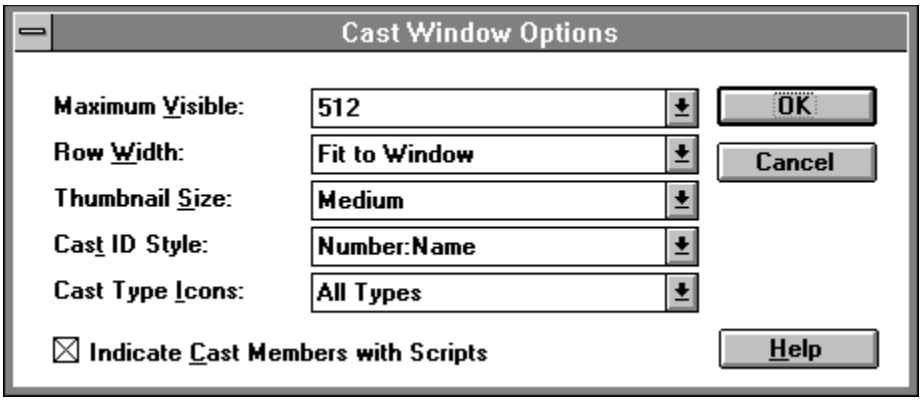
Number:Name--Displays cast number (in decimal format) and cast name, separated by a colon (":")., i.e., "340:Snoopy". If a name does not exist, just displays the cast number in decimal format. This is the default.

Cast Type Icons (Cast Window Options)

Controls how Director displays an icon in the lower right corner of each cast member, indicating the cast member's type.

Indicate Cast Members With Scripts (Cast Window Options)

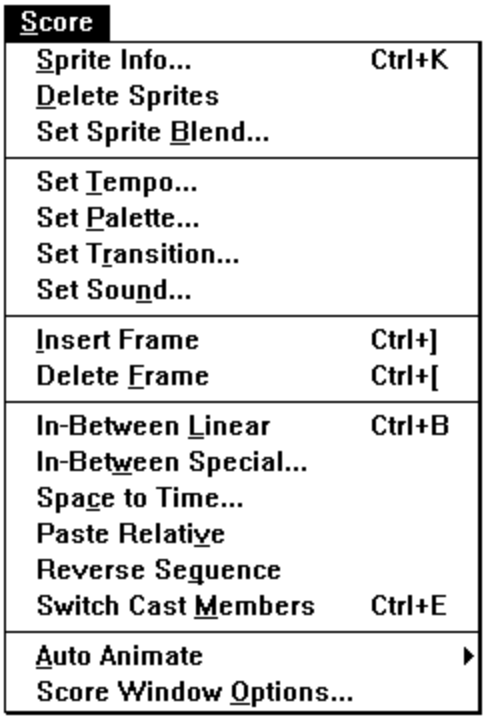
If checked, Director displays a script indicator in the lower left corner of each script cast member, to indicate that the cast member uses a script.


Cast Window Options
Maximum Visible:
512
Row Width:
Fit to Window
Thumbnail Size:
Medium
Cast ID Style:
Number:Name
Cast Type Icons:
All Types
Indicate Cast Members with Scripts
OK
Cancel
Help

Score menu

The Score menu contains commands for editing the score and setting effects.

Click a menu command for more information:



Sprite Info command

The Sprite Info command (Control-K) lets you change the size and location of a selected sprite on the stage. This command only affects the sprite's appearance on the stage and does not alter the actual size of the cast member.

Click here to see an illustration.

If you selected more than one sprite, this command lets you edit them as a group.

Click for more info:

Size
Location

Size (Sprite Info)

These options let you change the sprite's size.

Scale--To scale the sprite's appearance on the stage by a percentage, enter the percentage in the Scale % field. The sprite is scaled relative to its current size, not to the size of its parent cast member.

Width, Height--Enter an specific width and height for the sprite, in pixels.

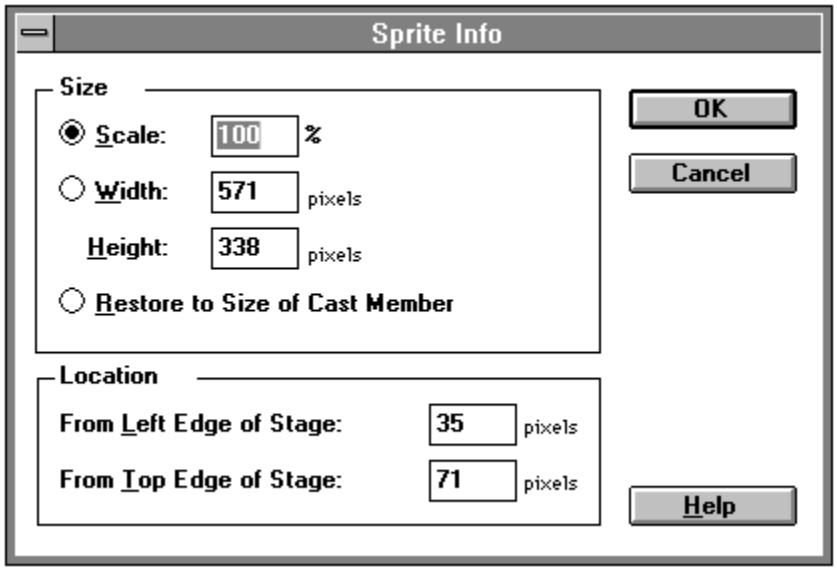
Restore to Size of Cast Member--Returns the sprite to the size it was when you first moved it into the score or onto the stage.

Location (Sprite Info)

These options let you change the top left corner position of the sprite on the stage.

From Left Edge of Stage--The value you enter represents the number of pixels the sprite is offset from the left edge of the stage.

From Top Edge of Stage--The value you enter represents the number of pixels the sprite is offset from the top of the stage.


Sprite Info
Size
Scale:
100
%
Width:
571
pixels
Height:
338
pixels
Restore to Size of Cast Member
Location
From Left Edge of Stage:
35
pixels
From Top Edge of Stage:
71
pixels
OK
Cancel
Help

Delete Sprites command

Delete Sprites (Control-Backspace) removes selected artwork from the stage. Whatever is selected in the score window or on the stage is removed from your animation. Delete Sprites has no effect on the cast.

Delete Sprites is particularly useful when you want to clear sprites from the stage when the score window is closed or you have several windows open and want to avoid inadvertently deleting a cast member from the cast window.

Set Sprite Blend command

Set Sprite Blend (Control-Alt-B) lets you specify the blend level for selected sprites in the score. This command allows you to apply a blend effect to sprites that uses the blend, background transparent, mask, or matte ink. You can use this command to "fade" cast members in or out on the stage as they are animating.

Click here to see an illustration.

You cannot apply a blend value to PICT sprites. You cannot apply a blend value to a digital video cast member if the Direct to Stage option is checked in the Cast Member Info dialog box.

Each sprite in the same frame can store its own blend value.

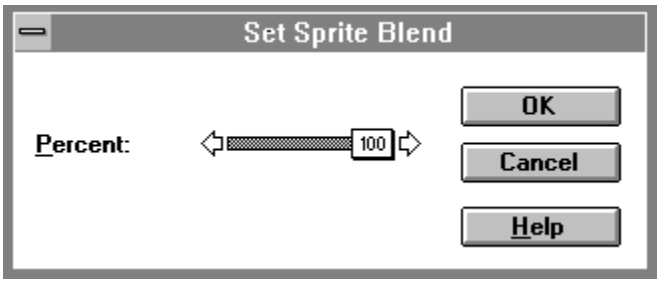
Use the score's blend display option to see the blend values. The score has a blend display option.

Choosing a value sets the blend value as a percentage for the selected cells.

To set blend values to fade cast members in or out:

1. Select the first cast members and assign them a starting blend value.
2. Select the last cast members and assign them an ending blend value.
3. Select the entire range of cast members and choose the In-Between Special command from the Score menu.

   Make sure only the Blend checkbox is checked in the In-Between Special dialog box.
4. Click OK.


Set Sprite Blend
Percent:
100
OK
Cancel
Help

Set Tempo command

The Set Tempo command lets you set a tempo or pause in your movie. The tempo setting controls the speed at which the playback head moves from frame to frame.

Click here to see an illustration.

Click for more information:

- Tempo
- Wait
- Wait for Mouse Click or Key
- Wait For Sound1 to Finish
- Wait For Sound2 to Finish
- Wait for QuickTime Movie to Finish in Channel

Select a cell in the tempo channel of the score window and choose Set Tempo from the Score menu. When you set a tempo, it also applies to all frames to the right of the tempo setting, until another tempo is encountered in the tempo channel.

Tip: Double-clicking a cell in the Tempo channel opens the Set Tempo dialog box.

Related topic: Using transitions

Tempo (Set Tempo)

Use the scroll bar to set the movie's tempo. This is the same as changing the tempo in the control panel.

Wait (Set Tempo)

Set the amount of delay in your movie using the scroll bar.

Wait for Mouse Click or Key (Set Tempo)

Lets you pause the playback head until the user clicks the mouse or presses a key. The cursor changes to a blinking mouse to indicate that the movie is paused.

You might want to include some text on the screen telling users to click the mouse or press a key to continue.

Wait For Sound1 to Finish (Set Tempo)

Lets you pause the playback head until a sound in sound channel 1 finishes playing.

Wait For Sound2 to Finish (Set Tempo)

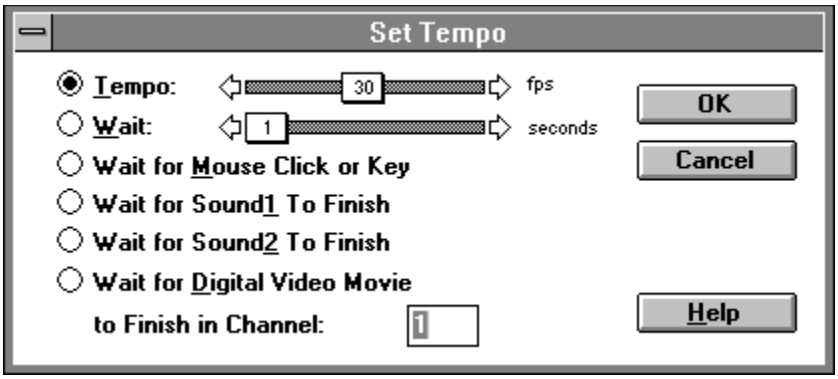
Lets you pause the playback head until a sound in sound channel 2 finishes playing.

Wait for Digital Video Movie to Finish in Channel (Set Tempo)

Lets you pause the playback head until a QuickTime movie finishes playing. If you have several QuickTime movies playing, make sure you specify the channel of the longest playing movie when using this option.

Using transitions

If you place tempo settings in the same frame as a transition, some Tempo channel settings such as wait and sound playing become disabled. To avoid this, don't place a transition in the same frame as your tempo settings. Instead, place the tempo settings in the frame immediately before or after the transition.


Set Tempo
Tempo:
30
fps
Wait:
1
seconds
Wait for Mouse Click or Key
Wait for Sound1 To Finish
Wait for Sound2 To Finish
Wait for Digital Video Movie
to Finish in Channel:
1
OK
Cancel
Help

Set Palette

The Set Palette command allows you to change palettes in a specific frame of the movie while your animation is playing. When you set a palette in the palette channel of the score, the color of the cast members in the movie will be determined by that palette, until another palette is encountered in the palette channel.

Click for more information:

Color Cycling
Palette Transition

Palettes can be transitioned from one to another during an animation. A simple form of this occurs when you set two palettes in the palette channel of the score with the Set Palette command in the Score menu. When you play the movie, the cast members change from the colors of one palette to the other. In fact, everything on the screen will change color, including the Apple in the menu bar.

The cast members' new colors depend upon the position of the previous colors. For example, if one of your cast members is yellow, and yellow occupies the fifth color in your palette, the cast member will become whatever color is in the fifth position in the new palette.

To get a visual demonstration of color cycling or palette transitions, choose a range of frames in the score to cycle or transition colors in, and open the color palettes window while the movie plays. You can see the colors transition or cycle in the open color palettes window.

Tip: Double-clicking a cell in the palette channel opens the Set Palette dialog box.

Color Cycling (Set Palette)

Click here to see an illustration of the dialog box with Color Cycling selected.

This permits you to cycle a range of colors in a palette. For example, if a cast member's color is the fifth color in the palette, and you specify a range of colors four, five, and six, the cast member will change colors when the movie is played, and it will cycle through colors four, five, and six.

To select a range to cycle, drag across the colors to be cycled in the Set Palette dialog box. You can also click a color, and Shift-click another color to select the colors you click and all colors in between.

Palette--Lets you choose the palette used for the selected cells in the palette channel.

Palette Transition (Set Palette)

Click here to see an illustration of the dialog box with Palette Transition selected.

Transition makes a smooth transition from one palette to another. Instead of your cast member abruptly changing colors when you switch palettes, this option will gradually blend from one palette to the next.

Speed--Lets you control how fast or slowly the palettes transition from one to another.

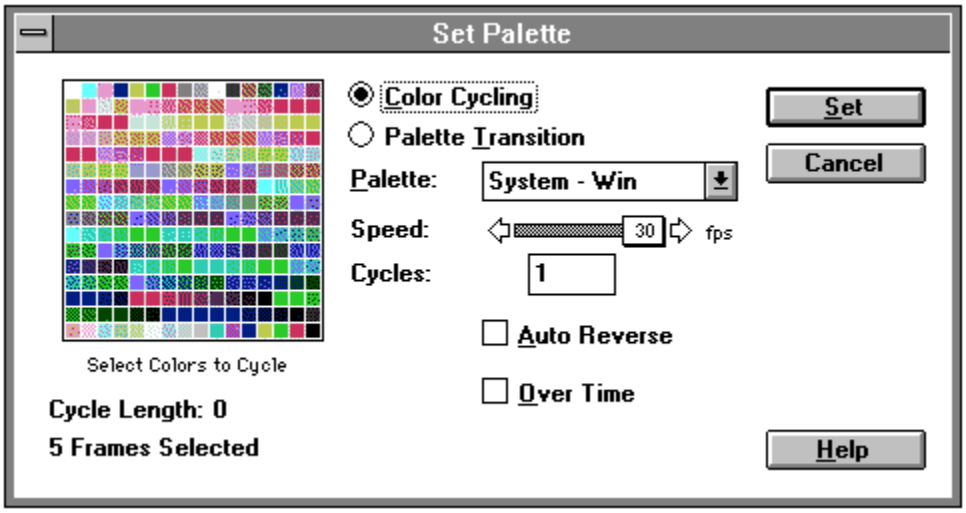
Normal--The screen doesn't fade during the palette transition.

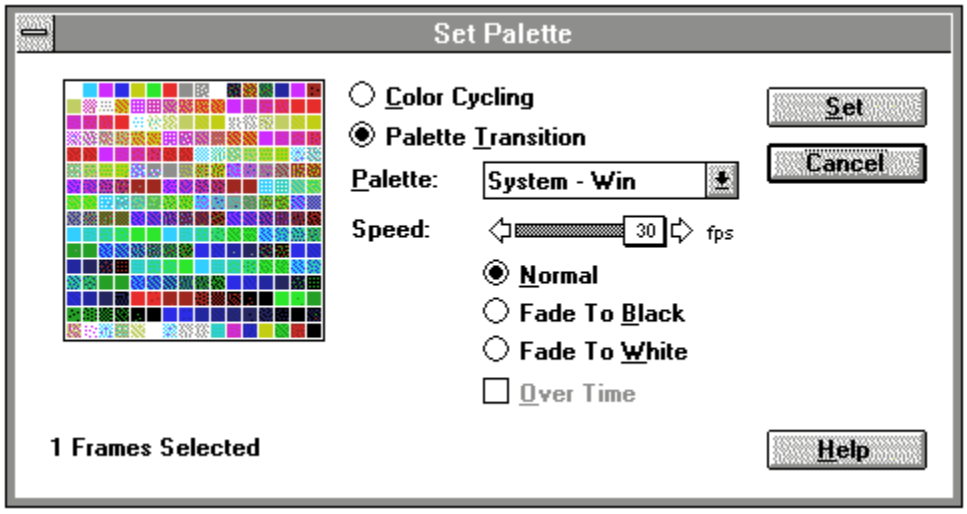
Fade to Black--Like other palette transitions, this can occur over time, or in between frames. This option makes the screen fade completely to black. A nice way to end a movie is to make the final frame a black cast member that covers the whole stage, and then fade to black over several frames before the last frame.

Fade to White--Like other palette transitions, this can occur over time, or in between frames. This option makes the screen fade completely to white. A nice way to end a movie is to make the final frame a white cast member that covers the whole stage, and then fade to white over several frames before the last frame.

Over Time--By default, the palette transition will take place between two frames. All movement will halt as the transition takes place. If you want the transition to occur over a number of frames, check this option. The range of cells you select before choosing Set Palette determines the length of the transition.

Note: Color cycling doesn't work in 16- and 24-bit environments. That's because color cycling can happen only when Director is using palettes and when you've set Windows display driver to 256 colors.


Set Palette
Color Cycling
Palette Transition
Palette:
System - Win
Speed:
30
fps
Cycles:
1
Auto Reverse
Over Time
Set
Cancel
Help
Select Colors to Cycle
Cycle Length: 0
5 Frames Selected


Set Palette
Color Cycling
Palette Transition
Palette:
System - Win
Speed:
30
fps
Normal
Fade To Black
Fade To White
Over Time
Set
Cancel
Help
1 Frames Selected

Set Transition command

To set a transition, select a cell in the transition channel of the score window, and choose Set Transition. The transition occurs when the playback head reaches that cell.

Click here to see an illustration.

Tip: Double-clicking a cell in the transition channel opens the Set Transition dialog box.

Transition applies to:

Stage Area--The transition takes place over the entire stage.

Changing Area--The transition takes place over the changing area of the frame.

Duration: Indicates the approximate amount of time (in quarters of a second) of the entire transition.

Chunk Size: Indicates how much of the screen changes in each movement of the transition. The smaller the chunk size, the smoother the transition.

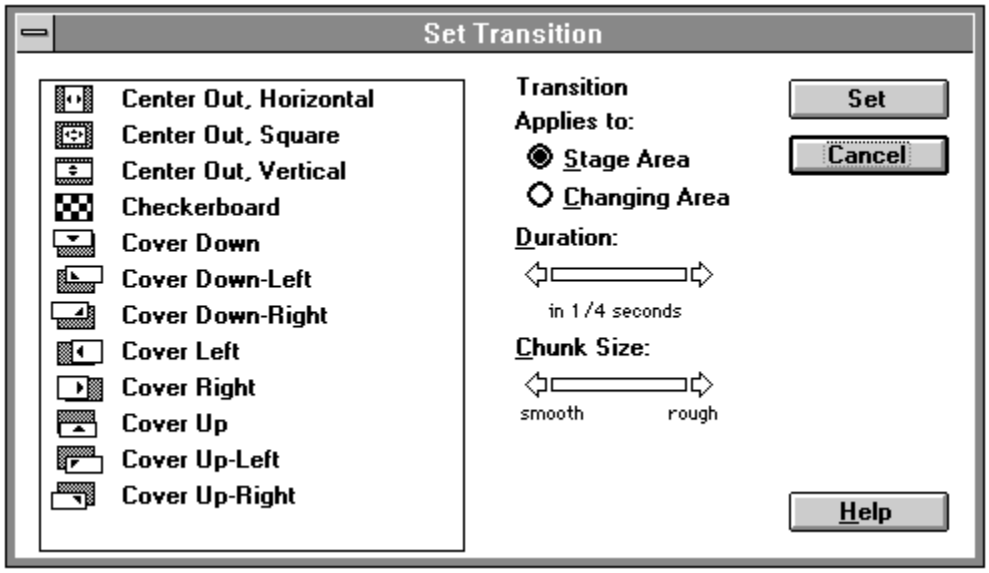
Notes:

To stop a transition during playback and advance the playback head to the next frame, press Control-period or Esc.

To play a sound while a transition takes place, place the sound in the frame immediately before the transition.

If you are developing a movie for Macintosh and Windows computers, don't use the Dissolve Pixels, Dissolve Pixels Fast, or Dissolve Patterns transitions. These transitions produce entirely different results on a Windows computer.

Palette transitions (including fade to black and fade to white) and dissolves don't work in 16- and 24-bit environments. That's because palette transitions can happen only when Director is using palettes and when you've set Windows display driver to 256 colors.


Set Transition
Center Out, Horizontal
Center Out, Square
Center Out, Vertical
Checkerboard
Cover Down
Cover Down-Left
Cover Down-Right
Cover Left
Cover Right
Cover Up
Cover Up-Left
Cover Up-Right
Transition
Applies to:
Stage Area
Changing Area
Duration:
in 1/4 seconds
Chunk Size:
smooth
rough
Set
Cancel
Help

Set Sound command

Lets you paste a sound into one of the sound channels in the score. Select one or more cells in one of the sound channels, and then choose this command. If the cast contains no sounds, or if you have not first made a selection in one of the sound channels in the score, this command is dimmed.

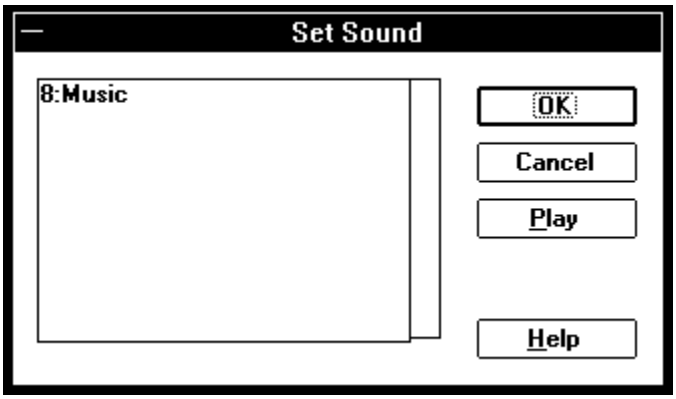
Click here to see an illustration.

A sound you set in sound channel 2 overrides a sound in sound channel 1.

A sound that's already playing in either sound channel overrides the sound in a digital video movie, and it also prevents the digital video movie's sound from playing even after the sound in the sound channel has stopped. Once the sound in a digital video movie has started, however, it overrides a sound in either sound channel.

When you choose Set Sound, the dialog box that appears lists all sounds stored as cast members in the cast.

Choosing a sound from the list pastes the sound into the selected frames in the sound channel in the score. The play button lets you preview the sound.


Set Sound
8:Music
OK
Cancel
Play
Help

Insert Frame command

Insert Frame (Control-J) adds a frame to your movie at the location of the playback head. With Insert Frame a copy of the current frame is added to the score. This can cause an apparent pause in the animation as two identical frames are played.

The new frame is inserted for all channels, and adds a frame to the movie's length.

Delete Frame command

Delete Frame (Control-[) deletes the frame at the location of the playback head, making the movie one frame shorter in length.

In-Between Linear command

In-Between Linear (Control-B) is used when you want a sprite to move in a straight line across the stage, grow or shrink smoothly, or remain stationary for a number of frames. Director fills in the selected cells in the score with the sprite's incremental motion between frames.

In-Between Linear interpolates the incremental positions of a sprite between key frames. Key frames are the locations of the sprite that are the starting and ending point for the sprite's motion. If the starting and ending points are the same, the sprite remains stationary. If the starting and ending points are different, In-Between Linear generates all the intermediate steps between key frames. In-Between Linear also works with sprites that are stretched or squeezed.

In-Between Special command

Use In-Between Special (Control-Shift-B) when you want a sprite to move in a curved path, to accelerate or decelerate across the stage, or to in-between a film loop.

Click here to see an illustration.

Click for more information:

- In-Between
- Acceleration
- Path

To make your sprite follow a curved path, you must set the sprite in at least three positions on the stage in different frames. As with the regular In-Between, select all the cells between the first and last position of the sprite, then choose this command.

In-Between (In Between Special)

You can independently in-between individual parameters such as Location, Size, Foreground Color, Background Color, and Blend. Check the parameters that you want to in-between.

Acceleration (In Between Special)

Ease-In and Ease-Out options can add realism to your animations by having sprites gradually build up to full speed, or decrease in speed as they come to a halt.

Accelerate over first n Frames (Ease-In)--Choose the number of frames over which you want to accelerate the sprite. If you choose to accelerate a sprite over the first eight frames, for example, a stationary sprite will ramp up to full speed over that many frames. To enter a number other than the choices in the drop-down list, choose Other.

Decelerate over last n Frames (Ease-Out)--Choose the number of frames over which you want to decelerate the sprite. For example, decelerating a sprite over eight frames will gradually bring the sprite to a halt by the eighth frame. To enter a number other than the choices in the drop-down list, choose Other.

Path (In Between Special)

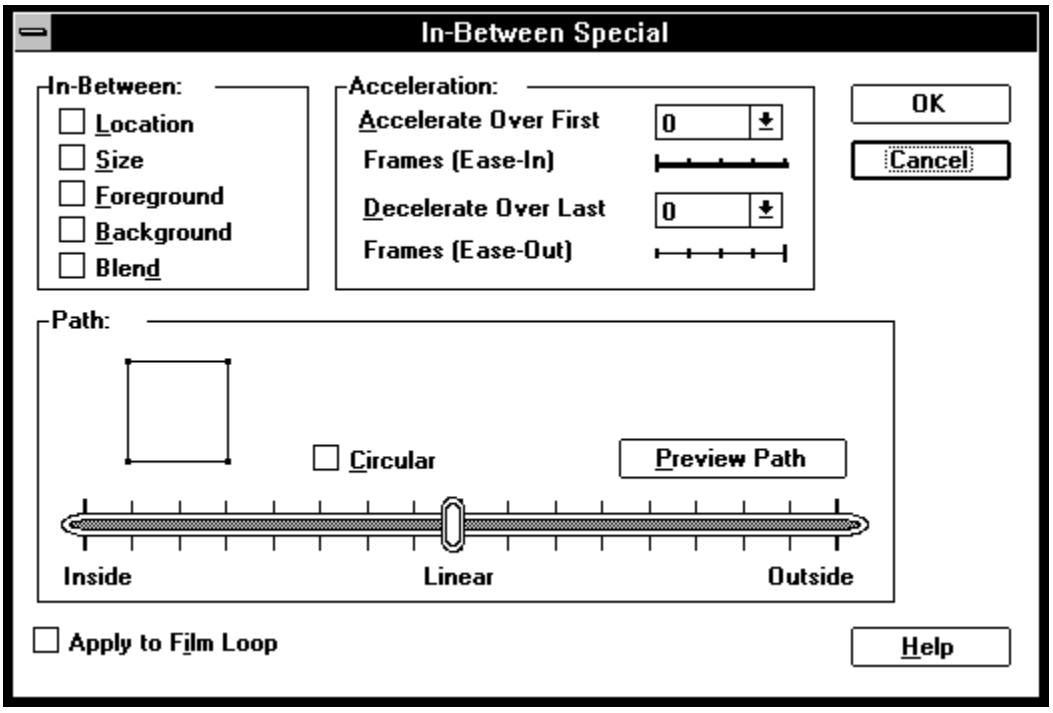
If the beginning and ending points of the sprite are the same, the diagram in the dialog box will be circular, indicating that the sprite will travel in a circle when in-betweened. If the beginning and end points are not the same, the diagram describes a curved path, indicating that the sprite ends in a position different than the starting point. This diagram does not show the actual path of the sprite, just the type of curve it will follow.

The Inside/Outside slider controls the degree to which the sprite's curved path follows the inside or outside boundaries of the path. If you drag the slider to the left, the sprite follows a curved path that is inside the sprite positions you set before choosing In-Between Special. If you move the slider to the center, the sprite will travel in straight lines between the points you set earlier. Dragging the slider all the way to the right causes the sprite to pass through the points you set as it travels a curved path.

Circular--This option only affects sprites that begin and end in the same point. If checked, the sprite will circle around the stage, but it won't pass through the starting point. The effect is to make a rounder circle. If this option is not checked, the sprite passes through the starting point during the animation.

Preview Path--Lets you see the results of your choices by closing the In Between Special dialog box and drawing a line on the stage indicating the path of the in-between. Click Preview Path to make sure your sprite is going where it's supposed to.

Apply to Film Loop--If checked, you can in-between a series of sprites. For example, if you have a sequence of six sprites that make a walking person, you can store all six as a single sprite, called a film loop. Then, the film loop can be in-betweened as if it were a single sprite. Be sure that a film loop is selected in the cast window before choosing In-Between Special.


In-Between Special
In-Between:
Location
Size
Foreground
Background
Blend
Acceleration:
Accelerate Over First
Frames (Ease-In)
0
Decelerate Over Last
Frames (Ease-Out)
0
OK
Cancel
Path:
Circular
Preview Path
Inside
Linear
Outside
Apply to Film Loop
Help

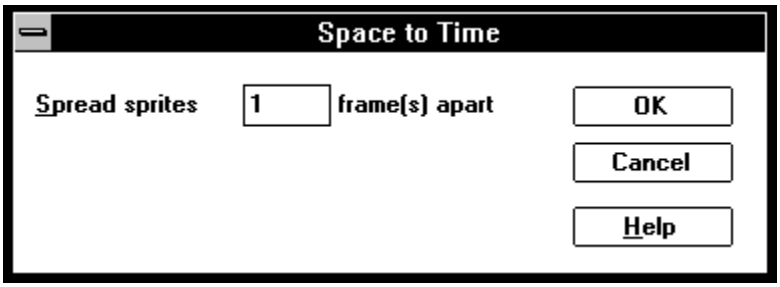
Space to Time command

Space to Time (Control-Shift-V) moves selected sprites in one frame to a single channel in the score so they play in a sequence of frames.

The dialog box lets you specify the number of frames apart to spread sprites. Consecutive cells (1 frame apart) is the default.

An example might be a bouncing ball. It's difficult to lay out each position of the ball without comparing it to previous positions. With Space to Time, you can drag the ball from the cast window repeatedly to lay out your sequence, select the cells in the score that contain the sprites you just positioned on the stage, then choose Space to Time, and all the sprites shift from their vertical positions in one frame to horizontal positions in one channel. You might also find this a useful way to lay out sprites before using In-Between Special.

Click here to see an illustration.


Space to Time
Spread sprites
1
frame(s) apart
OK
Cancel
Help

Paste Relative command

Use Paste Relative (Control-Shift-V) to paste a sequence of sprites at the point on the stage where a previous sequence ended. When you use this command, Director automatically adjusts the positions of cast members on the stage so that the first cast member in the pasted sequence follows the last cast member in the original sequence

For example, to animate a ball bouncing across the stage with a sequence of five sprites that describe one bounce, you can use Paste Relative to make the second sequence of sprites start where the first sequence ended. The effect is that the two sequences are chained, one after another, in one smooth, continuous motion. This works for any repetitive sequence of sprites.

When selecting cells in a sequence to be pasted relative to the original sequence, make sure the same cast member is at the beginning and end of the sequence, and that you overlap the first cell in the copy with the last cell in the original sequence.

Reverse Sequence command

Reverses the order of selected cells in the score.

Switch Cast Members command

The Switch Cast Members command (Control-E) replaces the cast member selected on the stage or in the score window with the cast member selected in the cast window. When you use Switch Cast Members, the registration point of the new cast member lines up with the registration point of the old cast member.

Auto Animate submenu

The Auto Animate commands permits you to quickly create titles, bar charts, and text effects to add to your movies. Director adds the auto animate notation to the score window, so you can edit it.

Click a type of Auto Animate command for specific information:

Auto Animate Banner
Auto Animate Bar Chart
Auto Animate Bullet Chart
Auto Animate Credits
Auto Animate Text Effects
Auto Animate Zoom Text

If you select a single cell in the score, the entire auto animate sequence will be added to the score beginning at the selected cell. Any existing score data will be replaced by the sequence. If you set an insertion point, the auto animate sequence is inserted in channel 1, beginning at the insertion point. If you select a range of cells, the auto animate sequence that is inserted will occupy the number of cells in the selection. If you do not make a selection or set an insertion point, the auto animate sequence is placed at the end of the movie beginning in channel 1.

Note: For auto animate sequences that use more than one channel, if there isn't enough room for all the channels to be placed in the score, they won't be recorded in the movie.

Auto Animate Banner command

Banner allows you to create an animated message that scrolls across the screen.

Click here to see an illustration.

Click for more information:

- Text Style
- Speed
- Initial Delay
- Repeat
- Preview

You can control the number of times the banner is displayed, the length of an initial delay, and the speed at which the effect occurs.

You can enter up to 255 characters in the text area. If you enter more text than Director can display in the text area at one time, the text scrolls to the left as you type.

Text Style (Auto Animate Banner)

Gives you complete control over the text font, size, style, transparency, and color of the banner text.

Speed (Auto Animate Banner)

Controls how fast the animation occurs, in this case how quickly the text moves across the screen.

Initial Delay (Auto Animate Banner)

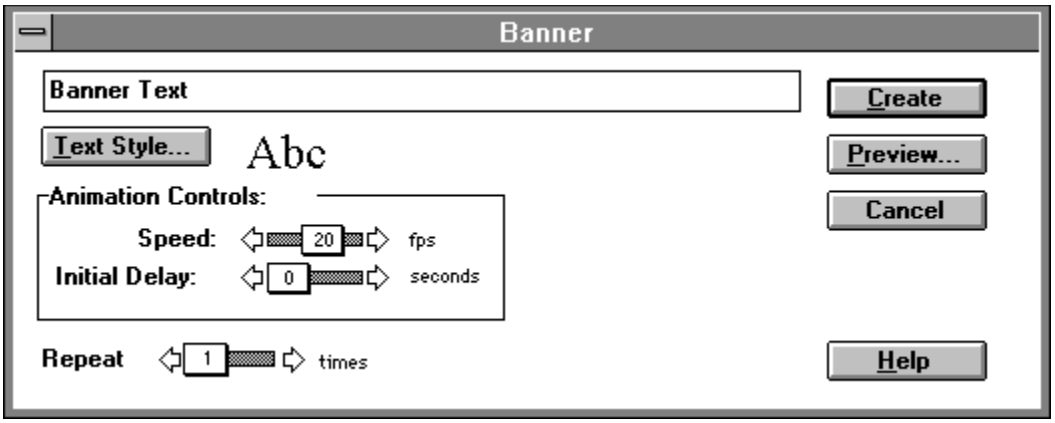
Controls the length of time before the effect starts.

Repeat (Auto Animate Banner)

Controls how many times the effect is repeated.

Preview (Auto Animate Banner)

Displays the banner as it will appear in the presentation, including any background images. Use the Preview screen to adjust the location of the text. To set the location, click the mouse. Setting the starting location sets the position of the top of the text. Note that the starting location affects only the vertical dimension; wherever the starting location, the banner text travels from off the right side of the screen to off the left side of the screen.


Banner
Banner Text
Create
Text Style...
Abc
Preview...
Animation Controls:
Cancel
Speed:
20
fps
Initial Delay:
0
seconds
Repeat
1
times
Help

Auto Animate Bar Chart command

Bar Chart allows you to create an animated bar chart.

Click here to see an illustration.

Click for more information:

- Chart Style
- Speed
- Preview

The title and labels of the bar chart appear first and are stationary; the bars grow into place through animation. Director allows you to control the scale of the chart, the height of the bars, the style of the bars, the speed of their growth, and the ending delay.

You can use up to six bars. The width of the bars is set automatically depending on the number of bars you are using. The actual length of the horizontal and vertical axes does not change, just the values on the scale. The height of the bars is proportional to the scale you set.

You can enter up to 255 characters in the Title text area and in the Vertical label text area. If you enter more text than Director can display in the text area at one time, the text scrolls to the left as you type.

Chart Style (Auto Animate Bar Chart)

Title Style, Label Style--These options give you complete control over the text font, size, style, transparency, and color of the title text and the label text.

Bar Style--Provides several choices for different types of bars.

Bar Labels--Enter a label that appears below each bar. You can enter up to 31 characters for each label. If the labels are so long as to overlap, you can either use shorter labels or adjust the font, size, and style of the labels.

Value--Determines the height of each bar, in relation to the scale of the vertical axis. You can enter any positive values between 0 and 32,000.

Range--Determine the scale shown on the bar chart. The scale affects the height of the bars, which grow to a height proportional to the scale. You can set the maximum and minimum to any positive values between 0 and 32,000.

Speed (Auto Animate Bar Chart)

Controls how quickly the bars grow into place.

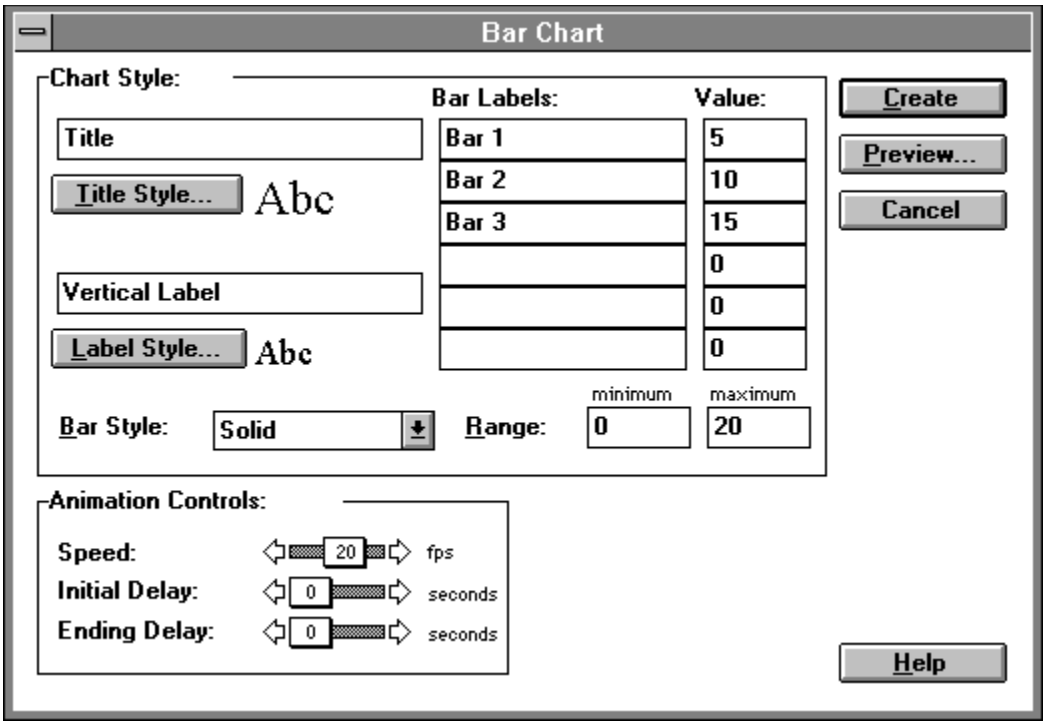
Speed--Controls how quickly the bars grow into place.

Initial Delay--Controls the length of time before the effect starts.

Ending Delay--Controls how long the completed bar chart is displayed.

Preview (Auto Animate Bar Chart)

Displays the bar chart as it will appear in the presentation, including any background images. Use the Preview screen to adjust the location of the bar chart. To set the location, click the mouse. Setting the starting location sets the position of the upper left corner of the chart. The upper left corner is determined by the top of the title text and the left edge of the vertical label text.


Bar Chart
Chart Style:
Bar Labels:
Value:
Create
Title
Bar 1
5
Preview...
Title Style...
Abc
Bar 2
10
Bar 3
15
Cancel
Vertical Label
0
0
Label Style...
Abc
0
minimum
maximum
Bar Style:
Solid
Range:
0
20
Animation Controls:
Speed:
20
fps
Initial Delay:
0
seconds
Ending Delay:
0
seconds
Help

Auto Animate Bullet Chart command

Bullet Chart allows you to create an animated bulleted list.

Click here to see an illustration.

Click for more information:

- Text Formats
- Animation Controls
- Preview

Typically, the title of the bulleted list appears first and is stationary; the items and their bullets travel into place through animation.

You can determine the direction from which the bulleted items come. You can choose from several different bullets. You can animate the title as well as the bullets. You can have no animation at all, simply presenting a bulleted list. Or you can dispense with bullets altogether, presenting only a title and a list.

You can enter up to 255 characters in the Title text area and in each Bullet text area. If you enter more text than Director can display in the text area at one time, the text scrolls to the left as you type.

Text Formats (Auto Animate Bullet Chart)

Bullet type--Provides choices for different types of bullets. Choices include a Dot, a Square, a Hand, a Check, and an Arrow. Three choices allow for animated bullets that move into place. These are the Flying Hand, the Animated Check, and the Flying arrow. You can also choose to have no bullets, or to have no bullets and the text and title centered.

Line Spacing--Controls the amount of space between items on the chart.

Bullet Style, Title Style--Give you complete control over the text font, size, style, transparency, and color of the title text and the bullet text.

Animation Controls (Auto Animate Bullet Chart)

Motion--Provides choices for how the bulleted items appear. The choices include from Right, Bottom, Left, Top, Upper Right, Lower Right, and so on. Choices also include Stationary (no movement), Wipe from Right, Wipe from Left, and Dissolve.

Speed--Controls how fast the animation occurs, in this case how quickly the bulleted items move into place.

Initial Delay--Controls the length of time before the effect starts.

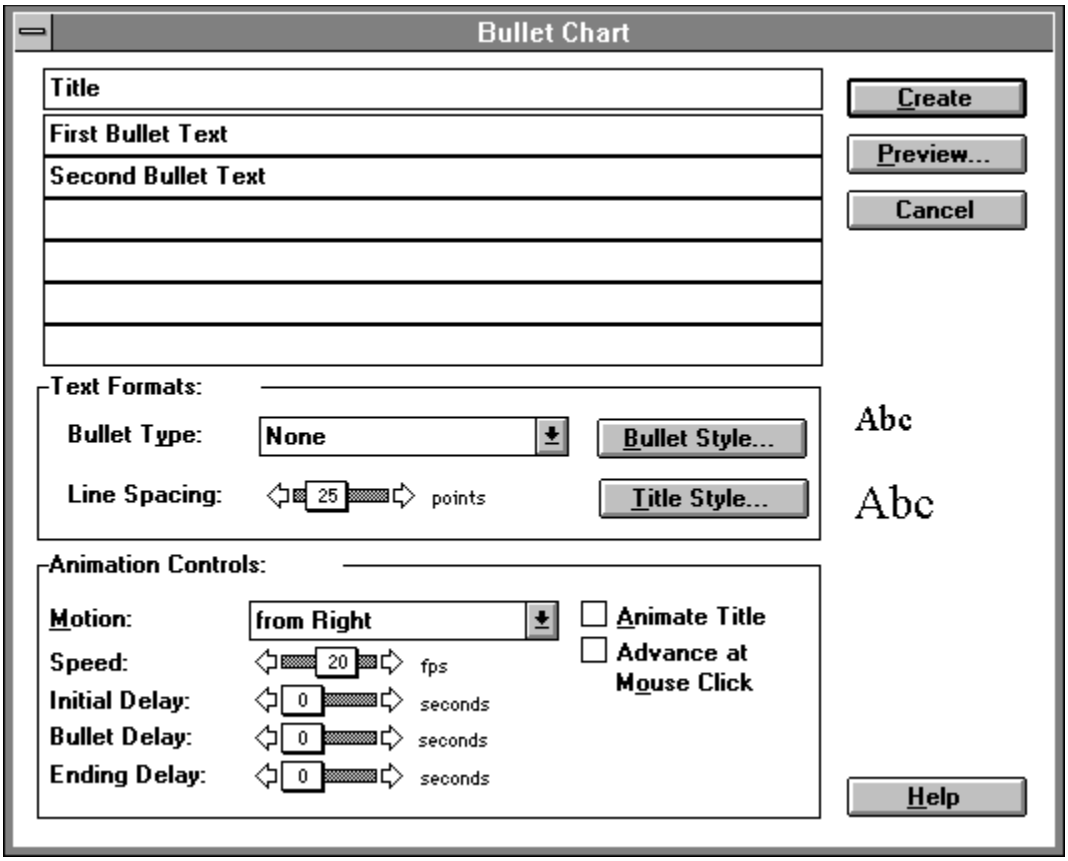
Bullet Delay--Controls the length of the pause between lines of text moving into place. A short Bullet Delay means that there is a very short pause between one line moving into place and the beginning of the next line's movement.

Ending Delay--Controls how long the completed chart is displayed.

Animate Title--Causes the title text to also move into place. An animated title obeys the Motion, Speed, and Bullet delay settings.

Preview (Auto Animate Bullet Chart)

Displays the bullet chart as it will appear in the presentation, including any background images. Use the Preview screen to adjust the location of the bullet chart. To set the location, you click with the mouse. Setting the starting location sets the position of the upper left corner of the title.


Bullet Chart
Title
First Bullet Text
Second Bullet Text
Create
Preview...
Cancel
Text Formats:
Bullet Type:
None
Bullet Style...
Abc
Line Spacing:
25
points
Title Style...
Abc
Animation Controls:
Motion:
from Right
Animate Title
Advance at Mouse Click
Speed:
20
fps
Initial Delay:
0
seconds
Bullet Delay:
0
seconds
Ending Delay:
0
seconds
Help

Auto Animate Credits command

Credits allows you to create an animated credit list that scrolls up the screen. "Credits" comes from the motion picture practice of giving credit to everyone who worked on a movie in a scrolling list at the end.

Click here to see an illustration.

Click for more information:

- Text Style
- Justification
- Animation Controls
- Repeat
- Preview

You can control the number of times the credit list is repeated, the speed at which the effect occurs, the length of an initial delay, and the justification of the text (left, right, or center).

Enter text in the text area, using the Return key to end a line of text and start a new line. There is no limit to the width of an individual line, except the practical limit of the screen width. Text that is too long to fit on one line of this dialog box wraps to the next line. Text appears in the preview as it will in the final slide.

You can enter up to 255 characters in the credit list.

Text Style (Auto Animate Credits)

Gives you control over the text font, size, style, transparency, and color of the credit text.

Justification (Auto Animate Credits)

Controls whether the credit text is justified flush left, flush right, or is centered.

Animation Controls (Auto Animate Credits)

Speed--Controls how quickly the text moves up the screen.

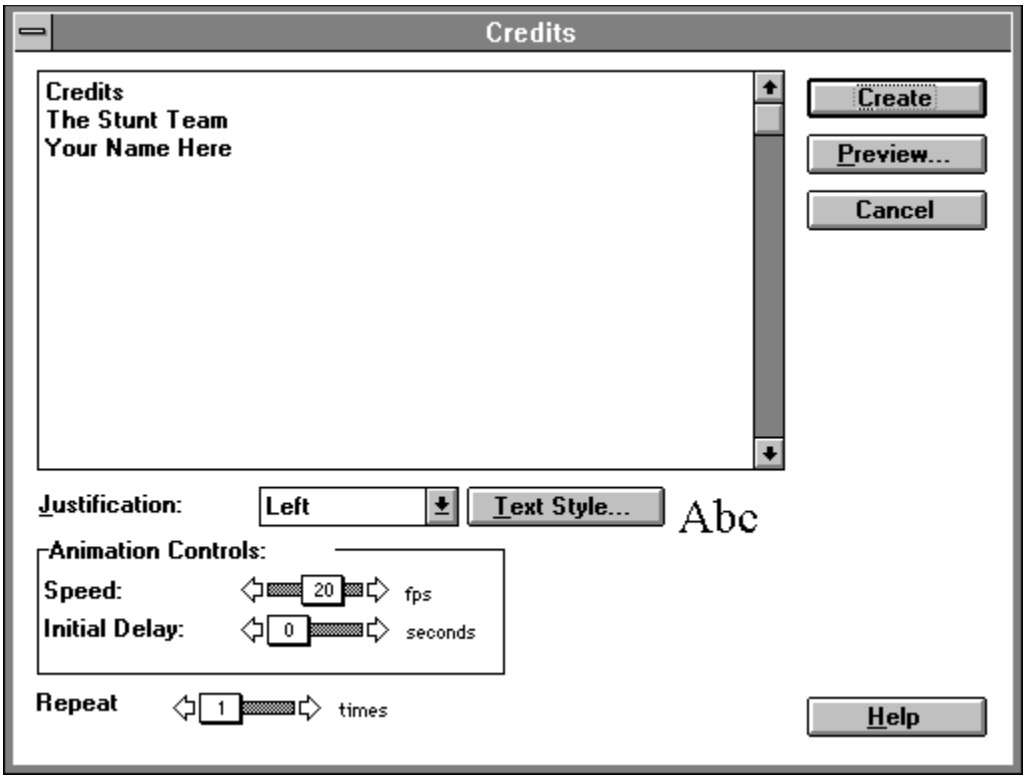
Initial Delay--Controls the length of time before the effect starts.

Repeat (Auto Animate Credits)

Controls how many times the effect is repeated.

Preview (Auto Animate Credits)

Displays the credit list as it will appear in the presentation, including any background images. Use the Preview screen to adjust the location of the text. To set the location, you click with the mouse. Setting the starting location sets the position of the center of the text. Note that the starting location affects only the horizontal dimension; wherever the starting location, the credit list travels from below the bottom of the screen to off the top of the screen.


Credits
Credits
The Stunt Team
Your Name Here
Create
Preview...
Cancel
Justification:
Left
Text Style...
Abc
Animation Controls:
Speed:
20
fps
Initial Delay:
0
seconds
Repeat
1
times
Help

Auto Animate Text Effects command

Text Effects allows you to create one of three animated effects on text. The effects include Sparkle, in which an animated highlight appears and disappears; Letter slide, in which the text is built up as letters slide one at a time; and Typewriter, in which the letters appear one at a time, as if typed.

Click here to see an illustration.

Click for more information:

- Text Style
- Animation Controls
- Repeat
- Preview

You can control the number of times the effect is repeated, the speed at which the effect occurs, the length of an initial delay, and the duration of the pause after the end of the effect.

You can enter up to 255 characters in the text area. If you enter more text than Director can display in the text area at one time, the text scrolls to the left as you type.

Text Style (Auto Animate Text Effects)

Gives you complete control over the text font, size, style, transparency, and color of the text.

Animation Controls (Auto Animate Text Effects)

Effect--Provides three choices for the effect you want. Choices include Sparkle, Letter Slide, and Typewriter.

Speed--Controls how quickly the text moves up the screen.

Initial Delay--Controls the length of time before the effect starts.

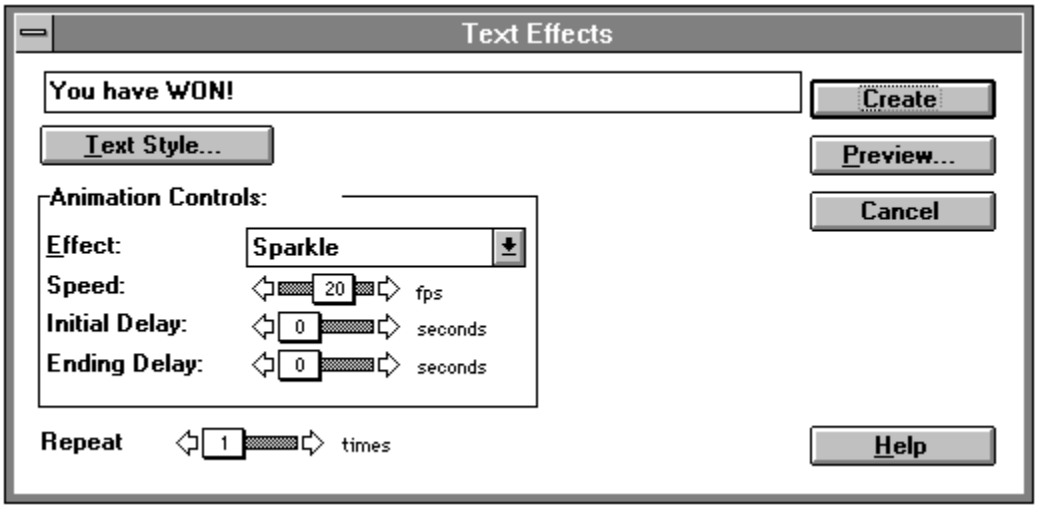
Ending Delay--Controls how long the text is displayed after the effect is finished.

Repeat (Auto Animate Text Effects)

Controls how many times the effect is repeated.

Preview (Auto Animate Text Effects)

Displays the text effect as it will appear in the presentation, including any background images. Use the Preview screen to adjust the location of the text. To set the location, you click with the mouse. Setting the starting location sets the position of the center of the text.


Text Effects
You have WON!
Create
Text Style...
Preview...
Cancel
Animation Controls:
Effect:
Sparkle
Speed:
20
fps
Initial Delay:
0
seconds
Ending Delay:
0
seconds
Repeat
1
times
Help

Auto Animate Zoom Text command

Zoom Text allows you to create an animated zoom effect. "Zoom" comes from the motion picture camera technique of changing smoothly from a long shot to a closeup (zoom in), or from a closeup to a long shot (zoom out).

Click here to see an illustration.

Click for more information:

- Text Style
- Animation Controls
- Repeat
- Preview

You can choose whether the text you enter grows from nothing to full size (zoom in), from full size to nothing (zoom out), or first zooms in then zooms out. The text grows and diminishes through animation. Director allows you to control the number of times the effect is repeated, the speed at which the effect occurs, the length of an initial delay, and how long the full-size text remains on the screen.

You can enter up to 75 characters in the text area. If you enter more text than Director can display in the text area at one time, the text scrolls to the left as you type.

Text Style (Auto Animate Zoom Text)

Gives you complete control over the text font, size, style, transparency, and color of the text.

Animation Controls (Auto Animate Zoom Text)

Zoom Type--Determines whether the text zooms in, zooms out, or first zooms in and then zooms out.

Speed--Controls how fast the animation occurs, in this case how quickly the text grows from nothing into full size or diminishes from full size to nothing.

Initial Delay--Controls the length of time before the effect starts.

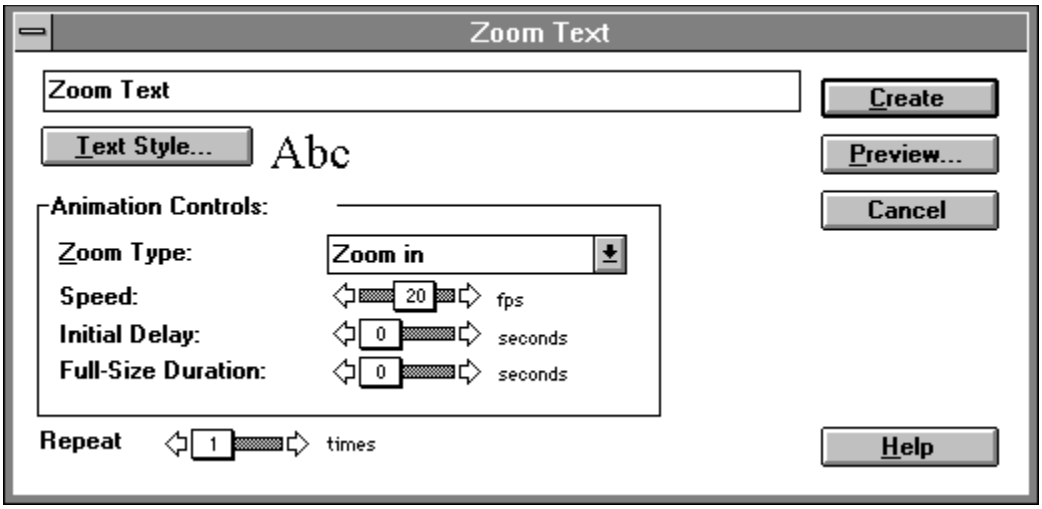
Full-Size Duration--Controls how long the full-size text is displayed. Note that the text is full-sized at the beginning of a zoom out effect, in the middle of a zoom in and out effect, and at the end of a zoom in effect.

Repeat (Auto Animate Zoom Text)

Controls how many times the effect is repeated.

Preview (Auto Animate Zoom Text)

Displays the zoom as it will appear in the presentation, including any background images. Use the Preview screen to adjust the location of the text. To set the location, you click with the mouse. Setting the starting location sets the position of the middle of the text.


Zoom Text
Zoom Text
Create
Text Style...
Abc
Preview...
Cancel
Animation Controls:
Zoom Type:
Zoom in
Speed:
20
fps
Initial Delay:
0
seconds
Full-Size Duration:
0
seconds
Repeat
1
times
Help

Score Window Options command

This command controls display options in the score window.

Click here to see an illustration.

Click for more information:

- Display Options
- Extended Display Information

Display Options (Score Window Options)

Colored Cells--If checked, you can choose a color for selected cells using the cell color selector on the left side of the score window. Otherwise, the cell color selector is hidden.

If you've already applied color to cells, unchecking this option hides cell colors but doesn't remove them. Score window scrolling performance is faster if you hide cell colors.

Magnified Cells--If checked, this option enlarges the cells in the score window for better viewing. You can perform all the regular score operations while the view is magnified.

Playback Head Follows Selection--This option toggles between two ways of selecting frames in the score. If checked, the playback head travels as you make a selection in the score window. You can see the selected frames on the stage as you select them. If not checked, your selection in the score window has no effect on the playback head and does not advance the movie as you make a selection. You might want to turn this option off if the score includes frames that take a long time to draw, such as frames with blend ink and anti-aliasing applied to them. (In earlier versions of Director, this setting was called Easy Select.)

You can temporarily switch this option to its opposite setting by pressing the Alt key while making a selection in the score window.

Drag and Drop--When Drag and Drop is selected, you can select information in one or more cells in the score, drag the information to a different part of the score, and "drop" the information in the cells at the new location. When Drag and Drop isn't selected, you can move information from one part of the score to another only by cutting it from its current location and pasting it in a new location.

When you're working in the score, you can temporarily reverse the current Drag and Drop setting by holding down the Spacebar.

Extended Display Information (Score Window Options)

The Extended Display options let you choose the notation information that appears in the numbered sprite channels in the score if you choose the Extended display command from the score window's Display pop-up menu.

Cast Type, Movement, Anti-Alias, Blend displays the cast member type (text, PICT, or bitmap) and a directional arrow to indicate the cast member's position with respect to the previous cast member in that channel. It also indicates whether a cast member has an anti-alias setting or a blend percentage applied to it.

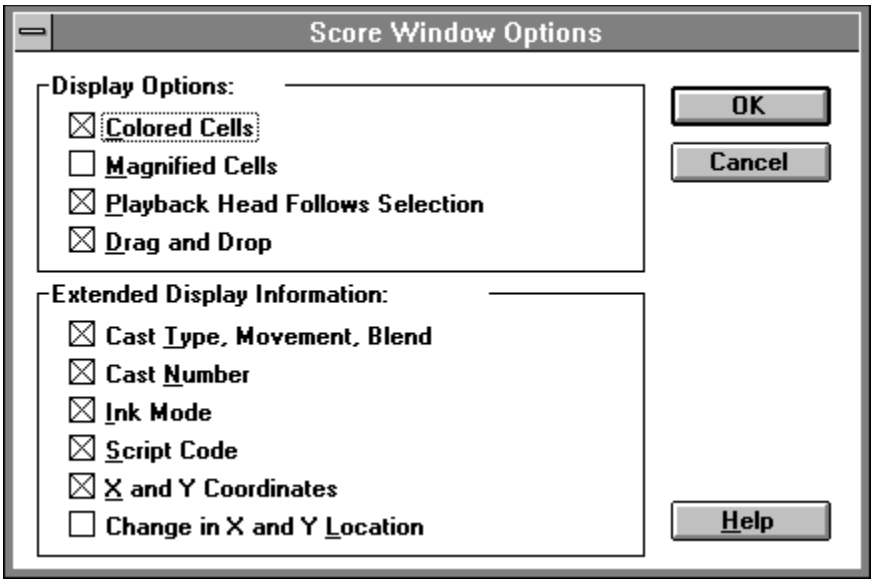
Cast Number Cast Number displays the cast member position number from the cast window. If you've selected Name as the Cast ID Style in the Cast Window Options dialog box, then the first few letters of the cast member name are displayed instead.

Ink Mode displays the type of ink applied to the cast member in that cell.

Script Code displays the number of the script associated with that cell or a plus (+) sign if the script is a cast member script.

X & Y Coordinates show the screen coordinates of the cast member.

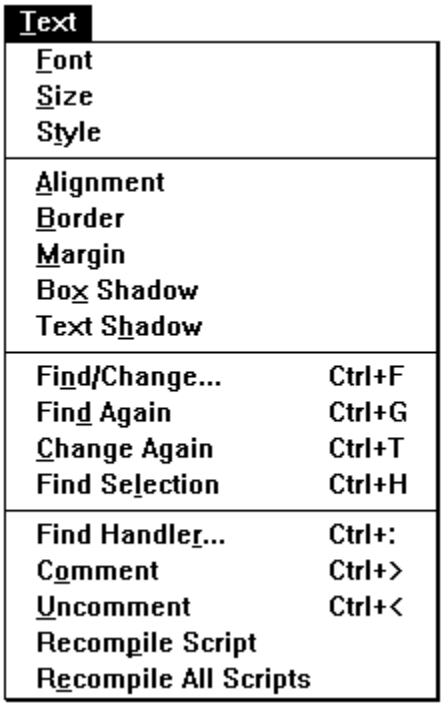
Change in X and Y Location indicates the change in X and Y coordinates relative to the previous cast member in that channel.


Score Window Options
Display Options:
Colored Cells
Magnified Cells
Playback Head Follows Selection
Drag and Drop
Extended Display Information:
Cast Type, Movement, Blend
Cast Number
Ink Mode
Script Code
X and Y Coordinates
Change in X and Y Location
OK
Cancel
Help

Text menu

The Text menu controls the font, size, style, and alignment of text in the text, paint, markers, and cast windows, and on the stage.

Click a menu command for more information:



To apply a font, size, or style to existing text in the text window, select the text before choosing a Font, Size, or Style command. If there is no selection in the text window, the font, size, or style you choose will be applied to future text entered in a text window.

In a text window, the Font, Size, or Style settings you choose can be applied on a per-character basis. The Alignment, Border, Margin, Box Shadow, or Text Shadow settings you choose are applied to the window's entire contents.

In the paint window, you can only choose one font, size, style, and text shadow for the text you are typing. (The Alignment, Border, Margin, and Box Shadow commands are not available in the paint window.) You can choose them before typing, or after typing but before clicking another tool or repositioning the insertion point. If you want to change the bitmapped text in the paint window after clicking the mouse button, you must erase the previous type and create new type in the style you want.

In the markers window, you can only choose one font, size, and style for all text in the window.

In the cast window, the Font, Size, Style, Alignment, Border, Margin, Box Shadow, and Text Shadow commands are available if one or more text cast members are selected. You can use these commands to change the text attributes of all selected text cast members at once.

Note: If you format text, delete it, and then type new text, the new text appears in the default format. Deleting all the text in a text field on the stage or in the text window deletes the formatting.

Font submenu

Lets you choose any font that is installed in your System.

Size submenu

Controls the font size, in points. Point sizes that appear in outlined numbers are specifically designed for the currently selected font and will look better than other font sizes.

Style submenu

Lets you choose a style for text.

Alignment submenu

Controls the alignment of text between the left and right borders of the text area. You can change the position of the right margin by dragging the right margin handle in a text window, or by dragging the handle of selected text on the stage.

Border submenu

Adds a box around a text object on the stage.

Margin submenu

Changes the distance between edges of the border and the text inside the border. Margins can only be used with text objects on the stage.

Box Shadow submenu

Adds a drop shadow to the text box. Drop shadows can only be used with text objects on the stage.

Text Shadow submenu

Adds a drop shadow to bitmapped text or a text object. Drop shadow on text is a good way to ensure that your text will remain legible in color if you are planning to overlay text to videotape.

Find/Change command

Find/Change (Control-F) lets you quickly search for and replace text in the text or script windows. All searches start at the insertion point and work forward. The Find/Change dialog box lets you enter the text you want to find in the text or script window and the text you want to replace it with.

Click here to see an illustration.

Click for more information:

- Find field
- Change to field
- Whole Words
- Wrap-Around Search
- Search All Cast Members
- Find button
- Change button
- Change All
- Cancel

Director stores the settings you specify in the Director 4.0 Preferences file (DIRECTOR.PRF), but does not save the text you enter in the Find or Change to fields.

Searching begins at the location of the insertion point. If you want to search from the beginning of the window, make sure the cursor is at the upper-left corner of the window before you begin the search.

Find field (Find/Change)

Enter the text you want to find. Searching is not case-sensitive: ThisHandler, thisHandler, and THISHANDLER are all the same for search purposes.

Change to field (Find/Change)

Enter the replacement text.

Whole Words Only (Find/Change)

Only finds occurrences of the search text as a separate word.

Wrap-Around Search (Find/Change)

Specifies whether or not Director continues the search from the top of the window when it reaches the last line in the window. If this option is checked but Search All Cast Members is not checked, Director continues searching from the top of the current window after it reaches the bottom of the window. If both options are checked, Director searches all cast members of the same type (either text or script, depending on the window in which you initiated the search), beginning with the currently selected cast member, and wrapping around to the first cast member of that type if necessary.

Search All Cast Members (Find/Change)

Specifies whether Director searches just the current cast member, or searches all cast members of the same type (either text or script, depending on the window in which you initiated the search). If checked, Director searches all cast members of the same type, beginning with the current cast member, and wrapping around to the first cast member if Wrap-Around Search is checked. If Search All Cast Members is not checked, Director only searches the current cast member's text or script window.

Find button (Find/Change)

Clicking Find closes the dialog box and finds the first occurrence of the specified text. The scope of the search depends on whether or not Search All Cast Members is checked. If a match is found, Director selects the text. If no match is found, Director beeps.

Change button (Find/Change)

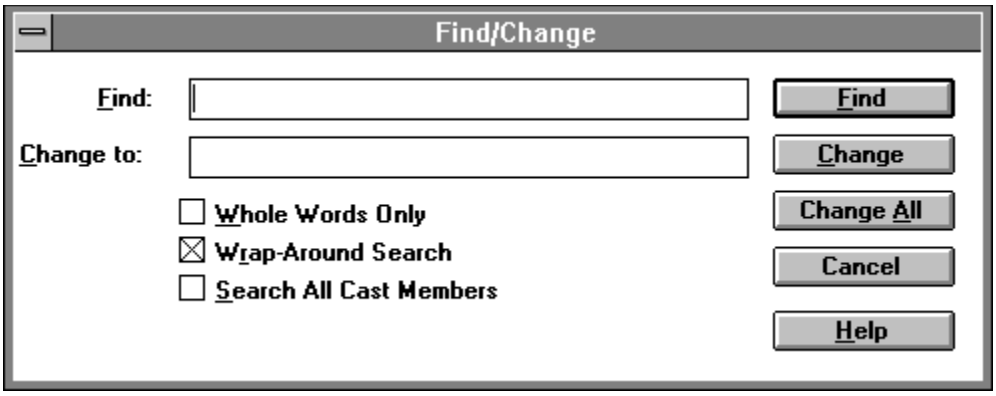
Clicking Change closes the dialog box, finds the next occurrence of the specified text, starting from the insertion point, and replaces it with the specified text. The scope of the search depends on whether or not Search All Cast Members is checked. If no matching text is found, Director beeps.

Change All (Find/Change)

Clicking Change All closes the dialog box and replaces all occurrences of the specified text. Director first displays an alert informing you that this operation cannot be undone. If Search All Cast Members is checked, Director replaces all occurrences of the specified text for all cast members of the same type; otherwise, Director limits the operation to the current text or script window.

Cancel (Find/Change)

Clicking Cancel closes the dialog box without performing a search or replace. However, any changes you made to the settings in the dialog box are saved in the Director 4.0 Preferences file (DIRECTOR.PRF).


Find/Change
Find:
Change to:
Whole Words Only
Wrap-Around Search
Search All Cast Members
Find
Change
Change All
Cancel
Help

Find Again command

Find Again (Control-G) finds the next occurrence of the text you entered in the Find: field in the Find/Change dialog box.

Change Again command

Change Again (Control-T) replaces the selected text and then searches for the next occurrence of the same text. You must choose this command again if you want to change the next occurrence of the selected text.

Find Selection command

Find Selection (Control-H) lets you find the next occurrence of the currently selected text in the text or script window.

Find Handler command

Find Handler (Control-:) lets you view the names of all handlers in the current script or movie. You can also use this command to open the script window that contains the selected handler.

Click here to see an illustration.

Find Handler--These buttons only appear if a script window is the active window. If any other window is active when you choose this command, all script handlers are listed.

- **Current Script**--Lists the handlers defined in the current script.

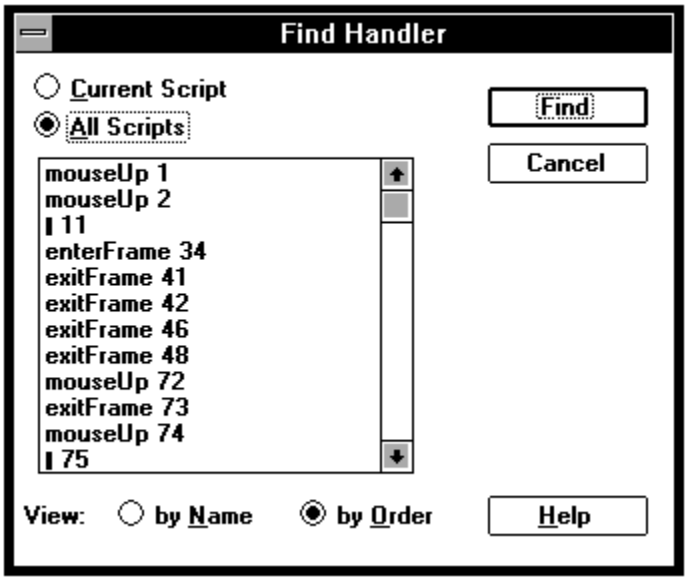
- **All Scripts**--Lists the handlers defined in all scripts.

View

- **By Name**--Lists handlers alphabetically, by name.

- **By Order**--lists handlers in the order in which they appear in their respective scripts. Handlers from different scripts are listed in the order in which the scripts appear in the cast window.

Selecting a handler and clicking OK opens the Script window in which the handler is defined.


Find Handler
Current Script
All Scripts
Find
Cancel
mouseUp 1
mouseUp 2
11
enterFrame 34
exitFrame 41
exitFrame 42
exitFrame 46
exitFrame 48
mouseUp 72
exitFrame 73
mouseUp 74
75
View:
by Name
by Order
Help

Comment command

Inserts the Lingo comment characters "--" at the beginning of the selected line(s).

Tip: Control-> (greater than) is the keyboard shortcut for this command.

Uncomment command

Removes the Lingo comment characters "--" from the beginning of the selected line(s).

Tip: Control-< (less than) is the keyboard shortcut for this command.

Recompile Script command

Director saves any unsaved changes in the active script window, and then compiles the window's script. This command is dimmed unless a script window is the active window.

Tip: Control-Shift-R is the keyboard equivalent for this command.

Recompile All Scripts command

Recompiles all Lingo scripts and checks them for errors. If a script error is found, the appropriate script window opens and the error is selected.

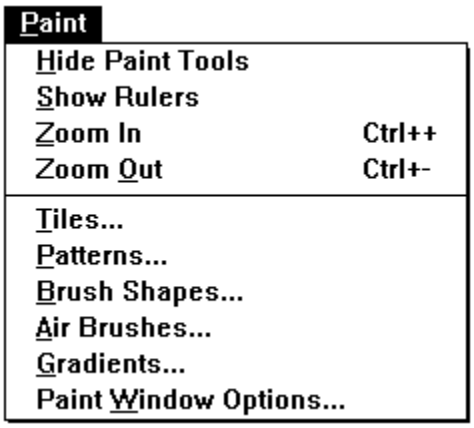
If a script window is the active window, Director first saves any unsaved changes in the current script window and compiles its script before continuing.

Tip: Control-Alt-R is the keyboard shortcut for this command.

Paint menu

The Paint menu appears when the paint window is open. It contains commands for editing artwork created or imported to the paint window.

Click a menu command for more information:



Hide/Show Paint Tools commands

The Hide Paint Tools command (Control-Shift-J) eliminates the tool palette from the paint window, leaving you with more room to create your cast members. After you choose Hide Paint Tools, this command changes to Show Paint Tools.

Hide/Show Rulers commands

The Show Rulers command (Control-Shift-K) displays vertical and horizontal rulers in the paint window. The default setting for the unit of measure is inches. You can change the scale to picas, centimeters, or pixels by clicking the upper left corner where the vertical and horizontal rulers meet. The zero point of the rulers can be moved to a new location by dragging from the corner of the rulers. The current position of the pointer is indicated by dotted lines in the rulers. After you choose Show Rulers, the command changes to Hide Rulers.

Zoom In command

Shows an enlargement of the artwork in the active easel. The enlargements can be 2, 4, and 8 times. Choose Zoom In from the Paint menu again to increase the magnification.

When you zoom in, a smaller representation of the 100% size artwork appears in the upper right corner of the paint window. To return to normal size, click inside the smaller window or choose Zoom Out.

Tip: The keyboard shortcut for this command is Control-+ (plus). You can also magnify your artwork by double-clicking the pencil tool in the paint window's tool palette, or by Control-clicking the area you want to enlarge with any of the paint tools. To move around in a zoomed view, press the Spacebar.

Zoom Out command

Reduces the zoom factor on the artwork if you are using an enlarged view (i.e., if you have chosen Zoom In).

Tip: Control-clicking in the paint window, or pressing Control-- (minus) is a shortcut for choosing Zoom Out.

Tiles command

Tiles are a useful way to create patterns with more than two colors. The basis for a tile is a cast member or one of the default tiles. When you choose a portion of a cast member to be made into a tile the cast member becomes a building block for a pattern created with a field of tiles. You can use tiles in the paint window or with the QuickDraw shapes in the tools window. Tiles can only be created on a color Macintosh.

Click here to see an illustration.

Click for more information:

- Select Tile to Edit
- Width, Height
- This Tile Is

- Creating tiles from a cast member

The Tiles dialog box lets you select the tile position, the cast member to make into a tile, what portion of the cast member to use for the tile, and the tile size.

The selection rectangle in the cast member box at the upper left of the Tiles dialog box determines which part of the cast member is used for the tile. Drag the rectangle to reposition it on a different part of the cast member or click a new spot in the cast member view to reposition the dotted rectangle.

Select Tile to Edit (Tiles dialog box)

Click the tile you want to edit. An enlarged version of the tile is displayed in the dialog box.

Width, Height (Tiles dialog box)

Lists the available tile sizes in pixels. You can make a tile as small as 16 by 16 pixels or as large as 128 by 128 pixels. As the size of the tile increases, more of the cast member is used for the tile.

This Tile Is (Tiles dialog box)

Controls which cast member is the basis for your tile. Click Built-In to indicate that the selected tile is one of the default tiles. Click the Cast member radio button if you want to make a tile from a cast member in your movie. When you click the Cast member radio button, left and right arrows appear that permit you to step through all the graphic cast members in your movie.

Creating tiles from a cast member

You can use tiles to make multi-colored patterns. There are eight default tiles that are always available for you to use, or you can create your own. The tiles you create are stored with the Director movie they were created in.

Once created, the tile appears in the patterns pop-up menu and can be used like any other pattern. The current foreground and background colors have no effect on the color of the tiles but switching palettes changes the color of the tiles.

To create a tile:

1. Choose Tiles from the Paint menu.

   The radio buttons determine whether the tile you select is built-in or made from a cast member.

2. Click Cast Member.

   This radio button is dimmed if all the cast members in the movie are 1-bit cast members. Tiles can only be made from cast members with color depths greater than 1-bit.

3. Click the left or right arrows to choose a cast member.

   As you click the arrows, the cast members in the paint window appear in the left side of the Tiles dialog box.

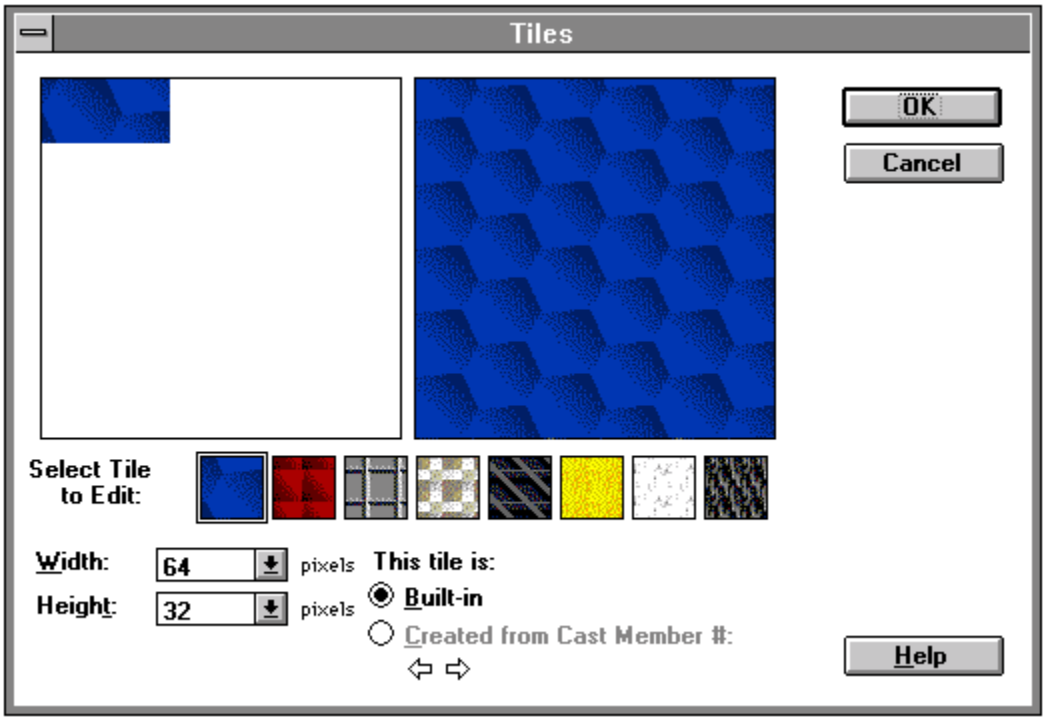
4. Choose a size using the Width and Height drop-down lists.

   As you choose a width and height for your tile, the dotted box in the left side of the Tiles dialog box changes shape to indicate the tile's size. You can also drag the dotted box to make a tile from a different part of the cast member.

5. Click OK.

   The tile you made is displayed in the bottom row of the pattern palette.

You can choose and use the tile just as you would select and use a pattern.


Tiles
OK
Cancel
Select Tile to Edit:
Width:
64
pixels
Height:
32
pixels
This tile is:
Built-in
Created from Cast Member #:
Help

Patterns command

Use the Patterns command to change the custom paint patterns.

Click here to see an illustration.

The patterns drop-down list permits you to switch between Custom, Grays, Standard, and QuickDraw patterns. Only the custom patterns are editable.

To copy a built-in pattern to the custom set of patterns:

1. Choose one of the built-in sets of patterns from the drop-down list.

2. Click Copy All.

3. Choose Custom from the pop-up menu.

4. Click Paste All.

   The set of built-in patterns is pasted to the Custom pattern menu. You can now edit any one of the individual patterns.

The currently selected pattern is displayed at the left side of the dialog box. If you want to edit a different pattern, click the pattern in the pattern palette at the right side of the dialog box. Once you select the pattern you want to edit, you can click the pixels in the magnified view. Clicking a white pixel turns it black, and vice versa. The buttons below the editing area give you additional pattern editing abilities.

You can "pick up" patterns from areas outside the Patterns dialog box by clicking outside the dialog box. The selected pattern changes to the pattern in the area you click.

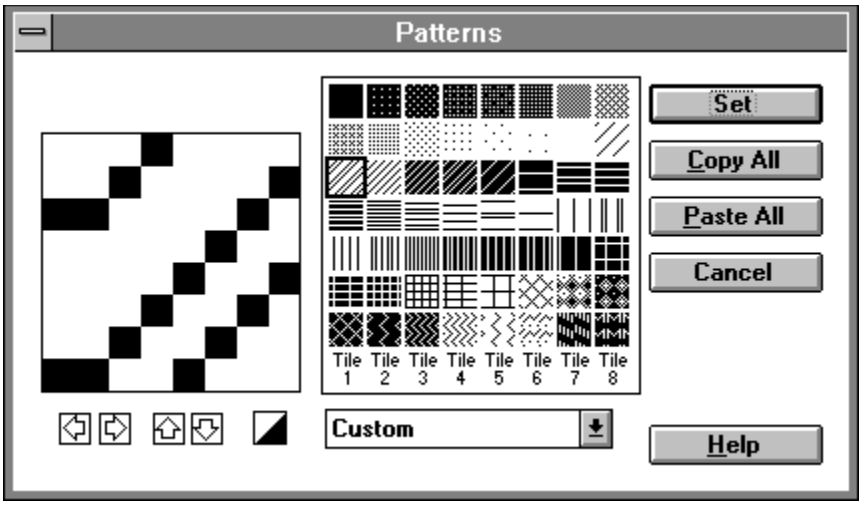
The **right and left arrows** bounce the pattern one pixel to the right or to the left.

The **up and down arrows** bounce the pattern up or down one pixel.

The **black and white square** inverts the colors of the pattern (e.g., black becomes white and white becomes black).

If you want to edit the Grays, Standard, or QuickDraw patterns, you can swap entire sets of patterns in the patterns drop-down list using the Copy All and Paste All buttons. When you have a set of custom patterns you like, click Copy All, and then click Cancel. Paste the patterns into the paint window. (Pasting a set of patterns into the paint window makes the set a cast member and stores it in the cast window.) Later, you can display the patterns in the paint window, select them with the selection rectangle, copy them, and then use the Paste All button to install them again as custom patterns.

Set--When you click Set the changes are reflected in the patterns pop-up palette in the tool and paint windows. Any custom patterns you create are stored in the Director 4.0 Preferences file (DIRECTOR.PRF).


Patterns
Tile 1
Tile 2
Tile 3
Tile 4
Tile 5
Tile 6
Tile 7
Tile 8
Custom
Set
Copy All
Paste All
Cancel
Help

Brush Shapes command

The Brush Shapes dialog box lets you change the shape of the paintbrush. You can choose Brush Shapes from the Paint menu, or double-click the paintbrush in the paint window's tool palette. The custom brushes you create are stored in the Director 4.0 Preferences file (DIRECTOR.PRF).

Click here to see an illustration.

The brush shapes pop-up menu in the dialog box permits you to choose from the standard default brush shapes, or create your own brush shapes from the set of custom brush shapes. Only the Custom brush shapes are editable.

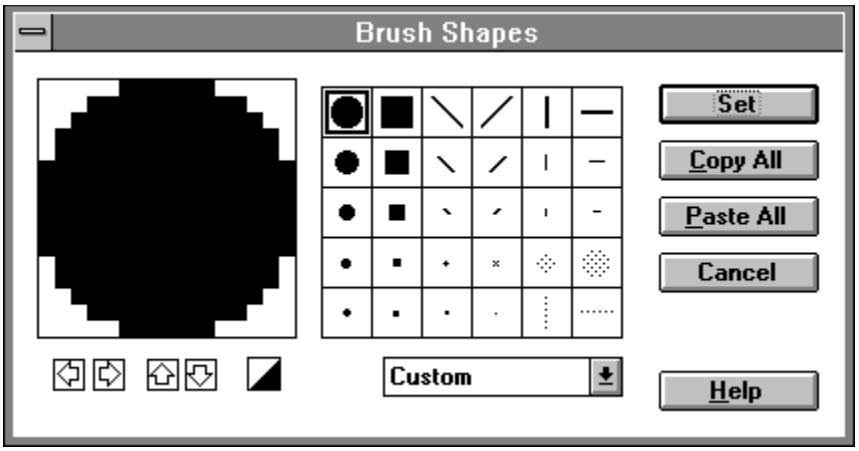
Change a custom brush shape by clicking one of the brush shapes on the right. You can edit the current brush shape by clicking the magnified image of the brush shape. Clicking a blank pixel fills it and clicking a filled pixel makes it blank.

The **right and left arrows** move the brush shape one pixel to the right or to the left.

The **up and down arrows** move the brush shape up or down one pixel.

Click the **black and white square** to reverse the colors of the brush shape (e.g., black becomes white and white becomes black).

You can store sets of custom brush shapes in the Scrapbook for future use. Click Copy All to copy the brush shapes you want to save, click Cancel, and then paste the shapes into an empty slot in the cast window. To use the brush shapes again later, display them in the paint window, select them with the selection rectangle, copy them, and then use the Paste All button in the Brush Shapes dialog box to install them again as custom shapes.


Brush Shapes
Set
Copy All
Paste All
Cancel
Custom
Help

Air Brushes command

The Air Brushes dialog box defines the size of the area the air brush covers, the size of the dots in the air brush's spray, and the flow speed of the air brush's paint. The three radio buttons at the bottom of the Air Brushes dialog box control whether the air brush sprays uniformly sized dots, randomly sized dots, or dots shaped like the currently selected paint brush.

Click here to see an illustration.

Click for more information:

- Normal
- Speckle
- Brush Shape
- Size
- Dot Size
- Flow Speed

Tip: Double-clicking the air brush in the paint window tool palette is a shortcut for choosing this command.

Size (Air Brushes dialog box)

To set the size of the air brush's spray area, drag the Size scroll bar.

Dot Size (Air Brushes dialog box)

To set the size of the dots sprayed by the air brush, drag the Dot Size scroll bar.

Flow Speed (Air Brushes dialog box)

Controls how fast the air brush covers an area with paint. To change the flow, drag the Flow Speed scroll bar.

Normal (Air Brushes dialog box)

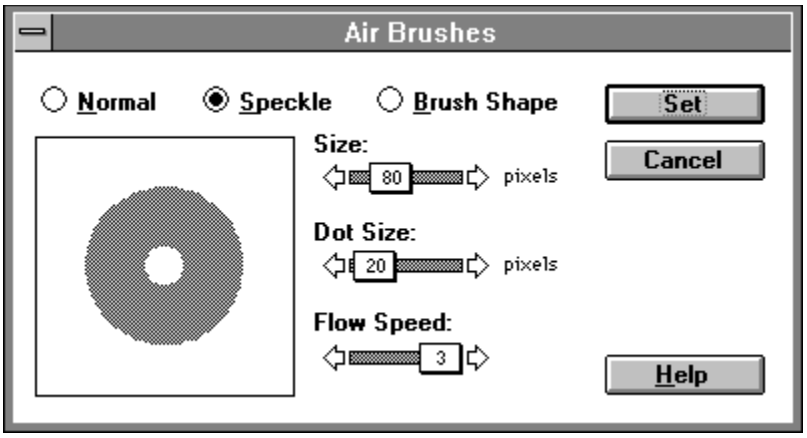
Click Normal if you want the drops sprayed by the air brush to be uniformly sized.

Speckle (Air Brushes dialog box)

Click Speckle to paint with randomly sized drops.

Brush Shape (Air Brushes dialog box)

Click Brush Shape to spray with drops shaped like the current paint brush.


Air Brushes
Normal
Speckle
Brush Shape
Set
Cancel
Size:
80
pixels
Dot Size:
20
pixels
Flow Speed:
3
Help

Gradients command

The Gradient effect creates a blend of colors that you can use for backgrounds, highlights, shading, and special effects. Limited gradient effects can be created with a black and white monitor.

Click here to see an illustration.

Click for more information:

- Direction
- Cycles
- Method
- Spread
- Range

The foreground color and gradient destination color can be selected in the paint window with the gradient destination color chip or with the controls in the Gradients dialog box. When in the paint window, use the pop-up palette on the foreground color chip to select the foreground color. Pick the destination color with the pop-up palette at the right side of the gradient selection bar above the foreground and background color chips. The current foreground color is displayed at the left of the gradient destination color chip and the current destination color is displayed on the right side of the gradient destination color chip.

When you choose Gradients from the Paint menu you can set the foreground and destination colors as well as the pattern to use with your gradient. The Gradients dialog box also has several drop-down lists to control the style of your gradient fill. Each choice you make is immediately previewed in the Gradients dialog box.

Direction (Gradients dialog box)

This drop-down lists determines the way the gradient fills an area in the paint window. The lists determine where the dark and light colors of your blend are located.

Top to Bottom--Puts the foreground color at the top and the destination color at the bottom.

Bottom to Top--Puts the destination color at the top and the foreground color at the bottom.

Left to Right--Puts the foreground color on the left and the destination color on the right.

Right to Left--Puts the foreground color on the right and the destination color on the left.

Directional--Lets you determine the direction of the gradient. When you select this option, you will be able to set the direction of the gradient in the paint window with the paint tool used to fill the area. For example, if you use the paint bucket to gradient fill an area, a directional line appears as soon as you click the paint bucket in the area to be filled. Move the line without clicking the mouse button to establish the direction of the gradient. When the line is in the direction you want, click to fill the area with the gradient in the direction you specify. The length of the line has no effect.

Sun Burst--Starts filling at the edge of the artwork and moves in concentric circles to the center

Cycles (Gradients dialog box)

The options on the Cycles drop-down list control the number of times the gradient is created within one filled area, and whether or not the colors cycle through the palette in one direction only, or auto reverse at the end of one pass through the palette. Sharp cycles have a banded appearance, while smooth cycles go from foreground to destination, then back to foreground.

One Cycle--Takes the gradient once through the range of colors you define.

Two Sharp--Takes the gradient through the range of colors twice, from foreground to destination and from foreground to destination.

Two Smooth--Takes the gradient from foreground to destination then from destination to foreground.

Three Sharp--Takes the gradient from foreground to destination three times.

Three Smooth--Takes the gradient from foreground to destination, destination to foreground, foreground to destination.

Four Sharp--Takes the gradient four times from foreground to destination.

Four Smooth--Takes the gradient from foreground to destination, destination to foreground, foreground to destination, and destination to foreground.

Method (Gradients dialog box)

The Method drop-down list commands control whether the gradient is made with the pattern you select with the pattern chip pop-up palette in the paint window, or with a dithered pattern. You can clearly see the difference in methods when you have a pattern rather than a solid color selected.

Pattern Best Colors--Ignores the order of the colors in the palette and only uses colors that create a continuous blend of the foreground and destination colors.

Pattern Best Colors See Thru--Ignores the order of the colors in the palette and only uses those colors that create a continuous blend of the foreground and destination colors. White pixels in patterns created with this method are transparent.

Pattern Adjacent Colors--Uses all the colors in the palette between the foreground and destination for the gradient.

Pattern Adjacent Colors See Thru--Uses all the colors in the palette between the foreground and destination for the gradient. White pixels in patterns created with this method are transparent.

Dither Best Colors--Ignores the order of the colors in the palette and only uses colors that create a continuous blend from foreground to destination colors and blends them with a dithered pattern. Dithering is a technique of creating color with two or more colors of pixels interspersed together.

Dither Adjacent Colors--Uses all colors between the foreground and destination colors and blends them with a dithered pattern.

Dither Two Color--Uses only the foreground and the destination colors and blends them with a dithered pattern.

Dither One Color--Uses only the foreground color and fades it with a dithered pattern.

Standard Dither--Ignores all colors between foreground and destination and adds several blended colors with a dithered pattern to create the gradient.

Multi Dither--Ignores all the colors between foreground and destination and adds several blended colors with a randomized dithered pattern to create a smooth gradient. You can interrupt the drawing of this kind of dither by clicking anywhere in the dialog box.

Spread (Gradients dialog box)

The commands in the Spread drop-down list let you choose how to distribute colors between the foreground and the destination colors of the gradient.

Equal--Provides an even spacing of colors between the foreground and the destination colors.

More Foreground--Increases the amount of the foreground color in the gradient.

More Middle--Increases the amount of the middle color in the gradient.

More Destination--Increases the amount of the destination color in the gradient.

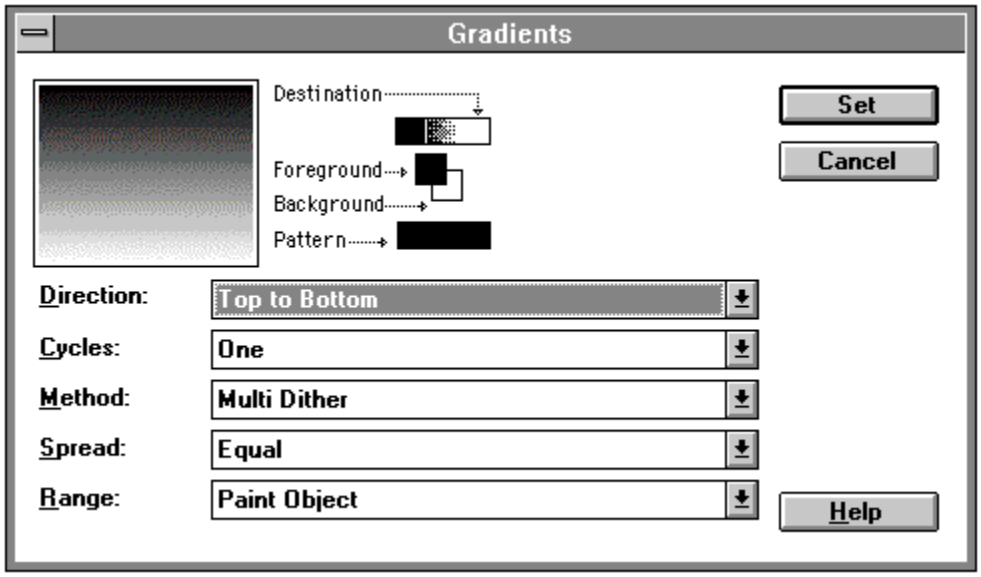
Range (Gradients dialog box)

Range commands let you determine whether the full range of the gradient is created over the paint object, cast member, or the entire paint window. The options provide greater control over how the gradient is created relative to the cast member's position on the stage or in the paint window.

Paint Object--Paints the full gradient as the fill or brush stroke of the object, regardless of the object's location in the paint window.

Cast member--Paints the full gradient with respect to the size of the cast member.

Window--Paints a full gradient only if the object is the length or width of the entire window, otherwise it paints a partial gradient corresponding to the object's location in the window.


Gradients
Destination
Foreground
Background
Pattern
Set
Cancel
Direction:
Top to Bottom
Cycles:
One
Method:
Multi Dither
Spread:
Equal
Range:
Paint Object
Help

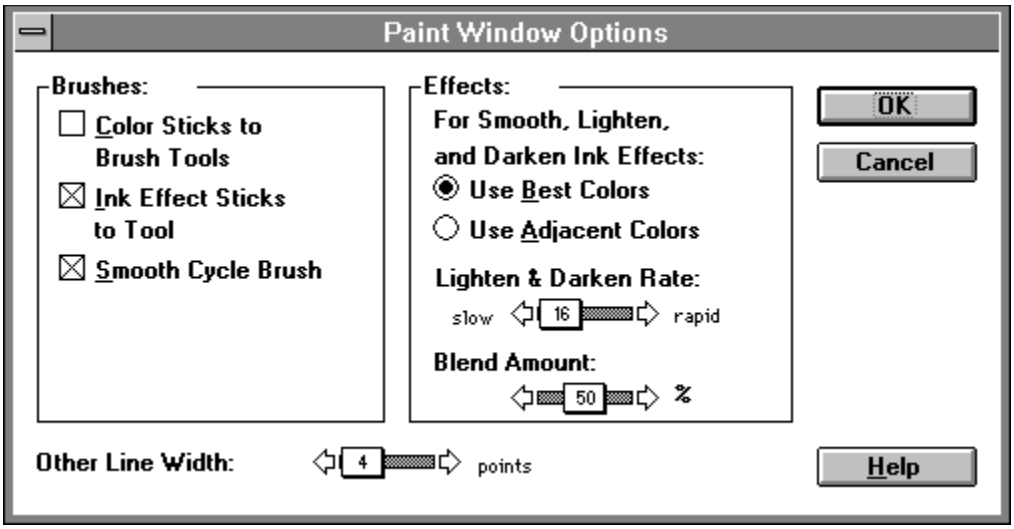
Paint Window Options command

The Paint Window Options command allows you to modify the settings of a number of tools and drawing methods in the paint window. All of the options you choose are saved in the Director 4.0 Preferences file (DIRECTOR.PRF).

Click here to see an illustration.

Click for more information:

- Brushes
- Effects
- Other Line Width


Paint Window Options
Brushes:
Color Sticks to Brush Tools
Ink Effect Sticks to Tool
Smooth Cycle Brush
Effects:
For Smooth, Lighten, and Darken Ink Effects:
Use Best Colors
Use Adjacent Colors
Lighten & Darken Rate:
slow
16
rapid
Blend Amount:
50
%
OK
Cancel
Help
Other Line Width:
4
points

Brushes (Paint Window Options)

Color Sticks To Brush Tools--If checked, the last color used with a tool stays selected the next time you use the paintbrush or air brush.

Ink Effect Sticks To Tool--If checked, the last ink used with a tool stays selected the next time you use the paintbrush or air brush.

Smooth Cycle Brush--Controls the way colors cycle when you draw with cycling ink. If checked, this option produces a cycle that goes from the foreground to the destination color and then destination to foreground. If not checked, colors cycle from foreground to destination, then repeat foreground to destination.

Effects (Paint Window Options)

These settings apply to the smooth, lighten, darken, and cycle effects.

Use Best Colors--Use Best Colors ignores the order of the colors in the palette and produces a continuous blend of the foreground and destination colors.

Use Adjacent Colors--Use Adjacent Colors uses all the colors in the palette between the foreground and destination colors when using smooth, lighten, darken, or cycle effects.

Lighten & Darken Rate--Sets the rate at which artwork changes when you use the Darken or Lighten effect in the paint window or when you choose Darken or Lighten from the Effects menu. You may have to experiment to find the best setting.

Blend Amount--Blend Amount sets the opacity of the selected color when using the Blend effect in the paint window. Use the scroll bar to vary the blend value between 0 and 100 percent.

Other Line Width (Paint Window Options)

Other Line Width allows you to set a thicker line width than the widths available in the paint window. You can change the line's width, as measured in pixels, with the scroll bar. The width you set will be the width that appears when you draw a line after selecting Other in the line width selector in the tool palette.

Effects menu

The Effects menu is available when there is a selection in the Paint window. It contains commands to transform your artwork in various ways. The full range of Effects menu commands is available to you when you use the selection rectangle to select the artwork. Not all the Effects menu commands are available to you when you select the artwork with the lasso.

Click a menu command for more information:

Invert Colors command

The Invert Colors command reverses the colors of the selected area in the paint window. To see what the reverse colors are, open the color palettes window, select all the colors in the palette, and choose Reverse Colors from the Palette menu. You'll see that the effect is an upside-down mirror image of the palette. If you are working in black and white, Invert Colors changes black to white and vice versa.

Flip Horizontal command

Turns the selected area in the paint window horizontally from right to left.

Flip Vertical command

Turns the selected area in the paint window vertically from top to bottom.

Trace Edges command

Trace Edges creates an outline around the edges of the selected artwork. The outline is the same color as the selected line, if the line is a solid color. If the original line is multicolored, an outline is created for each section of the line. You can add multiple outlines by choosing Trace Edges or Repeat Effect repeatedly.

Fill command

The Fill command fills a selected area with the current foreground color and pattern.

Darken command

Reduces the brightness of the selected artwork. It is the same as decreasing the brightness in the color palettes window or luminosity in the Windows Color dialog box.

Lighten command

Increases the brightness of anything in the selection rectangle. It is equivalent to increasing the brightness in the color palettes window or luminosity in the Windows Color dialog box.

Smooth command

Softens the edges of the selected artwork by adding pixels of blended color to the artwork's edges.

Switch Colors command

Switch Colors changes each pixel that is the currently selected foreground color to the currently selected destination color.

Note: This command only works for images whose color depth is 8-bits or less.

Rotate Left command

Rotates the selected area in the paint window 90 degrees counterclockwise.

Rotate Right command

Rotates the selected area in the paint window 90 degrees clockwise.

Free Rotate command

Allows you to rotate the selected area in the paint window any number of degrees clockwise or counterclockwise. When you choose Free Rotate, handles appear at the corners of the selection rectangle. To rotate the artwork, drag any handle in the desired direction.

The Free Rotate command is one of four special effects you can apply to artwork selected with the selection rectangle in the paint window. The other special effects are Perspective, Slant, and Distort. When you choose one of the special effects after selecting the artwork, handles appear at each corner of the selection. Dragging the handle produces the desired effect.

Perspective command

Perspective stretches the selected artwork to give it a perspective effect. When you choose Perspective, handles appear at the corners of the selection rectangle. Drag one or more handles to create the effect you want. For example, you can bring the two top handles closer together to create the illusion of linear perspective.

Tip: To make artwork appear to be vanishing into the distance, choose Perspective and move the handles on one side of your selection together and the handles on the opposite side of your selection apart.

Slant command

The Slant command skews the selected artwork. When you choose Slant, handles appear at the corners of the selection rectangle. Dragging a handle in the desired direction moves the opposing corner an equal amount in the same direction, maintaining a parallelogram shape.

Distort command

When you choose Distort, handles appear at the corners of the selection rectangle. Each corner can be dragged in any direction independent of the other corners. When you release the mouse button, the selected artwork assumes the shape that you have created.

Auto Distort command

The Auto Distort command automatically generates in-between positions for any cast member that is free rotated, made into a perspective, slanted, distorted, or stretched. After artwork has been altered with one of these five effects, and before you deselect the artwork, choose Auto Distort, and type the number of in-between cast members in the Create New Cast Members field in the Auto Distort dialog box. The new cast members are placed in the next available cast member positions.

Click here to see an illustration.

For a rotated bitmap image, Auto-Distort uses the center of the image as the rotation point. Consequently, you will have to use the Paint Window's registration tool to reset the registration point of each in-betweened cast member created by the Auto-Distort operation.

To reset the registration point:

1. Before you rotate the image, create a temporary dot in the image at the desired rotation point.

   Make this dot a unique color so you can change it to the proper color later.

2. Select the image and rotate it using the appropriate rotate command on the Effects menu.

3. Choose the Auto Distort command and specify the number of in-between cast members.

4. Use the registration tool to click the colored dot within each cast member in the rotation sequence.

5. Use the Switch Colors command to change the colored dot to the appropriate color.

Tip: Use the Auto Distort command in conjunction with Rotate Left, Rotate Right, Free Rotate, Perspective, Slant, or Distort to quickly create a number of cast members in between the artwork you selected and the artwork that has been changed. Auto Distort also works with artwork that is stretched or squeezed in the paint window.


Auto Distort
First: Use Rotate, Slant, Perspective, or Distort.
Then: Use this command to generate cast members for in-betweens.
Create
Cancel
Create
1
New Cast Members
Help

Repeat Effect command

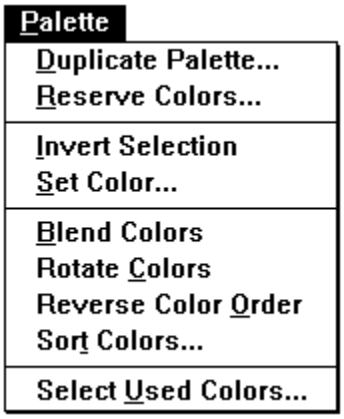
Repeats the last command chosen from the Effects menu for the selected area. It's a shortcut for choosing that command again.

Tip: Control-Y is the keyboard shortcut for this command.

Palette menu

The Palette menu appears when the color palettes window is the active window. It contains commands for editing the current palette and for sorting its colors, and options for changing the colors in the palette.

Click a menu command for more information:



Duplicate Palette command

Duplicates the current palette, asks you to name it, and puts it in the next available position in the cast window. Director automatically duplicates a palette if you change any of the colors in one of the built-in palettes. The duplicated palette is added to the list of available palettes in the color palettes window's pop-up menu. A duplicated palette in the cast of a movie becomes part of that file and is always available in that movie until it is cut or cleared from the cast window.

Reserve Colors command

Reserving colors in a palette is a way to isolate specific colors used in palette effects like color cycling. For example, if you are cycling colors and don't want to inadvertently use the cycling colors on a non-cycling cast member, reserve the cycling colors to prevent them from being used.

Click here to see an illustration.

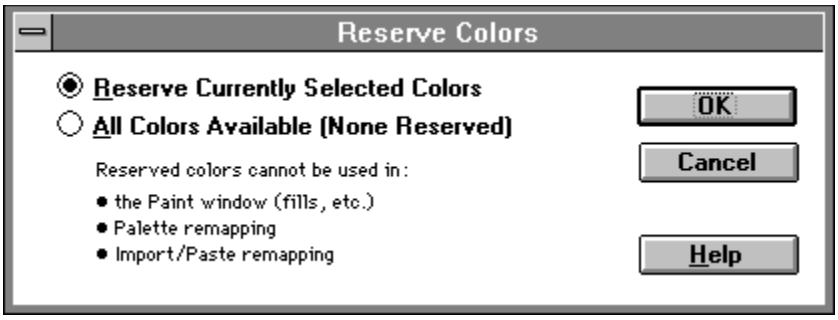
When Director automatically selects colors (such as in a gradient, when pasting or importing a bitmap, or when re-mapping a cast member), reserved colors are not available. The only exceptions are gradients that use adjacent colors. They will use any reserved colors in the gradients color range.

To reserve colors, select them in the color palettes window and then choose this command.

Make sure Reserve Currently Selected Colors is chosen, then click OK. Director puts a check mark next to the Reserve Colors command, to indicate that colors are reserved. (If you later make all colors available by choosing All Colors Available, Director removes the check.)

Reserved colors appear striped in the color palettes window and in the pop-up palettes in the paint window. The Select Reserved button also appears in the window when you've reserved colors to help you see which colors are reserved.

Click the Select Reserved button to select all the colors that are reserved.


Reserve Colors
Reserve Currently Selected Colors
All Colors Available (None Reserved)
Reserved colors cannot be used in:
• the Paint window (fills, etc.)
• Palette remapping
• Import/Paste remapping
OK
Cancel
Help

Invert Selection command

When you choose Invert Selection after choosing a color or range of colors in the color palettes window, your selection is replaced by a new selection, which consists of all the colors that were not part of your original selection.

Set Color command

The Set Color command is dimmed unless you have a color selected in the color palettes window. When you choose this command, the Windows Color dialog box appears. You can adjust the color of your selection in the color palettes window with the Color dialog box. For information about using the Color dialog box, see the user's guide that came with your computer.

Tip: Double-clicking a color in the color palettes window is the shortcut for selecting a color and choosing this command.

Blend Colors command

Creates a blend of the first and last color of a selected range in the palette, producing a smooth color transition from the first selected color to the last selected color. Use it to create blends of color for color cycling or smooth gradients.

Tip: Press the Control key while selecting colors with the pointer or hand tool to make a discontiguous selection.

Rotate Colors command

Rotating selected colors displaces all the selected colors one square to the left. The leftmost color wraps around and appears at the last bottom right square. Each time you choose Rotate Colors, the selected colors shift by one more square. As the colors reach the left edge of the selection, they wrap around to the right edge and continue their journey. It is similar to the movement of color within a palette that you can see while colors cycle.

This is precisely what goes on when you use color cycling. If you are using a paint tool in the paint window with a cycling ink effect, the colors rotate through the palette as you draw. Another example of color cycling is in the Set Palette dialog box in the Score menu. You can select a range of colors to cycle as your movie plays. Any cast member that is the same color as one of the cycled colors will change as the colors rotate through the palette.

Reverse Color Order command

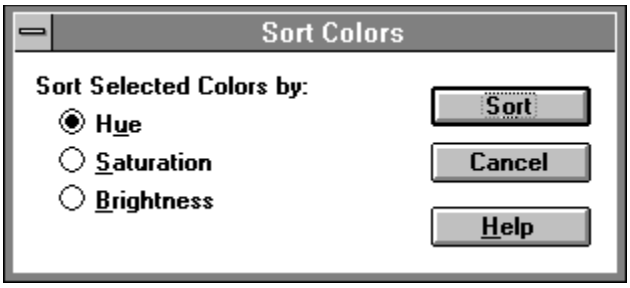
Reverses the order of the selected colors: the first color of the palette becomes the last. The colors, however, remain unchanged. If you reverse the palette that is associated with a cast member, it produces a negative image of the cast member.

Sort Colors command

The Sort Colors command allows you to sort the selected colors in the palette by Hue, Saturation, or Brightness.

Click here to see an illustration.

When you click Sort, the colors will appear in the palette according to the sorting order selected. If you sort colors after drawing cast members, the cast members that use the selected colors will also change color as the colors are sorted.


Sort Colors
Sort Selected Colors by:
Hue
Saturation
Brightness
Sort
Cancel
Help

Select Used Colors command

The Select Used Colors command highlights colors that are used by the bitmap cast member or cast members selected in the cast window. This command is useful when you want to cycle only the colors contained in one specific cast member, or when you need to find those colors you can cycle without affecting a cast member. You can also use the Invert Selection command, after you select the used colors, to select the colors that are not used by a cast member.

The Lingo menu appears when the message or script windows are active. This menu displays the complete set of Lingo commands that you can use to create scripts for your movie.

Use the Lingo menu to enter any Lingo command into a script. This saves you from typing the command and also eliminates typing errors. Simply choose the command you want from the appropriate submenu. The command is entered in the script at the insertion point. For a complete description of the Lingo menu, refer to the *Using Lingo* manual, or use the help pointer to display an explanation for a particular command.

Click the first letter of a Lingo element (or click Operators) for more information:

Click a Lingo element for more information:

- # (pound sign)
- - (minus sign, Lingo)
- -- (double hyphen)
- & (concatenator)
- && (concatenator)
- * (asterisk)
- + (plus sign)
- / (slash)
- - (minus sign, Lingo)
- < (less than)
- <= (less than or equal to)
- <> (not equal)
- = (equal sign)
- > (greater than)
- >= (greater than or equal to)
- [] (list brackets)
- ¬ (new line character)
- ( ) [parenthesis]

Click a Lingo element for more information:

- abbr
- abbrev
- abbreviated
- abort
- abs
- actorList
- add
- addAt
- addProp
- after
- alert
- ancestor
- and
- append
- atan

Click a Lingo element for more information:

- backColor of cast
- backColor of sprite
- BACKSPACE
- beep
- beepOn
- before
- birth
- blend of sprite
- bottom of sprite
- buttonStyle

Click a Lingo element for more information:

- cast
- cast backColor
- cast castType
- cast depth
- cast fileName
- cast forecolor
- cast height
- cast hilite
- cast loaded
- cast name
- cast number
- cast palette
- cast picture
- cast purgePriority of cast
- cast rect
- cast regPoint
- cast scriptText
- cast text
- cast width
- castmembers
- castNum of sprite
- castType of cast
- center of cast
- centerStage
- char…of
- chars
- charToNum
- checkBoxAccess
- checkBoxType
- checkMark of menuItem
- clearGlobals
- clickLoc
- clickOn
- close window
- closeDA
- closeResFile
- closeXlib
- colorDepth
- colorQD
- commandDown
- constrainH
- constraint of sprite
- constrainV
- contains
- continue
- controlDown
- controller of cast
- copyToClipBoard
- cos
- count
- crop of cast
- cursor
- cursor of sprite

Click a Lingo element for more information:

- date
- delay
- delete
- deleteAt
- deleteProp
- depth of cast
- digitalVideo
- digitalVideo cast center
- digitalVideo cast controller
- digitalVideo cast crop
- digitalVideo cast directToStage
- digitalVideo cast duration
- digitalVideo cast frameRate
- digitalVideo cast loop
- digitalVideo cast pausedAtStart
- digitalVideo cast preload
- digitalVideo cast sound
- digitalVideo cast video
- digitalVideo sprite movieRate
- digitalVideo sprite movieTime
- digitalVideo sprite startTime
- digitalVideo sprite stopTime
- digitalVideo sprite volume
- directToStage
- do
- done
- dontPassEvent
- doubleClick
- down
- drawRect of window
- duplicate cast
- duration of cast

Click a Lingo element for more information:

- editableText of sprite
- else
- EMPTY
- enabled of menuItem
- end
- ENTER
- enterFrame
- erase cast
- exit
- exit repeat
- exitFrame
- exitLock
- exp

Click a Lingo element for more information:

- factory
- fadeIn
- fadeOut
- FALSE
- field
- fileName of cast
- fileName of window
- findEmpty
- findPos
- findPosNear
- fixStageSize
- float
- floatP
- floatPrecision
- foreColor of cast
- foreColor of sprite
- forget window
- frame
- frameLabel
- framePalette
- frameRate of cast
- frameScript
- framesToHMS
- frameTempo
- freeBlock
- freeBytes

Click a Lingo element for more information:

- getaProp
- getAt
- getLast
- getNthFileNameInFolder
- getOne
- getPos
- getProp
- getPropAt
- global
- go
- go loop
- go next
- go previous

Click a Lingo element for more information:

halt
height of cast
height of sprite
hilite
hilite of cast
HMStoFrames

Click a Lingo element for more information:

idle
if
ilk
importFileInto
in
inflate rect
ink of sprite
inside
inside point
installMenu
instance
integer
integerP
intersect
intersect rect
intersects
into
item…of
itemDelimiter
items

Click a Lingo element for more information:

- key
- keyCode
- keyDown
- keyDownScript
- keyUp
- keyUpScript

Click a Lingo element for more information:

- label
- labelList
- last
- lastClick
- lastEvent
- lastFrame
- lastKey
- lastRoll
- left of sprite
- length
- line…of
- lines
- lineSize of sprite
- list
- list brackets ([])
- listP
- loaded of cast
- locH of sprite
- locV of sprite
- log
- long
- loop
- loop of cast

Click a Lingo element for more information:

- machineType
- map
- map point
- map rect
- marker
- mAtFrame
- max
- maxInteger
- mci
- mDescribe
- mDispose
- me
- memorySize
- menu
- menuItem
- menuItems
- menus
- method
- mGet
- min
- mInstanceRespondsTo
- mMessageList
- mName
- mNew
- mod
- modal of window
- modified of cast
- mouseCast
- mouseChar
- mouseDown
- mouseDownScript
- mouseH
- mouseItem
- mouseLine
- mouseUp
- mouseUpScript
- mouseV
- mouseWord
- move cast
- moveableSprite of sprite
- moveToBack
- moveToFront
- movie
- movieFileFreeSize
- movieFileSize
- movieName
- moviePath
- movieRate of sprite
- movieTime of sprite
- mPerform
- mPut
- mRespondsTo
- multiSound

Click a Lingo element for more information:

- name of cast
- name of menu
- name of menuItem
- next
- next repeat
- not
- nothing
- number of cast
- number of castmembers
- number of chars in
- number of items in
- number of lines in
- number of menuItems
- number of menus
- number of words in
- numToChar

Click a Lingo element for more information:

objectP
of
offset
offset rect
on
on enterFrame
on exitFrame
on idle
on keyDown
on keyUp
on mouseDown
on mouseUp
on startMovie
on stepMovie
on stopMovie
open
open window
openDA
openResFile
openXlib
optionDown
or

Click a Lingo element for more information:

palette of cast
param
paramCount
pass
pasteClipBoardInto
pathName
pause
pausedAtStart
pauseState
perFrameHook
pi
picture of cast
pictureP
play
play done
playFile
point
power
preLoad
preLoad of cast
preLoadCast
preLoadEventAbort
preLoadRAM
previous
printFrom
property
puppet of sprite
puppetPalette
puppetSound
puppetSprite
puppetTempo
puppetTransition
purgePriority of cast
put
put…after
put…before
put…into

Click a Lingo element for more information:

quickTimePresent
quit
QUOTE

Click a Lingo element for more information:

- ramNeeded
- random
- randomSeed
- rect
- rect of cast
- rect of window
- rect point
- regPoint
- repeat while
- repeat with
- repeat with…down to
- repeat with i in list
- restart
- result
- RETURN
- return
- right of sprite
- rollOver
- romanLingo

Click a Lingo element for more information:

saveMovie
scoreColor
script of menuItem
scriptNum of sprite
scriptText of cast
searchCurrentFolder
searchPath
selection
selEnd
selStart
set…to and set…=
setaProp
setAt
setCallBack
setProp
shiftDown
short
showGlobals
showLocals
showResFile
showXlib
shutDown
sin
size of cast
sort
sound close
sound fadeIn
sound fadeOut
sound of cast
sound playFile
sound stop
soundBusy
soundEnabled
soundLevel
sourceRect
sprite
sprite…intersects
sprite…within
spriteBox
sqrt
stage
stageBottom
stageColor
stageLeft
stageRight
stageTop
startMovie
starts
startTime of sprite
startTimer
stepMovie
stillDown
stop
stopMovie

Click a Lingo element for more information:

TAB
tan
tell
text of cast
textAlign of field
textFont of field
textHeight of field
textSize of field
textStyle of field
the
then
ticks
time
timeoutKeyDown
timeoutLapsed
timeoutLength
timeoutMouse
timeoutPlay
timeoutScript
timer
title of window
titleVisible of window
to
top of sprite
trace
traceLoad
traceLogFile
trails of sprite
TRUE
type of sprite

Click a Lingo element for more information:

union rect
unload
unloadCast
updateMovieEnabled
updateStage

Click a Lingo element for more information:

value
version
video of cast
visible of sprite
visible of window
voidP
volume of sound
volume of sprite

Click a Lingo element for more information:

- when keyDown
- when mouseDown
- when mouseUp
- when timeOut
- while
- width of cast
- width of sprite
- window
- windowList
- windowType
- with
- within
- word…of

Click a Lingo element for more information:

xFactoryList

Click a Lingo element for more information:

zoomBox

(pound sign in Lingo)

Syntax: *#symbolName*

This symbol definition operator defines a symbol. In addition to integers, floating-point numbers, strings, and objects, Lingo also has a symbol data type. A *symbolName* begins with an alphabetical character and may be followed by any number of alphabetical or numerical characters.

The valid operations on symbols are:

- Assignment to a variable
- Comparison
- Being passed as a parameter to a handler or method
- Being returned as a value from a handler or method.

Symbols take up much less space than strings and can be manipulated faster than strings can. Essentially, symbols have the speed and memory advantages of integers but give you the descriptive power of strings.

A symbol is a self-contained unit, which can be used to represent a condition or flag. It does not consist of individual characters in the same sense as a string. However, you can convert a symbol to a string for display purposes by using the string function.

Related topics: symbolP function

Lingo example: # (pound sign)

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the variable named state to the symbol #Playing:

```
put #Playing into state
```

- (minus sign, Lingo)

Syntax (negation): *-expression*

This arithmetic operator reverses the sign of the value of the expression.

- The usage *-expression* reverses the sign of the value of the expression. This is an arithmetic operator with a precedence level of 5.

- The usage *expression1 - expression2* subtracts *expression2* from *expression1*. When both expressions are integers, the difference is an integer. When either or both expressions are floating-point numbers, the difference is a floating-point number. This is an arithmetic operator with a precedence level of 3.

Syntax (subtraction): *expression1 - expression2*

This arithmetic operator performs an arithmetic subtraction on two numerical expressions, subtracting *expression2* from *expression1*. When both expressions are integers, the difference is an integer. When either or both expressions are floating-point numbers, the difference is a floating-point number.

This is an arithmetic operator with a precedence level of 3.

Example

Lingo example: minus sign

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement reverses the sign of the expression 2 + 3.

```
put -(2 + 3)
```

The result is -5.

This statement subtracts the integer 2 from 5, and then displays the result in the message window:

```
put 5 - 2
```

The result is 3, which is an integer.

This statement subtracts the floating-point number 1.5 from 3.25, and then displays the result in the message window:

```
put 3.25 - 1.5
```

The result is 1.75, which is a floating-point number.

Double hyphen (Lingo)

Syntax: `--` [*comment*]

This comment delimiter symbol indicates the beginning of a script comment. On any line, what is between the comment symbol (double hyphen) and the end-of-line return character is interpreted as a comment instead of a Lingo statement.

Example

Lingo example: double hyphen

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler uses a double hyphen to make the second line a comment:

```
on resetColors
   -- This handler resets the sprite's colors.
   set the foreColor of sprite 1 to 35 -- bright red
      set the backColor of sprite 1 to 36 -- light blue
end resetColors
```

& (concatenator in Lingo)

Syntax: *expression1* & *expression2*

This operator performs a string concatenation of two expressions. If either *expression1* or *expression2* evaluates to a number, it is first converted to a string. The resulting expression is a string.

This is a text operator with a precedence level of 2.

Example

Lingo example: & (concatenator)

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement concatenates the strings "abra" and "cadabra":

```
put "abra" & "cadabra"
```

The result is the string "abracadabra".

This statement concatenates the strings "$" and the content of the variable named price. The concatenated string is then assigned to the text cast member Price:

```
put "$" & price into field "Price"
```

&& (concatenator in Lingo)

Syntax: *expression1* `&&` *expression2*

This text operator concatenates two expressions, inserting a space character between the original string expressions. If either *expression1* or *expression2* evaluates to a number, it is first converted to a string. The resulting expression is a string.

This is a text operator with a precedence level of 2.

Example

Lingo example: && (concatenator)

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement concatenates the strings "abra" and "cadabra", and inserts a space between the two:

```
put "abra" && "cadabra"
```

The result is the string "abra cadabra".

This statement concatenates the strings "Today is" and today's date in the long format, and inserts a space between the two:

```
put "Today is" && the long date
```

If today's date were Tuesday, March 15, 1994, the result would be Tuesday, March 15, 1994.

parentheses (Lingo)

Syntax: （*expression* ）

This grouping operator performs a grouping operation on an expression. It is used to control the order of execution of the operators in an expression, and override the automatic precedence order. It causes the expression contained within the parentheses to be evaluated first. When parentheses are nested, the contents of inner ones are evaluated before the contents of outer ones.

This is a grouping operator with a precedence level of 5.

Example

Lingo example: parentheses

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

These statements use the grouping operator to change the order in which operations occur. The result appears below each statement:

```
put (2 + 3) * (4 + 5)

-- 45

put 2 + (3 * (4 + 5))

-- 29

put 2 + 3 * 4 + 5

-- 19
```

* (multiplication sign in Lingo)

Syntax: *expression1* * *expression2*

This arithmetic operator performs an arithmetic multiplication on two numerical expressions. If both expressions are integers, the product is an integer. If either or both expressions are floating-point numbers, the product is a floating-point number.

This is an arithmetic operator with a precedence level of 4.

Example

Lingo example: * (multiplication sign)

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement multiplies the integers 2 and 3, and then displays the result in the message window:

```
put 2 * 3
```

The result is 6, which is an integer.

This statement multiplies the floating-point numbers 2.0 and 3.1414, and then displays the result in the message window:

```
put 2.0 * 3.1416
```

The result is 6.2832, which is a floating-point number.

+ (plus sign in Lingo)

Syntax: *expression1* + *expression2*

This arithmetic operator performs an arithmetic sum on two numerical expressions. If both expressions are integers, the sum is an integer. If either or both expressions are floating-point numbers, the sum is a floating-point number.

This is an arithmetic operator with a precedence level of 4.

Example

Lingo example: + (plus sign)

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement adds the integers 2 and 3, and then displays the result in the message window:

```
put 2 + 3
```

The result is 5, which is an integer.

This statement adds the floating-point number 2.5 and 3.25, and then displays the result in the message window:

```
put 2.5 + 3.25
```

The result is 5.75, which is a floating-point number.

/ (division sign in Lingo)

Syntax: *expression1* / *expression2*

This operator performs an arithmetic division on two numerical expressions, dividing *expression1* by *expression2*. If both expressions evaluate to integers, the quotient is an integer. If either or both expressions evaluate to floating point numbers, the quotient is a floating point number.

This is an arithmetic operator with a precedence level of 4.

Example

Lingo example: / (division sign)

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement divides the integer 22 by 7, and then displays the result in the message window:

```
put 22 / 7
```

The result is 3, which is an integer.

This statement divides the floating-point number 22.0 by 7.0, and then displays the result in the message window:

```
put 22.0 / 7.0
```

The result is 3.1429, which is a floating-point number.

< (less than in Lingo)

Syntax: *expression1* < *expression2*

This comparison operator compares two expressions. When *expression1* is less than *expression2*, the condition is TRUE. When *expression1* is greater than or equal to *expression2*, the condition is FALSE.

This operator can compare strings as well as integers and floating-point numbers.

This is a comparison operator with a precedence level of 1.

<= (less than or equal to in Lingo)

Syntax: expression1 <= *expression2*

This comparison operator compares two expressions. When *expression1* is less than or equal to *expression2*, the condition is TRUE. When *expression1* is greater than *expression2*, the condition is FALSE.

This operator can compare strings as well as integers and floating-point numbers.

This is a comparison operator with a precedence level of 1.

<> (not equal in Lingo)

Syntax: *expression1* `<> expression2`

This comparison operator compares two expressions. When *expression1* is not equal to *expression2*, the condition is TRUE. When *expression1* is equal to *expression2*, the condition is FALSE.

This operator can compare strings as well as integers and floating-point numbers.

This is a comparison operator with a precedence level of 1. This operator also works with symbols and objects.

= (equal sign in Lingo)

Syntax: *expression1* = *expression2*

This comparison operator compares two expressions or strings. When *expression1* is equal to *expression2*, the condition is TRUE. When *expression1* is not equal to *expression2*, the condition is FALSE.

This operator can compare strings as well as integers and floating-point numbers.

This is a comparison operator with a precedence level of 1. This operator also works with symbols and objects.

> (greater than in Lingo)

Syntax: *expression1* `> expression2`

This comparison operator compares two expressions. When *expression1* is greater than *expression2*, the condition is TRUE. When *expression1* is less than or equal to *expression2*, the condition is FALSE.

This operator can compare strings as well as integers and floating-point numbers.

This is a comparison operator with a precedence level of 1.

>= (greater than or equal to in Lingo)

Syntax: *expression1* >= *expression2*

This comparison operator compares two expressions. When *expression1* is greater than or equal to *expression2*, the condition is TRUE. When *expression1* is less than *expression2*, the condition is FALSE.

This operator can compare strings as well as integers and floating-point numbers.

This is a comparison operator with a precedence level of 1.

[ ] (list brackets Lingo)

Syntax: `[entry1, entry2, entry3, ...]`

The list brackets specify that the entries enclosed are a list.

There are four types of lists:

- Unsorted linear lists
- Sorted linear lists
- Unsorted property lists
- Sorted property lists.

Each entry in a linear list is a single value that has no other property associated with it. Each entry in a property list consists of a value and a property. The property appears before the value and is separated from the value by a colon. You cannot store a property in a linear list. When using strings as entries in a list, enclose the string in quotation marks.

For example, [6, 3, 8] is a linear list. The numbers have no properties associated with them. However, [#gears:6, #balls:3, #ramps:8] is a property list. Each number has a property, in this case a piece of machinery, associated with it. This property list could be useful for tracking how many of each piece of machinery are currently on the stage in the mechanical simulation. Properties can appear more than once in a property list.

Lists can be sorted in alphanumeric order. A sorted linear list is ordered by the values in the list. A sorted property list is in order by the properties in the list. You sort a list by using the appropriate command for a linear list or property list.

A linear list or a property list can contain no values at all. An empty list consists of two square brackets ([]). To clear a list, set the list to [].

You can modify, test, or read items in a list.

Lists are an alternative to factories and mGet and mPut, which were supported in previous versions of Director. In Director 4, it is recommended that you use lists; they are a simpler way of accomplishing the same result.

You do not have to worry about explicitly disposing of lists. Lists are automatically disposed of when they are no longer referred to by any variable. When the list is held within a global variable, it persists from movie to movie.

Example

Related topics: add, addAt, append, count, deleteAt, findPos, findPosNear, getaProp, getAt, getLast, getPos, setAt, setaProp, sort commands; ilk, list, min, max functions

Lingo example: [] (list operators)

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement defines a list by making the variable machinery equal to the list:

```
set machinery = [#gears:6, #balls:3, #ramps:8]
```

This handler sorts the list machinery, and then displays the result in the message window:

```
on sortList machinery
   sort machinery
   put machinery
end sortList
```

The result is [#balls:3, #gears:6, #ramps:8].

This statement creates an empty linear list:

```
set x = []
```

This statement creates an empty property list:

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

¬ (continuation symbol in Lingo)

Syntax: *first part of a statement on this line ¬*
second part of same statement on next line ¬
third part of same statement

The special character, when used as the last character in a line, makes the statement continue on the next line. Lingo then interprets the lines as one continuous statement. You can do this on several successive lines.

Create this character by pressing Alt-Return under Windows or Option-Return on the Macintosh.

Example

Lingo example: ¬ (continuation symbol)

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement uses the ¬ character to wrap the statement onto several lines:

```
set the castNum of sprite mySprite ¬
   to the number of cast ¬
   "This is a long cast name."
```

abbr, abbrev, abbreviated (Lingo)

See the date and time functions.

abort (Lingo)

Syntax: `abort`

This command has Lingo exit the current handler and any handler that called it without executing any of the remaining statements in the handler. This differs from the `exit` keyword, which returns to the handler from which the current handler was called.

The `abort` command does not quit Director or stop the movie from playing.

Example

Lingo example: abort

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement has Lingo exit the handler and any handler that called it when the amount of free memory is less than 50K:

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

abs (Lingo)

Syntax: `abs (`*numericExpression*`)`

This function calculates the absolute value of a numerical expression. If *numericExpression* is an integer, its absolute value is also an integer. If *numericExpression* is a floating-point number, its absolute value is also a floating-point number.

The `abs` function is useful for tracking mouse and sprite movement. Use it to convert coordinate differences (which can be either positive or negative) into distances (which are always positive).

Example

Lingo example: abs

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

Example 1:
This statement calculates the absolute value of -2.2 and displays the result in the message window:

```
put abs(-2.2)
```

Example 2:
This statement determines whether the absolute value of the difference between the current mouse position and the value of the variable startV is greater than 30. If it is, the foreground color of sprite 6 is changed.

```
if abs (the mouseV - startV) > 30 then ¬set the forecolor of sprite 6 to 95
```

actorList (Lingo)

Syntax: `the actorList`

All objects in `the actorList` receive a `stepFrame` message when the playback head advances to a new frame. This makes using `the actorList` a more powerful alternative to the `perFrameHook` property used in previous versions of Lingo.

You can clear child objects from `the actorList` by setting `the actorList` to `[]`, which is an empty list.

Example

Related topic: birth command

Lingo example: actorList

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement creates a child object from the parent script Moving Ball. All three values are parameters that the script requires:

```
add the actorList, birth(script "MovingBall", 1, ¬
   200,200)
```

This statement displays the contents of the actorList in the message window:

```
put the actorList
```

This statement clears the actorList:

```
set the actorList = []
```

add (Lingo)

Syntax: `add` *linearList*, *value*

This command adds the value specified by *value* to the linear list specified by *linearList*. For a sorted list, the value is placed in its proper order. For an unsorted list, the value is added to the end of the list.

Example

Lingo example: add

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement adds the value 2 to the list named bids. The resulting list is [3, 4, 1, 2]:

```
set bids = [3, 4, 1]

add bids, 2
```

This statement adds 2 to the sorted linear list [1, 4, 5]. The new item stays in alphanumeric order because the list is sorted:

```
add bids, 2
```

addAt (Lingo)

Syntax: `addAt` *list* , *position* , *value*

This command adds a value to the list at the position specified by *position* . Use this command when you need to add an item at a specific location in a list.

Example

Lingo example: addAt

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement adds the value 8 to the fourth position in the list named bids, which is [3, 2, 4, 3, 6, 7]:

```
set bids = [3, 2, 4, 5, 6, 7]

addAt bids, 4, 8
```

The resulting value of bids is [3, 2, 4, 8, 3, 6, 7].

addProp (Lingo)

Syntax: `addProp` *list* , *property* , *value*

This command adds the property specified by *property* and its value specified by *value* to the property list specified by list. For an unsorted list, the value is added to the end of the list. For a sorted list, the value is placed in its proper order.

When the property already exists in the list, Lingo creates a duplicate property. You can avoid duplicate properties by using the `setaProp` command to change the new entry's property.

Example

Lingo example: addProp

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement adds the property named kayne and its assigned value 3 to the property list named bids, which contains [#gee: 4, #ohasi: 1]. Because the list is sorted, the new entry is placed in alphabetical order:

```
addProp bids, #kayne, 3
```

The result is the list is [#gee: 4, #kayne: 3, #ohasi: 1].

This statement adds the entry kayne: 7 to the list named bids, which now contains [#gee: 4, #kayne: 3, #ohasi: 1]. Because the list already contains the property kayne, Lingo creates a duplicate property:

```
addProp bids, #kayne, 7
```

The result is the list [#gee: 4, #kayne: 3, #kayne:7, #ohasi: 1].

See put...after command.

alert (Lingo)

Syntax: `alert` *message*

This command causes a system beep and displays an alert dialog box containing the string specified by *message*, and an OK button. This command is useful for providing error messages in your movie. The message can contain up to 255 characters.

Example

Lingo example: alert

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

The following statement produces an alert stating that there is no CD-ROM drive connected:

```
alert "There is no CD-ROM drive connected."
```

This statement produces an alert stating that a file was not found:

```
alert "The file" && QUOTE & filename & QUOTE ¬
   && "was not found."
```

ancestor (Lingo)

Syntax: *property* `ancestor`

The ancestor property allows child objects to use handlers that are not contained within the parent script. The `ancestor` property is typically used with two or more parent scripts. This is useful when you want child objects to share certain behaviors that are inherited from an ancestor, while differing in other behaviors that are inherited from the parents.

The `ancestor` property is typically assigned in the child object's birth handler within the parent script. When you send a message to a child object that does not have a defined handler, that message is forwarded to the script defined by the ancestor property.

For a complete discussion of this topic, see Chapter 10, "Parent Scripts and Child Objects," in the *Using Lingo* manual.

The ancestor script can contain independent property variables that can be accessed by child objects. To refer to property variables within the ancestor script, you must use this syntax:

This statement changes the property variable `legCount` within an ancestor script to 4:

```
set the legCount of me to 4
```

Use the syntax `the` *variableName* `of` *scriptName* to access property variables that are not contained within the current object. This statement allows the variable `myLegCount` within the child object to access the property variable `legCount` within the ancestor script:

```
put the legCount of me into myLegCount
```

Example

Related topic: birth function; me and property keywords

Lingo example: ancestor

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

The following four scripts present an example of using the ancestor property. Each of these scripts is a cast member. Using the ancestor script Animal and the parent scripts Dog and Man, they interact with one another to define objects.

The first script Dog sets the property variable breed to Mutt; sets the ancestor of Dog to the Animal script; and sets the legCount variable that is stored in the ancestor script to 4:

```
property breed, ancestor

on birth me

   set breed = "Mutt"

   set ancestor of me to birth(script "Animal")

   set the legCount of me to 4

   return me

end birth
```

The second script Man sets the property variable race to African; sets the ancestor of Man to the Animal script; and sets the legCount variable that is stored in the ancestor script to 2:

```
property race, ancestor

on birth me

   set race to "African"

   set ancestor to birth(script "Animal")

   set the legCount of me to 2

   return me

end birth
```

The third script Animal stores the property variable legCount for each child object and defines the eat handler:

```
property legCount

on birth me

   return me

end birth

on eat me, what

   put "Eating " & what

end
```

The fourth script creates a child object of Man, displays its variables in the message window, and calls the eat handler and displays it in the message window. Since the eat handler is not in the parent script Man,

Lingo finds the eat handler in the ancestor script Animal:

```
set manChild to birth(script "man")
put the legCount of manChild
-- 2put the race of manChild
-- "African"
eat manChild, "apple"
-- "Eating apple"
```

and (Lingo)

Syntax: *logicalExpression1* `and` *logicalExpression2*

This logical operator determines whether two logical expressions are both TRUE. Only when both *logicalExpression1* and *logicalExpression2* are TRUE, the result is TRUE (1). When either or both expressions are FALSE, the result is FALSE (0).

The precedence level of this logical operator is 4.

Related topics: not and or logical operators

append (Lingo)

Syntax: `append` *list*, *value*

This command adds the specified value to the end of the list, regardless of the list's type. This differs from the `add` command, which adds a value to a sorted list in accordance with the list's order.

A sorted list becomes unsorted after you use the append command to add a value to it. You can re-sort the list by using the `sort` command.

Example

Related topics: add command

Lingo example: append

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement adds the value 2 at the end of the sorted list named bids, which contains [1, 3, 4] even though this is not according to the list's sorted order:

```
set bids = [1, 3, 4]

append bids, 2
```

The resulting value of bids is [1, 3, 4, 2].

atan (Lingo)

Syntax: `atan (`*number*`)`

This function calculates the arc tangent of the specified number. The result is between -pi/2 and +pi/2.

Example

Lingo example: atan

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This expression gives the arctangent of pi/4 radians:

```
atan (pi()/4.0)
```

Note that the ¹ symbol cannot be used in a Lingo expression.

backColor of cast (Lingo)

Syntax: `set the backColor of cast` *castName* `to` *colorNumber*

This cast property sets the background color of a text cast member.

Example

Lingo example: backColor of cast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement changes the background color of the text in cast member 1 to the color in palette entry 250:

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

backColor of sprite (Lingo)

Syntax: `the backColor of sprite` *whichSprite*

This sprite property determines the background color of the sprite specified by *whichSprite*. The sprite must be a puppet before you can set its background color using Lingo. Setting the backColor using a Lingo script is the same as choosing the background color from the tools window when the sprite is selected on the stage.

The background color applies only to 1-bit bitmap and shape cast members. It does not affect how a text or button cast member is displayed. An 8-bit bitmap is affected, but generally not in a useful way.

The `backColor of sprite` value ranges from 0 to 255 for 8-bit color, and from 0 to 15 for 4-bit color. The numbers correspond to the index number of the background color in the current palette. (A color's index number appears in the color palette's lower left corner when you click the color.)

When you set this property within a script while the playback head is not moving, be sure to use the command `updateStage` to redraw the stage. If you are changing several sprite properties--or several puppet sprites--you only have to use one `updateStage` command at the end of all the changes.

One use of the `backColor of sprite` property that works consistently with 8-bit bitmap sprites is specifying which color is to be made transparent using the score ink effect Background Transparent. This is particularly useful when creating or importing anti-aliased graphics or objects from a 3-D rendering program for use over video.

Using a black stage color that is defined as the overlay color by the video card, as well as having images that are anti-aliased against a black background, usually works best. This will produce a dark gray shadow behind the graphic when used over a video source. This is the least objectionable shadow color.

The `backColor of sprite` property can be tested and set.

Example

Related topics: foreColor sprite property; stageColor property

Lingo example: backColor of sprite

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

The following statement sets the variable oldColor to the background color of sprite 5:

```
put the backColor of sprite 5 into oldColor
```

The following statement randomly changes the background color of a random sprite from sprite 11 to sprite 13 to color number 36:

```
set the backColor of sprite (10 + random(3)) to 36
```

BACKSPACE (Lingo)

Syntax: `BACKSPACE`

This character constant represents the backspace key, marked "delete" on the Macintosh keyboard.

Example

Lingo example: BACKSPACE

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This on keyDown handler checks whether the backspace key was pressed and, if it was, calls the author-defined handler clearField:

```
on keyDown

   if the key = BACKSPACE then clearField

   dontPassEventend keyDown
```

beep (Lingo)

Syntax: `beep [`*numberOfTimes* `]`

This command causes the computer to beep the number of times specified by *numberOfTimes*. If *numberOfTimes* is missing, the beep occurs once.

- For the Macintosh, the beep sound is the Alert Sound selected in the Sound control panel. If the Speaker Volume in the Sound control panel is set to 0, the menu bar flashes instead.

- Under Windows, the beep sound is the sound assigned in the Sound dialog box.

Example

Lingo example: beep

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement causes two beeps if the text field Answer is empty:

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

This handler causes up to three beeps whenever a key is pressed:

```
on keyDown
   beep random(3)
end
```

beepOn (Lingo)

Syntax: `the beepOn`

This property determines whether the computer beeps when the user clicks outside an active sprite--a sprite that has a script associated with it. If the `beepOn` property is set to TRUE, clicking outside active sprites results in a beep.

The `beepOn` property can be tested and set. The default value is FALSE.

Example

Lingo example: beepOn

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement displays an alert telling the user to click again when the user clicks outside active sprites:

```
if the beepOn = TRUE then alert "Click again."
```

This statement sets the beepOn property to TRUE:

```
set the beepOn to TRUE
```

This statement sets the beepOn to the opposite of its current setting:

```
set the beepOn to (not the beepOn)
```

before (Lingo)

See put...before command.

birth (Lingo)

Syntax: `birth (script` *parentScriptName* `,` *value1* `,` *value2* `, ...)`

This predefined function is used to create child objects from parent scripts. You may define a birth handler within a parent script that creates child objects. The child object shares all the handler definitions of the parent script. The child object has the same property variable names that are declared in the parent script, but each child object has its own values for these properties.

Because the child object is a value, it can be assigned to variables, placed in lists, and passed as parameter.

Being able to assign individual property values to child objects is the primary advantage of using birth handlers. A birth handler must be named `birth`, and must accept `me` as a parameter and return `me`.

Example

Related topics: ancestor property, me keyword

Lingo example: birth

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

Example 1:
These statements use a birth handler to create a child object of a parent script. The parent script is a movie script cast member named Bird, which contains these handlers:

```
on birth me
   return me
end
on fly me
   put "I am flying"
end
```

These statements create a child object called myBird, and make it fly by calling the fly handler in the Bird parent script:

```
set myBird to birth (script "Bird")
fly myBird
```

These statements use a new Bird parent script, which contains the property variable speed:

```
property speed
on birth me, initSpeed
   set speed to initSpeed
   return me
end
on fly me
   put "I am flying at " & speed & "mph"
end
```

Example 2:
The following statements create 2 child objects called myBird1 and myBird2. When the fly handler is called from the child object, the speed of the object is displayed in the message window:

```
set myBird1 to birth (script "Bird", 15)
set myBird2 to birth (script "Bird", 25)
fly myBird1
fly myBird2
```

This text would appear in the message window:

```
-- "I am flying at 15 mph"
-- "I am flying at 25 mph"
```

blend of sprite (Lingo)

Syntax: `the blend of sprite`

Using this sprite property, you can set or determine the puppet sprite's blend value. Blend values can be from 0 to 100, which correspond to the blend values in the Set Blend dialog box.

Example

Lingo example: blend of sprite

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the blend value of sprite 3 to 40 percent:

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

This statement displays the blend value of sprite 3 in the message window:

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

bottom of sprite (Lingo)

Syntax: `the bottom of sprite` *whichSprite*

This sprite property is the bottom vertical coordinate of the bounding rectangle of the sprite specified by *whichSprite*.

The `bottom of sprite` property can be tested, but not set directly. Use the `spriteBox` command to set the bottom vertical coordinate of a sprite.

Example

Note: Sprite coordinates are measured in numbers of pixels, starting with (0,0) at the upperleft corner of the Stage. Stage coordinates are measured in numbers of pixels, starting with (0,0) at the upperleft corner of the monitor.

Related topics: height, left, locH, locV, right, top, and width sprite properties; spriteBox command

Lingo example: bottom of sprite

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement assigns the vertical coordinate of the bottom of sprite numbered (i + 1) to the variable named lowest:

```
put the bottom of sprite (i + 1) into lowest
```

buttonStyle (Lingo)

Syntax: `the buttonStyle`

This property determines the visual response of buttons when a user clicks a button, and then moves the pointer over other buttons without releasing the mouse button.

The `buttonStyle` property can have these values:

- **0--list style:** When the `buttonStyle` property is set to 0 (list style), subsequent buttons highlight when the pointer passes over them. If the user releases the mouse button while the pointer is over such a button, the script associated with that button is activated.

- **1--dialog style:** When the `buttonStyle` property is set to 1 (dialog style) only the first button highlights. Subsequent buttons are not highlighted. If the user releases the mouse button while the pointer is over a button other than the original button clicked, the script associated with that button is not activated.

The `buttonStyle` property can be tested and set, and the default value is 0 (list style). You can use this property in any type of script.

Related topics: checkBoxAccess and checkBoxType properties

Lingo example: buttonStyle

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

The following statement sets the buttonStyle property to 1 (dialog style):

```
set the buttonStyle to 1
```

This statement remembers the current setting of the buttonStyle property by putting the current buttonStyle in the variable buttonStyleValue:

```
put the buttonStyle into buttonStyleValue
```

cast (Lingo)

Syntax: `the` *property* `of cast` *whichCastmember*

This keyword specifies to Lingo that the next expression refers to a specific cast member.

If *whichCastmember* is a string, it is used as the cast name. If *whichCastmember* is an integer, it is used as the cast number.

Example

Lingo example: cast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

The following statement sets the hilite of the button cast member named Enter Bid to TRUE:

```
set the hilite of cast "Enter Bid" to TRUE
```

This statement puts the name of sound cast member 132 into the variable soundName:

```
put the name of cast 132 into soundName
```

This statement determines whether cast member 9 has a name assigned:

```
if stringP(the name of cast 9) then exit
```

castmembers (Lingo)

See the number of castmembers property.

castNum of sprite (Lingo)

Syntax: `the castNum of sprite` *whichSprite*

This sprite property determines the number of the cast member associated with the sprite specified by *whichSprite*.

Setting this property lets you switch the cast member assigned to a sprite. The sprite must be a puppet to be able to do this.

A typical use of this is exchanging cast members when a sprite is clicked to simulate the reversed image that appears when a standard button is clicked. You can also make some action in the movie depend on which cast member is assigned to a sprite.

When you set this property within a script while the playback head is not moving, be sure to use the updateStage command to redraw the stage. If you are changing several sprite properties--or several puppet sprites--you only have to use one updateStage command after making all of the changes.

The `castNum of sprite` property can be tested and set.

Example

Related topic: number of cast property

Lingo example: castNum of sprite

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

The following statement switches the cast member assigned to sprite 3 to cast member number 35:

```
set the castNum of sprite 3 to 35
```

This statement assigns the cast member Narrator to sprite 10 by setting the castNum of sprite to Narrator's cast number:

```
set the castNum of sprite 10 = cast "Narrator"
```

castType of cast

Syntax: `the castType of cast` *cast member*

This property determines the type of the cast member specified by *cast member*. The result is given as a symbol--a Lingo element that starts with the symbol operator (#).

The possible cast types are:

#bitmap
#button
#digitalVideo
#empty (No cast member is in the specified position)
#filmLoop
#movie
#palette
#picture
#script
#shape
#sound
#soundLevel properties

Example

Lingo example: castType of cast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

The following statement displays the type of cast member number 45, which is a PICT image, in the message window:

```
put the castType of cast 45

-- #picture
```

center of cast (Lingo)

Syntax: `the center of cast` *castName*

This cast property interacts with the `crop of cast` cast property. It can be tested and set.

- When `the crop of cast` is FALSE, `the center of cast` has no effect.
- When `the crop of cast` is TRUE and `the center of cast` is TRUE, cropping occurs around the center of the digital video cast member.
- When `the crop of cast` is TRUE and `the center of cast` is FALSE, cropping starts at the top left corner of the sprite that refers to the digital video cast member.

Example

Related topic: the crop of cast digital video cast property

Lingo example: center of cast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement causes the digital video cast member Interview to be displayed in the top left corner of the sprite:

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

centerStage (Lingo)

Syntax: `the centerStage`

This property determines whether the stage is centered on the monitor when the movie is loaded. The statement that includes this property is placed in the move that precedes the movie you want it to affect.

- If this property is TRUE, the stage is centered.

- If this property is FALSE, the stage is not centered.

This property is useful for checking stage location before a movie plays from a projector. Place handlers that use this property in the preceding movie before using the `go to movie` command.

The `centerStage` property can be tested and set. The default value is TRUE.

Example

Related topic: fixStageSize property

Lingo example: centerStage

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sends the movie to a specific frame if the stage is not centered:

```
if the centerStage = FALSE then ¬
   go to frame "off center"
```

This statement changes the centerStage property to the opposite of its current value:

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

char...of (Lingo)

Syntax: `char` *whichCharacter* `of` *chunkExpression*
`char` *firstCharacter* `to` *lastCharacter* `of` *chunkExpression*

This chunk expression keyword identifies a character or a range of characters in a chunk expression. Chunk expressions are used to refer to any character, word, item, or line in any source of text such as text cast members and variables that hold strings.

- The expression *whichCharacter* identifies a specific character.

- The expressions *firstCharacter* and *lastCharacter* identify a range of characters.

The expressions must be integers that specify a character or range of characters in the chunk. Characters include letters, numbers, punctuation marks, spaces, and control characters like Tab and Return.

You can test and set the `char...of` keyword.

Example

Related topics: mouseCast, mouseItem, mouseLine, and mouseWord integer functions; word...of, item...of, and line...of chunk expression keywords; number of chars in chunk function; and chars, length, and offset functions

Lingo example: char...of

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement displays the first character of the string $9.00:

```
put char 1 of "$9.00"
-- "$"
```

This statement displays the entire string $9.00:

```
put char 1 to 5 of "$9.00"
-- "$9.00"
```

This statement changes the first five characters of the second word of the third line of a text cast member:

```
put "?????" into char 1 to 5 of word 2 of line 3 ¬
   of field "quiz"
```

chars (Lingo)

Syntax: `chars(`*stringExpression* `,` *firstCharacter* `,` *lastCharacter* `)`

This function identifies a substring of characters in *stringExpression*. The substring starts at firstCharacter and ends at *lastCharacter*. The expressions *firstCharacter* and *lastCharacter* must specify a position in the string.

If *firstCharacter* and *lastCharacter* are equal, then a single character is returned from the string. If *lastCharacter* is greater than the string length, only a substring up to the length of the string is identified. If *lastCharacter* is before *firstCharacter*, the function gives the value EMPTY.

Example

Related topics: char...of chunk expression keyword; length and offset functions; number of chars in chunk function

Lingo example: chars

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement identifies the sixth character in the word Macromedia:

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

-- "m"
```

This statement identifies the sixth through tenth characters of the word Macromedia

```
put chars("Macromedia", 6, 10)

-- "media"
```

This statement tries to identify the sixth through twentieth characters of the word Macromedia. Because the word has only ten characters, the result includes only the sixth to tenth characters.

```
put chars ("Macromedia", 6, 20)

-- "media"
```

This statement tests whether the word Macromedia has a twentieth character:

```
if chars ("Macromedia", 20, 20) = EMPTY then put"TRUE"

-- 1
```

charToNum (Lingo)

Syntax: `charToNum(`*stringExpression* `)`

This function identifies the ASCII code number that corresponds to the first character of *stringExpression*.

You can use the Lingo `charToNum` function to test for the ASCII value of characters created with the combination of the Control key and one other alphanumeric key. (When the Control key is pressed, it modifies the ASCII value of the key.)

Related topics: numToChar function

checkBoxAccess (Lingo)

Syntax: `the checkBoxAccess`

This property determines what happens when the user clicks a checkbox or radio button created with button tools in the tools window. There are three possible results:

0--Lets the user set checkboxes and radio buttons on and off (this is the default)

1--Lets the user set checkboxes and radio buttons on but not off

2--Prevents the user from setting checkboxes and radio buttons at all; the buttons can only be set by scripts

The `checkBoxAccess` property can be tested and set. The default value is 0.

Example

Related topics: hilite of cast property; checkBoxType property

Lingo example: checkBoxAccess

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the checkBoxAccess property to 1, which lets the user click checkboxes and radio buttons on but not off:

```
set the checkBoxAccess to 1
```

This statement records the current setting of the checkBoxAccess property by putting the value in the variable oldAccess:

```
put the checkBoxAccess into oldAccess
```

checkBoxType (Lingo)

Syntax: `the checkBoxType`

This property determines what is inserted in checkboxes to indicate they are selected. There are three possible styles:

0--Inserts an "x." This is the default

1--Inserts a black rectangle

2--Inserts a filled black rectangle

The `checkBoxType` property can be tested and set. The default value is 0.

Example

Related topics: hilite of cast property; checkBoxAccess property

Lingo example: checkBoxType

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the checkBoxType property to 1, which shows a black rectangle in checkboxes when the user clicks them:

```
set the checkBoxType to 1
```

This statement records the current setting of the checkBoxType property by putting the value in the variable oldBoxType:

```
put the checkBoxType into oldBoxType
```

checkMark of menuItem (Lingo)

Syntax: `the checkMark of menuItem` *whichItem* `of menu` *whichMenu*

This menu item property determines whether the specified custom menu item is displayed with a checkmark.

- When it is TRUE, a checkmark appears next to the custom menu item.
- When it is FALSE, no checkmark appears.

The *whichItem* expression can be either a menu item name or a menu item number. The *whichMenu* expression can be either a menu name or a menu number.

The `checkMark of menuItem` property can be tested and set. The default value is FALSE.

Example

Related topics: installMenu command; enabled of menuItem, name of menuItem, number of menuItems, and script of menuItem menu item properties; name of menu and number of menus menu item properties; menu keyword

Lingo example: checkMark of menuItem

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler unchecks any items that are checked in the custom menu specified by the argument theMenu. For example, unCheck ("Format") unchecks all the items in the Format menu.

```
on unCheck theMenu
   put the number of menuItems of menu theMenu into n
   repeat with i = 1 to n
      set the checkMark of menuItem i of menu ¬
      theMenu to FALSE
   end repeat
end unCheck
```

clearGlobals (Lingo)

Syntax: `clearGlobals`

This command sets all user-defined global variables to 0.

Example

Lingo example: clearGlobals

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

If you initialize a global variable with a string or value,

```
global foo

put "Director" into foo
```

The global variable foo contains the string Director until another string or value is put into the global, or until the clearGlobals command is issued. This can be useful when initializing global variables, or when opening a new movie that requires a new set of global variables.

After this command is issued, the global variable foo contains 0.

```
clearGlobals
```

clickLoc (Lingo)

Syntax: `the clickLoc`

This function identifies the last place on the screen where the mouse was clicked. The location is given as a `point` function.

Example

Lingo example: clickLoc

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

The following on mouseDown handler displays the last mouse click location:

```
on mouseDown

   put the clickLocend mouseDown
```

clickOn (Lingo)

Syntax: `the clickOn`

This function returns the last active sprite clicked by the user. An active sprite is a sprite that has a sprite script associated with it.

When you want to detect whether the user clicks a sprite with no script, you must assign a dummy script to it ("--" for example) so that it can be detected by the `clickOn`.

To detect a click on the stage, test whether the `clickOn` equals 0.

Related topics: doubleClick, mouseDown, and mouseUp functions

close window (Lingo)

Syntax: `close window` *windowIdentifier*

This command closes the window specified by *windowName*.

- To specify a window by name, use the syntax `close window` "*name* ", where you replace name with the name of a window. Be sure to use the complete path name.

- To specify a window by its number in the `windowList`, use the syntax `close window` *number*, where you replace *number* with the window's number in the window list.

Lingo permits you to attempt to close a window that is already closed.

Example

Related topics: open window command; windowList function

Lingo example: close window

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement closes the window named Panel:

```
close window "Panel"
```

This statement closes the window that is number 5 in the window list:

```
close window 5
```

closeDA (Lingo)

Syntax: `closeDA`

On the Macintosh, this command closes all open desk accessories under System 6.x. System 7.0 and later treat desk accessories like applications, so closeDA has no effect in these versions. Under Windows, this command performs no operation and generates no error message.

Related topic: openDA command

closeResFile (Lingo)

Syntax: `closeResFile [`*whichFile* `]`

On the Macintosh, this command closes the resource file specified by the string expression *whichFile*. When the resource file is in a different folder than the current movie, *whichFile* must specify a pathname. When no file is specified, all open resource files are closed. Under Windows, this command performs no operation and generates no error message.

It is good practice to close any file you have opened as soon as you are finished using it.

Example

Related topics: openResFile and showResFile commands

Lingo example: closeResFile

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement closes all resource files that were opened by the openResFile command:

```
closeResFile
```

This statement closes the file Special Fonts when it is in the same folder as the movie:

```
closeResFile "Special Fonts"
```

This statement closes the file Special Fonts in the folder Special Tools on the same disk as the movie. The disk is identified by the variable currentDrive:

```
closeResFile currentDrive & ¬
   "Special Tools:Special Fonts"
```

closeXlib (Lingo)

Syntax: `closeXlib [`*whichFile* `]`

This command closes the Xlibrary file specified by the string *whichFile* . If the Xlibrary is in another folder than the current movie, *whichFile* must specify either a relative or absolute pathname. If no file is specified, all open Xlibraries are closed.

XObjects, which are extensions to Lingo, are stored in Xlibrary files. Xlibrary files are resource files that contain XCOD (XObjects) resources. HyperCard XCMDs and XFCNs can also be stored in Xlibrary files.

Under Windows, using the .DLL extension for XObjects is optional.

It is good practice to close any file you have opened as soon as you are finished using it.

Related topics: openXlib and showXlib commands

Lingo example: closeXlib

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement closes all open Xlibrary files:

```
closeXlib
```

This statement closes the Xlibrary VdeoDisc when it is in the same folder as the movie:

```
closeXlib "VdeoDisc"
```

This statement closes the Xlibrary TransXOb in the folder or directory New_XOb on the same disk as the movie. The disk is identified by the variable currentDrive:

```
closeXlib currentDrive & ¬
   "New_XOb:TransXOb"
```

colorDepth (Lingo)

Syntax: `the colorDepth`

This property determines the color depth of the computer's monitor.

On the Macintosh, using this property lets you check the color depth of different monitors and change it when appropriate.

Under Windows, using this property lets you check the monitor's color depth, but not change it. (Changing color depth under Windows requires using the System Setup program or installing the proper video card driver.)

Possible values are the following:

1--Black-and-white

2--4 colors

4--16 colors

8--256 colors

16--32,768 colors

32--16,777,216 colors

When you assign a monitor a `colorDepth` higher than the monitor's highest available color depth, the monitor becomes set to its maximum color depth.

On Macintosh computers with more than one monitor, `the colorDepth` property refers to the monitor that the stage is on. If the stage spans more than one monitor, `the colorDepth` indicates the greatest depth of those monitors; setting `the colorDepth` attempts to put all those monitors to the specified depth.

The `colorDepth` property can be tested and set. The default value is the value set in the Monitors control panel.

Example

Related topics: colorQD function; switchColorDepth property

Lingo example: colorDepth

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement makes playing the movie "AllColor" dependent on whether the monitor color depth is set to 256 colors:

```
if the colorDepth = 8 then play movie "AllColor"
```

This statement uses the colorQD function to check whether the monitor can display color, and then sets the color depth to 256 colors if it is:

```
if the colorQD = TRUE then set the colorDepth to 8
```

colorQD (Lingo)

Syntax: `the colorQD`

This function indicates whether the Color QuickDraw software is available on a Macintosh.

- The `colorQD` is TRUE (1) for a color-capable Macintosh. (For any computer under Windows, `the colorQD` is always TRUE regardless of what is on the computer.)

- The `colorQD` is FALSE (0) for a black-and-white-only Macintosh.

Note: A machine capable of displaying color may not have it switched on. This command is best used in conjunction with `colorDepth`.

Example

Related topics: colorDepth and switchColorDepth properties

Lingo example: colorQD

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether the Macintosh is color capable and plays the movie "Color Movie" if it is:

```
if the colorQD = TRUE then play movie "Color Movie"
```

This statement checks whether the Macintosh is color capable and sets the color depth to 256 colors if it is:

```
if the colorQD = TRUE then set the colorDepth to 8
```

commandDown (Lingo)

Syntax: `the commandDown`

This function determines whether the `Command` key is being pressed on the Macintosh or the Ctrl key is being pressed under Windows.

- The `commandDown` function is TRUE when the Command key is being pressed on the Macintosh or the Ctrl key is being pressed under Windows.

- The `commandDown` function is FALSE when the Command key is not being pressed on the Macintosh or the Ctrl key is not being pressed under Windows.

You can use the `commandDown` together with the element `the key` to determine when the Macintosh's Command key or the PC's Control key is pressed in combination with another key. This lets you create handlers that are executed when the user presses specified Command-key or Control-key combinations.

Note that Command key and Control-key equivalents for Director's authoring menus take precedence while playing the movie, unless you have installed custom Lingo menus, or are playing a projector version of the movie.

Example

Related topics: controlDown, key, keyCode, optionDown, and shiftDown functions

Lingo example: commandDown

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

These statements have Lingo pause the movie whenever the user presses Command-p on the Macintosh or Control-p under Windows. By setting the keyDownScript property to doCommandKey, the on startMovie handler makes the doCommandKey handler the first event handler executed when a key is pressed. The doCommandKey handler checks whether the Command-p or Control-p is pressed at the same time and pauses the movie if it is.

```
on startMovie

   set the keyDownScript to "doCommandKey"

end startMovie

on doCommandKey

   if (the commandDown) and (the key = "p") then pause

end
```

constrainH (Lingo)

Syntax: `constrainH (`*whichSprite* , *integerExpression*)

This function evaluates *integerExpression*, and then gives a value that depends on the horizontal coordinates of the left and right edges of *whichSprite*.

- When the value is between the left and right coordinates, the value doesn't change.
- When the value is less than the left horizontal coordinate, the value is changed to the value of the left coordinate.
- When the value is greater than the right horizontal coordinate, the value is changed to the value of the right coordinate.

The `constrainH` and `constrainV` functions constrain only one axis each; the `constraint of sprite` property limits both. Note that this function does not change the sprite's properties.

Example

Related topics: constrainV function; constraint of sprite, left, and right sprite properties

Lingo example: constrainH

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

These statements check the constrainH for sprite 1 when it has left and right coordinates of 40 and 60:

```
put constrainH(1, 20)

-- 40

put constrainH(1, 55)

-- 55

put constrainH(1, 100)

-- 60
```

This statement constrains a moveable slider (sprite 1) to the edges of a gauge (sprite 2) when the mouse pointer goes past the edge of the gauge:

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

constraint of sprite (Lingo)

Syntax: `the constraint of sprite` *whichSprite*

This sprite property determines the constraints on the position of the sprite specified by *whichSprite*. When `the constraint of sprite` property is turned on, the sprite specified by *whichSprite* is constrained to the bounding rectangle of another sprite.

The `constraint of sprite` property affects moveable sprites, and the `locH` and `locV` sprite properties. The constraint point of a moveable sprite cannot be moved outside the bounding rectangle of the constraining sprite. (The constraint point for a bitmap sprite is the registration point. The constraint point for a shape sprite is the top left corner of the shape sprite.) When a sprite has a constraint set, the constraint limits override any `locH` and `locV` sprite property settings.

To remove a constraint of sprite property, set it to 0:

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

The `constraint of sprite` property can be tested and set. The default value is 0.

The `constraint of sprite` property is useful for constraining a moveable sprite to the bounding rectangle of another sprite. This is a way to simulate a "track" for a slider control or restrict where on the screen a user can drag an object in a game.

Example

Related topics: constrainH and constrainV functions; locH and locV sprite properties

Lingo example: constraint of sprite

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement constrains sprite (i + 1) to the boundary of sprite 14.

```
set the constraint of sprite (i + 1) to 14
```

This statement checks whether sprite 3 is constrained and activates the handler showConstraint if it is. (The operator <> performs the same operation as "not equal to.")

```
if the constraint of sprite 3 <> 0 then ¬
   showConstraint
```

constrainV (Lingo)

Syntax: `constrainV (`*whichSprite* , *integerExpression*)

This function evaluates *integerExpression*, and then gives a value that depends on the vertical coordinates of the top and bottom edges of the sprite specified by *whichSprite*.

- When the value is between the top and bottom coordinates, the value doesn't change.
- When the value is less than the top coordinate, the value is changed to the value of the top coordinate.
- When the value is greater than the bottom coordinate, the value is changed to the value of the bottom coordinate.

Note that this function does not change the sprite properties.

Example

Related topics: bottom of sprite, constraint of sprite, and top of sprite sprite properties; constrainH function

Lingo example: constrainV

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

These statements check the constrainV for sprite 1 when it has top and bottom coordinates of 40 and 60:

```
put constrainV(1, 20)

-- 40

put constrainV(1, 55)

-- 55

put constrainV(1, 100)

-- 60
```

This statement constrains a moveable slider (sprite 1) to the edges of a gauge (sprite 2) when the mouse pointer goes past the edge of the gauge:

```
set the locV of sprite 1 to ¬ constrainV(2, the mouseV)
```

contains (Lingo)

Syntax: *stringExpression1* contains *stringExpression2*

This operator compares two strings.

- When *stringExpression1* contains *stringExpression2*, the condition is TRUE (1).

- When *stringExpression1* does not contain *stringExpression2*, the condition is FALSE (0).

The `contains` comparison operator has a precedence level of 1.

The `contains` comparison operator is useful for checking whether the user types a specific character or string of characters. You can also use the contains operator to search one or more text fields for specific strings of text.

Example

Note: The string comparison is not sensitive to case or diacritical marks; "a" and "Å" are treated the same.

Related topics: offset function; starts comparison operator

Lingo example: contains

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement determines whether a string from a text field contains only numeric input before converting it using the value() function. You can use this handler to test it:

```
on isNumber aLetter
   put "1234567890." into digits
   if digits contains aLetter then
      return TRUE
   else
      return FALSE
   end if
end isNumber
```

continue (Lingo)

Syntax: `continue`

The `continue` command resumes playing the movie after a pause.

Example

Related topics: delay and pause commands; pauseState function

Lingo example: continue

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement has the movie resume playing when it is paused and the Return key is pressed:

```
set the keydownScript to "if the key = RETURN ¬
   then continue"
```

controlDown (Lingo)

Syntax: `the controlDown`

This function determines whether the Control key on the Macintosh or the Ctrl key on the PC is being pressed.

- The `controlDown` function is TRUE when the Control key is being pressed.
- The `controlDown` function is FALSE when the Control key is not being pressed.

You can use the `controlDown` function together with the element the key to check for combinations of the Control key and another key.

Related topics: charToNum, commandDown, key, keyCode, optionDown, and shiftDown functions

Lingo example: controlDown

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This keyDown handler checks whether the key that is pressed is the Control or Ctrl key and activates the doControlKey handler if it is. The argument the key identifies which key was pressed in addition to the Control or Ctrl key.

```
on keyDown

   if the controlDown then doControlKey (the key)

end
```

controller of cast (Lingo)

Syntax: `the controller of cast` *castName*

A digital video movie cast member can be made to show or hide its controller with this cast property. Setting this property to 1 shows the controller; setting it to 0 hides it.

The `controller of cast` property applies to QuickTime and QuickTime for Windows movies only. Setting `controller of cast` for a Video for Windows movie performs no operation and generates no error message. Checking the `controller of cast` for a Video for Windows movie always returns FALSE.

The digital video movie must be in `directToStage` playback mode in order to display the controller. (Under Windows, QuickTime movies are always in `directToStage` playback mode. Therefore, you can use `controller of cast` any time under Windows.)

Example

Related topic: directToStage cast property

Lingo example: controller of cast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement has the digital video cast member Demo show its controller:

```
set the controller of cast "Demo" to 1
```

copyToClipBoard (Lingo)

Syntax: `copyToClipBoard cast` *castMember*

This command copies the specified cast member to the Clipboard.

You can use this command to copy cast members between movies or applications. The cast window does not need to be the active window when you use the `copyToClipBoard` command.

Example

Lingo example: copyToClipBoard

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement copies the cast member named chair to the Clipboard:

```
copyToClipBoard cast "chair"
```

This statement copies cast member number 5 to the Clipboard:

```
copyToClipBoard cast 5
```

cos (Lingo)

Syntax: `cos (angle )`

This function calculates the cosine of the specified angle. The angle must be expressed in radians.

Example

Lingo example: cos

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

The following statement calculates the cosine of pi ()/2 and displays it in the message window:

```
put cos (pi()/2)
```

Note that the ¹ symbol cannot be used in a Lingo expression.

count (Lingo)

Syntax: `count (`*list*`)`

This function returns the number of entries in the specified list.

Example

Lingo example: count

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement displays the number 3, the number of entries:

```
put count ( [10, 20, 30] )
-- 3
```

crop of cast (Lingo)

Syntax: `the crop of cast`

This cast property affects how the digital video cast member is displayed inside a sprite. It can be tested and set.

- When the `crop of cast` is FALSE the cast member is scaled--either stretched or shrunk--to fit inside the sprite rectangle.

- When the `crop of cast` is TRUE, the cast member is not scaled. It is cropped to fit inside the sprite rectangle.

Example

Related topic: center of cast digital video cast property

Lingo example: crop of cast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement instructs Lingo to crop any sprite that refers to the digital video cast member Interview.

```
set the crop of cast "Interview" to TRUE
```

cursor (Lingo)

Syntax: `cursor [`*castNumber* `,` *maskCastNumber* `]`

or

`cursor` *whichCursor*

This command changes the cast member that is used for a cursor. The cursor command stays in effect until you turn it off by setting the cursor to zero.

- Use the syntax `cursor` [*castNumber* , *maskCastNumber*] to specify the number of a cast member to use as a cursor and its optional mask. The hot spot of the cursor is the registration point of the cast member.

  The cast member that you specify must be a 1-bit cast member; it must be at least 16 by 16 pixels. If the cast member is larger, Director crops it to a 16 x 16 square, starting in the upper left corner of the image. If the cast member is smaller, draw a 17 x 17 square around it to achieve the proper dimensions when the image is cropped.

- Use the syntax `cursor` *whichCursor* to use the default cursors that are supplied by the system.

  The term *whichCursor* must be an integer that specifies the appearance of the cursor. The following values specify cursors:

  0 -- no cursor set
  -1 -- arrow (pointer) cursor
  1 -- I-beam cursor
  2 -- crosshair cursor
  3 -- crossbar cursor
  4 -- watch or hourglass cursor
  200 -- blank cursor

  To hide the cursor, set the cursor to 200 (a blank cursor).

Notes:

Do not confuse `cursor [1]` with `cursor 1`. The first uses castmember 1 as a custom cursor; the second selects the I-beam from the system cursor set.

Under Windows, a cursor can't be a resource; it must be a cast member. If a cursor isn't available under Windows because it hasn't been converted from a resource to a cast member, Lingo uses the standard arrow cursor instead.\It is recommended that you don't make custom cursors resources when you create movies that you intend to play on both the Macintosh and Windows.

During system events such as loading a file, the operating system may put up the watch cursor, and then change to the pointer cursor when returning control to the application. This overrides the `cursor` command settings from the previous movie. Therefore, in a presentation using a custom cursor for multiple movies, store any special cursor resource number as a global variable. Global Lingo variables stay in memory between movies. This allows you to use the `cursor` command at the beginning of any new movie that is loaded.

Example

Related topics: cursor of sprite property; openResFile command; rollOver function

Lingo example: cursor

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

When the movie plays on the Macintosh, this statement changes the cursor to a watch cursor whenever the value in the variable named status equals 1:

```
if status = 1 then cursor 4
```

cursor of sprite (Lingo)

Syntax: `the cursor of sprite` *whichSprite* `to [`*castNumber*`,` *maskCastNumber* `]`

`the cursor of sprite` *whichSprite* `to` *whichCursor*

This sprite property determines the cursor resource that is used when the pointer is over the sprite specified by the integer expression *whichSprite*. The cursor property stays in effect until you turn it off by setting the cursor to zero.

Note: Under Windows, a cursor can't be a resource; it must be a cast member. If a cursor isn't available under Windows because it hasn't been converted from a resource to a cast member, Lingo uses the standard arrow cursor instead.\It is recommended that you don't make custom cursors resources when you create movies that you intend to play on both the Macintosh and Windows.

The cursor property is an integer that specifies the resource ID number of the cursor. The following cursors are always available:

0 -- no cursor set
-1 -- arrow (pointer) cursor
1 -- I-beam cursor
2 -- crosshair cursor
3 -- crossbar cursor
4 -- watch or hourglass cursor
200 -- blank cursor

To hide the cursor, set `the cursor` to 200 (a blank cursor resource).

The `cursor of sprite` property is useful for changing the cursor when the mouse pointer is over specific regions of the screen. You can use this to indicate regions where certain actions are possible when the user clicks.

On the Macintosh, you can use a numbered cursor resource in the current open movie file as the cursor by replacing whichCursor with the number of the cursor resource.

See the cursor command for information about using custom cursors.

The `cursor of sprite` property can be tested and set.

Example

Related topics: cursor and openResFile commands

Lingo example: cursor of sprite

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement executes the handler setCursor when no cursor is set for sprite 3:

```
if the cursor of sprite 3 = 0 then setCursor
```

This statement sets the cursor to an I-beam when the cursor is over sprite 3:

```
if rollover(3) then set the cursor of sprite 3 to 1
```

This statement sets the cursor to a custom cursor named grabber:

```
set the cursor of sprite (i + 1) to grabber
```

date (Lingo)

Syntax:
```
the abbr date
the abbrev date
the abbreviated date
the date
the long date
the short date
```

This function gives the current date in the system clock in one of three formats: `abbreviated`, `long`, or `short`. If no format is specified, the default is short. The abbreviated format can also be referred to as `abbrev` and `abbr`.

Example

Note: The three date formats vary, depending on the country for which your system software was designed. These examples are for the United States.

Related topics: time function

Lingo example: date

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement gives the abbreviated date:

```
put the abbreviated date
-- "Sat, Sep 7, 1991"
```

This statement gives the long date:

```
put the long date
-- "Saturday, September 7, 1991"
```

This statement gives the short date:

```
put the short date
-- "9/7/91"
```

This statement tests whether the current date is January 1 by checking whether the first four characters of the date are 1/1. If it is January 1, the alert "Happy New Year!" appears:

```
if char 1 to 4 of the date = "1/1/" ¬
   then alert "Happy New Year!"
```

delay (Lingo)

Syntax: `delay` *numberOfTicks*

This command halts the movie for a given amount of time. The integer expression *numberOfTicks* specifies the number of ticks to wait. (There are 60 ticks per second.) The only interactivity possible during this time is halting Lingo entirely by pressing Control-Period (for example, mouse clicks are ignored).

The `delay` command works only when the playback head is moving. Place scripts using the delay command in either an `on enterFrame` or `on exitFrame` handler.

To mimic the behavior of a halt in a handler when the playback head is not moving, use the `startTimer` command and test for the timer property within a `repeat... while` loop.

Because it increases the time of individual frames, the `delay` command is useful for controlling the playback rate of a sequence of frames.

Example

Note: The `delay` command does not function when the playback head is not moving.

Related topics: startTimer command; timer property

Lingo example: delay

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler delays the movie for 2 seconds when the playback head exits the current frame:

```
on exitFrame
   delay 2 * 60
end exitFrame
```

This handler, which could be placed in a frame script, delays the movie a random number of ticks:

```
on keydown
   if the key = RETURN then delay random(180)
end keyDown
```

delete (Lingo)

Syntax: `delete` *chunkExpression*

This command deletes the specified chunk expression (character, word, item, or line) in any text container. Sources of text include fields (text cast members) and variables that hold strings.

Example

Related topics: char...of, field, item...of, line...of , word...of, chunk expression keywords; hilite text property

Lingo example: delete

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement deletes the first word of line 3 in the text cast member Address:

```
delete word 1 of line 3 of field "Address"
```

This statement deletes the first character of the string in the variable bidAmount:

```
if char 1 of bidAmount = "$" then delete char 1 ¬
   of bidAmount
```

deleteAt (Lingo)

Syntax: `deleteAt` *list* , *number*

This command deletes the item in the position specified by number from the list specified by *list*. The value *number* is the item's position in the order of the list.

Example

Related topic: addAt command

Lingo example: deleteAt

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement deletes the second item from the list named designers, which contains ["gee", "kayne", "ohashi"]:

```
set designers = ["gee", "kayne", "ohashi"]

deleteAt designers, 2
```

The result is the list ["gee", "ohashi"].

deleteProp (Lingo)

Syntax: `deleteProp` *list*, *property*

This command deletes the item that has the specified property from the specified list. For linear lists, this is the same as the `deleteAt` command. When there are more than one of the same property, only the first property in the list is deleted.

Example

Related topic: deleteAt command

Lingo example: deleteProp

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement deletes the property color from the list [#height: 100, #width: 200, #color: 34, #ink: 15], which is called spriteAttributes:

```
deleteProp spriteAttributes, #color
```

The result is the list [#height: 100, #width: 200, #ink: 15].

depth of cast (Lingo)

Syntax: `the depth of cast` *cast member*

This cast property gives the color depth of the bitmap cast member specified by *cast member*. Black and white is 1-bit color depth; four colors is 2-bit color depth; 256 colors is 8-bit color depth; thousands of colors is 16-bit color depth; and millions of colors is 24-bit or 32-bit color depth.

This property can be tested, but not set from Lingo.

Example

Lingo example: depth of cast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement determines the color depth of the cast member Shrine:

```
put the depth of cast "Shrine"
```

digitalVideo (Lingo)

See: center of cast
controller of cast
crop of cast
directToStage
duration of cast
frameRate of cast
loop of cast
movieRate of cast
movieTime of cast
pausedAtStart
preload of cast
sound of cast
startTime of cast
stopTime of cast
video of cast
volume of cast

directToStage (Lingo)

Syntax: `the directToStage of cast` *castName*

This property determines whether a digital video movie cast member plays in front of all other layers on the stage.

- When this property is set to TRUE (1), a digital video movie plays in front of all other layers. (Checking the `directToStage of cast` for a digital video movie always returns TRUE.)

- When this property is set to FALSE (0), a digital video movie cast member can appear in any layer of the stage's animation planes. (Under Windows, the `directToStage of cast` property is always TRUE. Setting the `directToStage of cast` to FALSE has no effect under Windows.)

No cast members appear in front of a `directToStage` digital video movie. Also, ink effects do not affect the appearance of a `directToStage` digital video movie. Using this property can significantly improve the playback performance of a digital video movie cast member.

do (Lingo)

Syntax: `do` *stringExpression*

This command evaluates *stringExpression* and executes the result as a Lingo statement. This command is useful for evaluating expressions that the user has typed, and for executing commands stored in string variables, fields, arrays, and files.

Note: This command does not allow global variables to be declared. In earlier versions of Director, the statement would work. In Director 4, they do not.

Example

Lingo example: do

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement performs the statement contained within quotes:

```
do "beep 2"
```

```
do getAt(commandList, 3)
```

dontPassEvent (Lingo)

Syntax: `dontPassEvent`

This command prevents Lingo from passing an event message to subsequent locations in the message hierarchy.

The `dontPassEvent` command applies only to the current event being handled. It does not affect future events.

The `dontPassEvent` command applies only within primary event handlers or handlers that they call. It has no effect elsewhere.

The `dontPassEvent` command only applies within event scripts (those starting with the word "when..."), or handlers called by an event script as in the example above. It does not have any effect elsewhere.

The `dontPassEvent` command only applies to the current event being handled. It does not affect future events.

Example

Lingo example: dontPassEvent

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler has the computer beep when the Tab or Enter key is pressed and keeps the message from passing on to subsequent locations in the message hierarchy:

```
on myKey

   if the key = TAB or the key = ENTER then beep

   dontPassEvent

end myKey
```

This statement makes myKey the primary event handler:

```
set the keyDownScript to "myKey"
```

When these are in effect at the same time, pressing the Tab or Enter key any time the movie is playing has the computer beep but not pass the keyDown message on to anywhere else in the movie.

doubleClick (Lingo)

Syntax: `the doubleClick`

This function determines whether the last two mouse clicks were considered a double-click.

- The `doubleClick` function is TRUE if the last two mouse clicks were a double-click.
- The `doubleClick` function is FALSE if the last two mouse clicks were not a double-click.

Example

Related topics: clickOn, mouseDown, mouseUp functions

Lingo example: doubleClick

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sends the playback head to the frame EnterBid when the user double-clicks the mouse button.

```
if the doubleClick then go to frame "EnterBid"
```

drawRect of window (Lingo)

Syntax: `the drawRect of window` *windowName*

This window property identifies the rectangular coordinates of a movie's window. The coordinates are given as a rect, with entries in the order left, top, right, and bottom.

This can be useful for scaling or panning movies.

The `drawRect of window` property can be tested or set.

Example

Lingo example: drawRect of window

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement displays the current coordinates of the movie window called Control Panel.

```
put the drawRect of window "Control Panel"
-- rect(10, 20, 200, 300)
```

This statement sets the coordinates of the movie window to the values of the rect movieRectangle:

```
set the drawRect of window "Control Panel" ¬
   to movieRectangle
```

duplicate cast (Lingo)

Syntax: `duplicate cast` *original* `[,` *new* `]`

This command makes a copy of the cast member specified by *original*. The optional *new* parameter specifies a specific cast window location for the duplicate cast member. Any cast member already in that position is replaced. If the *new* parameter isn't included, the duplicate cast member is placed in the first open cast window position.

Example

Lingo example: duplicate cast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement makes a copy of cast member Desk and places it in the first empty cast window position:

```
duplicate cast "Desk"
```

This statement makes a copy of cast member Desk and places it in cast window position 125:

```
duplicate cast "Desk", cast 125
```

duration of cast (Lingo)

Syntax: `the duration of cast` *castName*

This cast property is used with digital video cast members to determine the duration of the movie. This property cannot be set. The value returned for the movie's duration is in ticks (60ths of a second.)

Example

Lingo example: duration of cast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the variable HowLong to the duration of the digital video cast member Demo:

```
put the duration of cast "Demo" into HowLong
```

editableText of sprite (Lingo)

Syntax: `the editableText of sprite` *whichSprite*

This sprite property indicates whether a text sprite is editable.

- When text can be edited by the user, the `editableText of sprite` is TRUE.
- When the text cannot be edited by the user, the `editableText of sprite` is FALSE.

To use Lingo to make a text sprite editable, the sprite must first be a puppet sprite.

The `editableText of sprite` property lets you change whether text field can be edited as the movie plays. This lets you turn editable text on and off depending on current conditions in the movie.

You can also make a text cast member editable by using the Editable Text option in the Text Cast Member Info dialog box. You can make a text sprite editable by using the Editable option in the score.

The `editableText of sprite` property can be tested and set.

Example

Lingo example: editableText of sprite

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler first makes the text sprite a puppet and then makes it editable:

```
on myNotes
   puppetSprite 5, TRUE
   set the editableText of sprite 5 to TRUE
end
```

This statement checks whether a text sprite is editable and displays a message if it is:

```
if the editableText of sprite 13 = TRUE ¬
   then set the text of cast "Notice" to  "Please ¬
   enter your answer below."
```

EMPTY (Lingo)

Syntax: `EMPTY`

This character constant represents the empty string, "", a string with no characters.

You can scroll to a specific line in a scrolling text field by inserting `EMPTY` before the line.

Example

Lingo example: EMPTY

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement erases all characters in the text cast member Notice by setting the field to EMPTY:

```
set the text of cast "Notice" to EMPTY
```

This statement does the same as the previous statement but uses a different form:

```
put EMPTY into field "Notice"
```

enabled of menuItem (Lingo)

Syntax: `the enabled of menuItem` *whichItem* `of menu` *whichMenu*

This menu item property determines whether the menu item specified by *whichItem* is displayed in plain text or dimmed, and whether it is selectable. The term *whichMenu* specifies the menu that contains the menu item.

- If the `enabled of menuItem` is TRUE, the menu item appears in plain text and is selectable.

- If the `enabled of menuItem` is FALSE, the menu item appears dimmed and is not selectable.

The expression *whichItem* can be either a menu item name or a number that indicates the item's place in the menu. (The first menu item is number 1, and so on.) The expression *whichMenu* can be either a menu name or the number that indicates the menus place from the left of the menu bar. (The first menu on the left is menu number 1, and so on.)

The `enabled` property can be tested and set. The default value is TRUE.

Related topics: name of menu, number of menus, checkMark of menuItem and script of menuItem, number of menuItems properties

Lingo example: enabled of menuItem

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler enables or disables all the items in the specified menu. The argument theMenu specifies the menu; the argument Setting specifies TRUE or FALSE. For example, ableMenu ("Special", FALSE) disables all the items in the Special menu:

```
on ableMenu theMenu, vSetting

   put the number of menuItems of menu theMenu into n

      repeat with i = 1 to n

         set the enabled of menuItem i of menu theMenu ¬

            to vSetting

      end repeat

end ableMenu
```

end (Lingo)

This keyword marks the end of handlers, methods, and multi-line control structures.

Related topics: if...then, method, on, repeat while, and repeat with keywords; on mouseDown, on mouseUp, on keyDown, on startMovie, on stepMovie, on stopMovie, and on idle event handlers.

Lingo example: end

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

ENTER (Lingo)

Syntax: `ENTER`

This character constant represents the Enter key on the number pad. (Some PC keyboards also label the key that enters a carriage return as Enter, but the element `ENTER` only refers to the Enter key on the number pad.)

Example

Lingo example: ENTER

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

```
on keyDown
   if the key = ENTER then go to frame "addSum"
end
```

enterFrame (Lingo)

See on enterFrame event handler.

erase cast (Lingo)

Syntax: `erase cast` *whichCastmember*

This command deletes the specified cast member and leaves its cast window location empty.

Example

Lingo example: erase cast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement deletes the cast member named Gear:

```
erase cast "Gear"
```

This handler deletes cast members start through finish:

```
on deleteCast start, finish
   repeat with i = start to finish
      erase cast i
   end repeat
end
```

exit (Lingo)

Syntax: `exit`

This keyword has Lingo leave a handler or factory method, and return to the place from where the handler or factory was called. When the handler was called from another handler, Lingo returns to the handler that did the calling.

Example

Related topics: abort command; on and method keywords

Lingo example: exit

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

The first statement of this script checks whether the monitor is set to black and white, and exits if it is:

```
on setColors
   if the colorDepth = 1 then exit
   set the foreColor of sprite 1 to 35
end setColors
```

exit repeat (Lingo)

Syntax: `exit repeat`

This keyword has Lingo leave a repeat loop and go to the statement following the `end repeat` statement, but remain within the current handler or method.

The `exit repeat` keyword is useful for breaking out of a repeat loop when a specified condition--such as two values being equal or a variable being a certain value--exists.

Example

Related topics: repeat while and repeat with keywords

Lingo example: exit repeat

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler looks for the position of a letter in a string represented by the variable testString. As soon as the letter is found, the exit repeat command has Lingo leave the repeat loop and go to the statement return i:

```
on findVowel testString letter
   repeat with i = 1 to the number of chars ¬
      in testString
      if testString contains letter then exit repeat
   end repeat
   return i
end
```

exitLock (Lingo)

Syntax: `the exitLock`

This property determines whether the user can quit to the Finder or the Program Manager from projectors.

- When `the exitLock` is FALSE, the user can quit to the Finder or Program Manager by pressing the appropriate keys. On the Macintosh, pressing Command-period, Command-q or Command-w quits to the Finder. Under Windows, pressing Control-period or Esc quits to the Program Manager.

- When `the exitLock` is TRUE, the user cannot quit to the Finder or Program Manager by pressing Command-period, Command-q, Command-w, Control-period, or Esc.

The `exitLock` property can be tested and set. The default value is FALSE.

Example

Lingo example: exitLock

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the exitLock property to TRUE:

```
set the exitLock to TRUE
```

This handler checks whether Control-period or Control-q was pressed and whether the exitLock is set so that the user cannot exit to the Program Manager when the movie plays on Windows. When this is the case, the playback head goes to the frame "quit sequence," which could provide an alternative way to exit the movie:

```
on checkExit
   if the commandDown and ¬
   (the key = "." or the key = "q") and ¬
   the exitLock = TRUE then go to frame "quit¬
 sequence"
end checkExit
```

exp (Lingo)

Syntax: `exp(`*integer* `)`

This function calculates e, the natural logarithm base, to the power specified by *integer*.

Example

Lingo example: exp

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

factory (Lingo)

Syntax (as a keyword):

```
factory factoryName
method methodName1
    [statements ]
    end methodName1
method methodName2
    [statements ]
    end methodName2
[more methods ]
```

This keyword defines a factory. A factory is composed of a related group of procedures, called methods, that can be used to create objects, send messages to other objects, and process messages. A factory is used to create internal objects that have the same message syntax as XObjects.

In Director 4, it is recommended that you use lists and parent scripts rather than factories. Lists and parent scripts are a simpler way of achieving the same result.

See Appendix C in the *Using Lingo* manual for more information about factories, objects, methods, and instance variables.

Syntax (as a function): `factory(factoryName )`

This function identifies the factory or XObject specified by *factoryName* . If no factory or XObject with the given name is found, the factory function value is 0. If the factory is found, the object is returned. You can use the `objectP()` function to test the return value.

Example

Related topics: factory, instance, me, and method, keywords; mDispose, mGet, mInstanceRespondsTo, mNew, mPerform, mPut, and mRespondsTo predefined methods

Lingo example: factory (as a function)

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

The three statements

```
put "AppleCD" into playerName

put factory(playerName) into playerFactory

put playerFactory(mNew) into cdPlayer
```

are equivalent to

```
put AppleCD(mNew) into cdPlayer
```

but allow the XObject's name to be easily changed under Lingo control.

FALSE (Lingo)

Syntax: `FALSE`

This logical constant applies to an expression that is logically FALSE, such as 2 > 3. When treated as a number value, FALSE has the numerical value of 0.

Example

Related topics: TRUE logical constant

Lingo example: FALSE

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement turns off the soundEnabled property by setting it to FALSE:

```
set the soundEnabled to FALSE
```

field (Lingo)

Syntax: `field` *whichField*

This keyword refers to the text in a text cast member. It is equivalent to the `text of cast` property.

The text cast member is specified by whichField.

- When *whichField* is a string, it is used as the cast name.
- When *whichField* is an integer, it is used as the cast number.

Text can be read from or put into the field. You can also use chunk expressions with text fields.

Example

Related topics: cast keyword; char...of, item...of, line...of, and word...of chunk expression keywords

Lingo example: field

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement puts the characters 5 through 10 of the field name entry into the variable myKeyword:

```
put char 5 to 10 of field "entry" into myKeyword
```

This statement checks whether the user entered the word "desk" and goes to the frame "deskBid" if he or she did:

```
if field "bid" contains "desk" then go to "deskBid"
```

fileName of cast (Lingo)

Syntax: `the fileName of cast` *cast member*

This cast property refers to the name of the file assigned to the linked cast member specified by *cast member*. This is useful for switching which external linked file is assigned to a cast member while the movie plays, similar to the way you can switch cast members. When the linked file is in a different folder than the movie, you must include the file's pathname.

The `fileName of cast` property can be tested and set. After the filename is set, Director uses that file the next time the cast member is used.

Example

Lingo example: fileName of cast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement makes the digital video movie "ChairAnimation" the linked file assigned to cast member 40:

```
set the fileName of cast 40 = ChairAnimation
```

fileName of window (Lingo)

Syntax: `the fileName of window` *whichWindow*

This window property refers to the filename of the movie assigned to the window specified by *whichWindow*. When the linked file is in a different folder than the movie, you must include the file's pathname.

You assign a movie to a window by setting the fileName of window for the window to the movie's filename. This is required before you can play the movie in the window.

The `fileName of window` property can be tested and set.

Example

Lingo example: fileName of window

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement changes the file assigned to the window from Tool Box to Control Panel:

```
set the fileName of window "Tool Box" to ¬
   "Control Panel"
```

This statement displays the filename of the file assigned to the window named Navigator:

```
put the fileName of window "Navigator"
```

findEmpty (Lingo)

Syntax: `findEmpty(cast` *castNum* `)`

This function returns the next empty cast position after and including the specified *castNum*.

Example

Lingo example: findEmpty

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement finds the first empty cast member on or after cast member 100:

```
put findEmpty(cast 100)
```

findPos (Lingo)

Syntax: `findPos(`*list*`, `*prop*`)`

This command identifies which position the property specified by *property* holds in the property list specified by list.

The `findPos` command does the same thing as the `findPosNear` command, except that the result of the `findPos` command is <VOID> when the specified property is not in the list.

Example

Related topic: findPosNear command

Lingo example: findPos

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement identifies the position of the property c in the list Answers, which consists of [a:#10, b:#12, c:#15, d:#22]:

```
findPos(Answers, #c)
```

The result is 3, because c is the third property in the list.

findPosNear (Lingo)

Syntax: `findPosNear(`*list* `,` *prop* `)`

This command identifies which position the property specified by *property* holds in the property list specified by *list*.

The `findPosNear` command does the same thing as the `findPos` command, except that when the specified property is not in the list, the `findPosNear` command identifies the position of the closest property in the list, based on the sort order. This would be useful in finding the closest name in a sorted directory of names.

Example

Related topic: findPos command

Lingo example: findPosNear

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement identifies the position of a property in the sorted list Answers, which consists of [#Nile:2, #Pharaoh:4, #Raja:0]:

```
findPosNear(Answers, #Ni)
```

The result is 1, because Ni is closest to Nile, the first property in the list.

fixStageSize (Lingo)

Syntax: `the fixStageSize`

This property determines whether the stage size remains the same when you load a new movie, regardless of the stage size saved with that movie. When the `fixStageSize` property is TRUE, the stage size remains the same when you load a new movie.

The `fixStageSize` property cannot change the stage size for a movie that is currently playing. This property is primarily used for movies played back with the player.

The `fixStageSize` property can be tested and set. The default value is TRUE.

Example

Related topics: centerStage property

Lingo example: fixStageSize

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement determines if the fixStageSize property is turned on, and sends the playback head to a specified frame if it is:

```
if the fixStageSize = TRUE then ¬
   go to frame "proper size"
```

This statement sets the fixStageSize property to the opposite of its current setting:

```
set the fixStageSize to (not the fixStageSize)
```

float (Lingo)

Syntax: `float (`*expression*`)`

Converts an expression to a floating-point number. The number of digits that follow the decimal point is set using the `floatPrecision` property.

Example

Related topic: floatPrecision property

Lingo example: float

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement converts the integer 1 to floating-point 1.

```
put float(1)

-- 1.0
```

floatP (Lingo)

Syntax: `floatP(`*expression* `)`

This function indicates whether the value specified by expression is a floating-point number.

- The `floatP` function is TRUE (1) if *expression* is a floating-point number.
- The `floatP` function is FALSE (1) if *expression* is not a floating-point number.

The "P" in `floatP` stands for "predicate."

Example

Related topics: float, integerP, objectP, stringP, and symbolP functions

Lingo example: floatP

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement tests whether 3.0 is a floating-point number. The message window displays the number 1, indicating that it is TRUE:

This statement tests whether 3 is a floating-point number. The message window displays the number 0, indicating that it is FALSE:

```
put floatP(3)

-- 0
```

floatPrecision (Lingo)

Syntax: `the floatPrecision to` *integer*

This system property rounds off the display of floating-point numbers to the number of decimal places specified by *integer*. The maximum is 19 significant digits.

The `floatPrecision` property determines only the number of digits used to display floating-point numbers. The number of digits used to perform calculations doesn't change.

The `floatPrecision` property can be tested and set. The default value is 4.

Example

Lingo example: floatPrecision

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement rounds off the square root of 3.0 to three decimal places:

```
set the floatPrecision to 3put sqrt(3.0) into xput x
-- 1.732
```

This statement rounds off the square root of 3.0 to eight decimal places:

```
set the floatPrecision to 8put x
-- 1.73205081
```

foreColor of cast (Lingo)

Syntax: `set the foreColor of cast` *castName* `to` *colorNumber*

This cast property sets the foreground color of a text cast member.

Example

Lingo example: foreColor of cast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement changes the color of the text in cast member 1 to the color in palette entry 250.

```
set the foreColor of cast 1 to 250
```

foreColor of sprite (Lingo)

Syntax: `the foreColor of sprite` *whichSprite*

This sprite property determines the foreground color of the sprite specified by *whichSprite*. Setting the `foreColor` sprite property in a Lingo script is equivalent to choosing the foreground color from the tools window when the sprite is selected on the stage.

The foreground color applies only to 1-bit bitmap and shape cast members. It does not affect the display of a text or button cast member. An 8-bit, 16-bit, or 24-bit bitmap is affected, but generally not in a useful way.

The value of a sprite's background color ranges from 0 to 255 for 8-bit color, or from 0 to 15 for 4-bit color. The numbers correspond to the index number of the background color in the current palette. (A color's index number appears in the color palette's lower left corner when you click the color.)

Changing a sprite's foreground color during a mouseDown is a useful way to indicate when a sprite is clicked.

When you set this property within a script while the playback head is not moving, be sure to use the `updateStage` command to redraw the stage. If you are changing several sprite properties--or several sprites--you only have to use one `updateStage` command at the end of all the changes.

The `foreColor of sprite` property can be tested and set, although in order to set it with Lingo the sprite must be a puppet.

Example

Lingo example: foreColor of sprite

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

The following statement sets the variable oldColor to the foreground color of sprite 5:

```
put the foreColor of sprite 5 into oldColor
```

The following statement makes 36 the number for the foreground color of a random sprite from sprite 11 to sprite 13:

```
set the foreColor of sprite (10 + random(3)) to 36
```

forget window (Lingo)

Syntax: `forget window` *whichWindow*

This command tells Lingo to close and delete the window specified by *window* when the window is no longer in use and no other variables refer to it.

Example

Lingo example: forget window

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement has Lingo delete the window Demo when the movie no longer uses the window:

```
forget window "Demo"
```

frame (Lingo)

Syntax: `the frame`

This function refers to the current frame in the current movie.

Example

Related topics: label and marker functions

Lingo example: frame

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sends the playback head to the frame before the current frame:

```
go to (the frame - 1)
```

frameLabel (Lingo)

Syntax: `the frameLabel`

This frame property identifies the label assigned to the current frame. When the current frame has no label, the value of the `frameLabel` property is an empty string ("").

The `frameLabel` property can be tested. However, you cannot use the `frameLabel` to assign a label.

Example

Lingo example: frameLabel

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks the label of the current frame. In this case, the current frameLabel is Start:

```
put the frameLabel
-- "Start"
```

framePalette (Lingo)

Syntax: `the framePalette`

This frame property identifies the cast member number of the palette used in the current frame.

The `framePalette` property can be tested. However, you cannot assign a palette by using the `framePalette` property.

Example

Related topic: puppetPalette command

Lingo example: framePalette

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks the palette used in the current frame. In this case, the palette is cast member 45:

```
put the framePalette
-- 45
```

frameRate of cast (Lingo)

Syntax: `the frameRate of cast` *DVcastMember*

This digital video cast property specifies the frame rate at which the digital video movie specified by *DVcastMember* is played and controls the Play Every Frame setting for the cast member. The possible values for the frameRate of cast correspond to the radio buttons for selecting digital video playback options.

- When `the frameRate of cast` is between 1 and 255, the digital video movie plays every frame at that frame rate. The `frameRate of cast` property cannot be greater than 255.

- When the `frameRate of cast` is set to 0, the digital video movie plays at its normal setting, as if you had the Play Every Frame checkbox unchecked in the Cast Info dialog box.

- When `the frameRate of cast` is set to -1, the digital video movie plays every frame at its normal rate.

- When `the frameRate of cast` is set to -2, the digital video movie plays every frame as fast as possible.

Example

Related topics: movieRate of sprite, movieTime of sprite sprite properties

Lingo example: frameRate of cast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the frame rate of the digital video cast member Rotating Chair to 30 frames per second:

```
set the frame rate of cast "Rotating Chair" to 30
```

This statement has the digital video cast member Rotating Chair play every frame as fast as possible:

```
set the frame rate of cast "Rotating Chair" to -2
```

frameScript (Lingo)

Syntax: `the frameScript`

This frame property identifies the number of the frame script assigned to the current frame.

The `frameScript` property can be tested. However, you cannot assign a script by using the `frameScript` property.

Example

Lingo example: frameScript

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement displays the number of the script assigned to the current frame. In this case, the script number is 25:

```
put the frameScript

-- 25
```

framesToHMS (Lingo)

Syntax: `the framesToHMS(`*frames , tempo , dropFrame , fractionalSeconds* `)`

This function converts the specified number of frames to their equivalent length in hours, minutes, and seconds. This useful for predicting the actual playtime of a movie or controlling a video playback device.

- The integer expression *frames* specifies the number of frames.
- The integer expression *tempo* specifies the tempo in frames per second.
- The *dropFrame* argument is a logical expression. Normally, this is FALSE. This argument is meaningful only if FPS is set to 30 frames per second. (Drop frame is a method of compensating for the color NTSC frame rate which is not exactly 30 frames per second.)
- The *fractionalSeconds* argument determines what happens to residual frames. When TRUE replaces *fractionalSeconds*, the residual frames are converted to the nearest hundredth of a second. When FALSE replaces *fractionalSeconds*, the residual frames are returned as an integer number of frames.

The resulting string uses the form: "sHH:MM:SS.FFD", where:

s -- "-" if the time is less than zero, or space if the time is greater than or equal to zero

HH -- hours

MM -- minutes

SS -- seconds

FF -- fraction of a second if fractionalSeconds is TRUE, or frames if fractionalSeconds is FALSE

D -- "d" if dropFrame is TRUE, or space if dropFrame is FALSE

Example

Related topic: HMStoFrames function

Lingo example: framesToHMS

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement determines the length of a 2710-frame, 30 frame-per-second movie and displays the result in the message window. The dropFrame and fractionalSeconds arguments are both turned off:

```
put framesToHMS(2710, 30, FALSE, FALSE)

-- " 00:01:30.10 "
```

frameTempo (Lingo)

Syntax: `the frameTempo`

This frame property indicates the tempo used by the current frame.

The `frameTempo` property can be tested. However, you cannot set the tempo by using the `framePalette` property.

Example

Related topic: puppetTempo command

Lingo example: frameTempo

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks the tempo used in the current frame. In this case, the tempo is 15 frames per second:

```
put the frameTempo

-- 15
```

freeBlock (Lingo)

Syntax: `the freeBlock`

This function indicates the size of the largest free contiguous block of memory, in bytes. A kilobyte (K) is 1024 bytes. A megabyte (MB) is 1024 kilobytes. In order to load a cast member, you need a free block at least as large as the cast member.

Under Windows, `the freeBlock` returns the same value as `the freeBytes`. The value given under Windows might also include the size of the Windows swap file used for virtual memory on disk in addition to free RAM.

Example

Related topics: freeBytes, memorySize, and ramNeeded functions; size of cast cast property

Lingo example: freeBlock

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement determines whether the largest contiguous free block is smaller than 10K, and displays an alert if it is:

```
if the freeBlock < 10 * 1024 then ¬
   alert "Not enough memory!"
```

freeBytes (Lingo)

Syntax: `the freeBytes`

This function indicates the total number of bytes of free memory, which may or may not be contiguous. A kilobyte (K) is 1024 bytes. A megabyte (MB) is 1024 kilobytes.

This function differs from `freeBlock`, because it reports all free memory, not just contiguous memory.

Under Windows, `the freeBytes` returns the same value as `the freeBlock`. The value given under Windows might also include the size of the Windows swap file used for virtual memory on disk in addition to free RAM.

Example

Related topics: freeBlock, memorySize, and ramNeeded functions; size of cast cast property

Lingo example: freeBytes

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether more than 200K of memory is available, and plays a color movie if it is:

```
if the freeBytes > 200 * 1024 then ¬
   play movie "colorMovie"
```

getaProp (Lingo)

Syntax: `getaProp(list, positionOrProperty )`

This command identifies the value associated with the position or value specified by *positionOrProperty* in the list specified by list.

- When the list is a linear list, the result is the value at the position specified by *positionOrProperty*.
- When the list is a property list, the result is the value associated with the property specified by *positionOrProperty*.

The `getaProp` command gives the result void when the specified value is not in the list.

When used with linear lists, the `getaProp` command does the same as the `getAt` command.

Example

Related topics: getOne, getProp commands

Lingo example: getaProp

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement identifies the value in the third position of the linear list Answers, which consists of [10, 12, 15, 22]:

```
getaProp(Answers, 3)
```

The result is 15, because 15 is the third value in the list.

This statement identifies the value associated with the property #c in the property list Answers, which consists of [#a:10, #b:12, #c:15, #d:22]:

```
getaProp(Answers, #c)
```

The result is 15, which is the value associated with property c.

getAt (Lingo)

Syntax: `getAt(`*list*`, `*position*`)`

This command identifies the item in the position specified by position in the list specified by *list* .

This command does the same as the `getaProp` command for linear lists.

Example

Related topic: getaProp command

Lingo example: getAt

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement has the message window display the third item in the list Answers, which consists of [10, 12, 15, 22]:

```
getAt(Answers, 3)
```

The result is 15.

getLast (Lingo)

Syntax: `getLast(`*list*`)`

This command identifies the last value in the list specified by *list*.

Example

Lingo example: getLast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement identifies the last item in the list Answers, which consists of [10, 12, 15, 22]:

```
put getLast(Answers)
```

The result is 22.

This statement identifies the last item in the property list Bids, which consists of [#Gee:750, #Kayne:600, #Ohashi:850]:

```
put getLast(Bids)
```

The result is 850.

getNthFileNameInFolder (Lingo)

Syntax: `getNthFileNameInFolder(`*folderPath*`,` *fileNumber* `)`

This function returns the filename or folder name that is in the position specified by *Number* in the directory or folder at the path specifed by *pathName*. If the function returns an EMPTY string, you have specified a higher number than there are files in the folder.

Note: To specify other folder names, use the full path defined in the format for the specific platform the movie is running on.

For example, on the Macintosh, you would use a pathname such as `"HardDisk:Director:Movies:"`

To look for files on the Macintosh Desktop, you would use the path `"HardDisk:Desktop Folder:"`

To specify a pathname under Windows, you would use a directory path such as `"C:\Director\Movies"`

Example

Lingo example: getNthFileNameInFolder

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

The following handler returns a list of filenames in the folder at the path specified by the pathname property. To call the function, use parentheses after the handler name in the calling statement, as in put InFolder():

```
on InFolder
   put [] into fileList
   repeat with i = 1 to the maxInteger
      put getNthFileNameInFolder(the pathName, i) ¬
         into n
      if n = EMPTY then exit repeat
      append(fileList, n)
   end repeat
   return fileList
end
```

getOne (Lingo)

Syntax: `getOne(`*list* **,** *value* `)`

This command identifies the position or property associated with the value specified by *value* in the list specified by *list*.

- When the list is a linear list, the result is the value's position in the list.
- When the list is a property list, the result is the property associated with the value in the list.

For values in the list more than once, only the first occurrence is displayed. The `getOne` command gives the result 0 when the specified value is not in the list.

When used with linear lists, the `getOne` command does the same as the `getPos` command.

Example

Related topic: getPos command

Lingo example: getOne

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement identifies the position of the value 12 in the linear list Answers, which consists of [10, 12, 15, 22]:

```
getOne(Answers, 12)
```

The result is 2, because 12 is the second value in the list.

This statement identifies the property associated with the value 12 in the property list Answers, which consists of [#a:10, # b:12, # c:15, #d:22]:

```
getOne(Answers, 12)
```

The result is b, which is the property associated with the value 12.

getPos (Lingo)

Syntax: `getPos(`*list* `,` *value* `)`

This command identifies the position of the value specified by *value* in the list specified by *list*. When the specified value is not in the list, the `getPos` command gives the value 0.

For values in the list more than once, only the first occurrence is displayed. This command does the same as the `getOne` command when used for linear lists.

Example

Related topic: getOne command

Lingo example: getPos

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement identifies the position of the value 12 in the list Answers, which consists of [#a:10, #b:12, #c:15, #d:22]:

```
getPos(Answers, 12)
```

The result is 2, because 12 is the second value in the list.

getProp (Lingo)

Syntax: `getProp(`*list* `,` *property* `)`

This command identifies the value associated with the property specified by *property* in the property list specified by *list*.

The `getProp` command is identical to the `getaProp` command, except that the `getProp` command displays an error message when the specified property is not in the list.

Example

Related topic: getOne command

Lingo example: getProp

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement identifies the value in the third position of the property list Answers, which consists of [a:10, b:12, c:15, d:22]:

```
getProp (Answers, 3)
```

The result is 15, because 15 is the value assigned to the third property in the list.

getPropAt (Lingo)

Syntax: `getPropAt(`*list* `,` *index* `)`

This command identifies the property name associated with the position specified by *index* in the property list specified by *list*.

Example

Lingo example: getPropAt

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement displays the second property in the given list.

```
put getPropAt ([#a:10, #b:20],2)

-- #b
```

global (Lingo)

Syntax: `global` *variable1* `[,` *variable2* `] [,` *variable3* `]...`

This keyword identifies a variable as a global variable so that it can be shared by other handlers or movies.

A global variable can be declared by a script, a handler, or a method, and its value can be used by other scripts, handlers, and methods.

To use a global variable inside a handler, you must declare it to be a global variable; otherwise the handler assumes it is a local variable.

It is important to declare all the global variables within each handler or method that uses the global variable. Otherwise, any variable the handler uses is a local variable, even if it has been declared a global variable by another handler.

For further information, see Variables in Chapter 3 in the *Using Lingo* manual.

Example

Related topics: showGlobals, property commands

Lingo example: global

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

```
global startingPoint
set startingPoint = whichMenu
```

go (Lingo)

Syntax: `go [to] [frame]` *whichFrame*
`go [to] movie` *whichMovie*
`go [to] [frame]` *whichFrame* `of movie` *whichMovie*

This command causes the playback head to jump to the frame specified by *whichFrame* of the movie specified by *whichMovie*. The expression *whichFrame* can be a marker label or an integer frame number. The expression *whichMovie* must specify a movie file. (If the movie is in another folder, *whichMovie* must specify the pathname.)

It's better to refer to marker labels instead of frame numbers, because editing a movie can cause frame numbers to change. Thus a command like `go to frame 35` can become incorrect. It's also easier to read your script if you use marker labels.

The `go to movie` command loads frame 1 of the movie. If the command is called from within a handler or factory, the handler in which it is placed continues executing. If you want to suspend the handler while playing the movie, use the `play` command.

The following are set to FALSE (0) when loading a movie: `the beepOn`, `the constraint` properties, `the keyDownScript`, `the mouseDownScript`, `the mouseUpScript`; `the cursor of sprite` and `immediate of sprite` properties; `the cursor` and `puppetSprite` commands; and custom menus. However, `the timeoutScript` is not reset when loading a movie.

Example

Related topics: label, marker, and pathName functions; play command

Lingo example: go

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sends the playback head to the marker named start:

```
go to "start"
```

This statement sends the playback head to the marker named Memory in the movie named *Noh Tale to Tell*:

```
go to frame "Memory" of movie "Noh Tale to Tell"
```

go loop (Lingo)

Syntax: `go loop`

This command has the movie loop depending on whether there is a marker in the current frame or in a previous frame. It is equivalent to the statement `go marker`(0) that was used in earlier versions of Lingo.

- When there is a marker in the current frame, the movie loops in the current frame.
- When there is no marker in the current frame, the movie loops to the previous marker. When the score has no markers at all, the movie loops to frame 1.

Example

Related topics: go, go next, go previous commands

Lingo example: go loop

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement has the movie loop to the current or previous marker:

```
go loop
```

go next (Lingo)

Syntax: `go next`

This command sends the playback head to the next marker in the movie. It is equivalent to the statement `go marker(1)` that was used in earlier versions of Lingo.

Example

Related topics: go, go loop, go previous commands

Lingo example: go next

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sends the playback head to the next marker in the movie:

```
go next
```

go previous (Lingo)

Syntax: `go previous`

This command sends the playback head to the previous marker in the movie. It is equivalent to the statement `go marker(-1)` that was used in earlier versions of Lingo.

Example

Related topics: go, go loop, go next commands

Lingo example: go previous

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sends the playback head to the previous marker in the movie:

```
go previous
```

halt (Lingo)

Syntax: `halt`

This command has Lingo exit the current handler and any handler that called it. After exiting all handlers, the `halt` command then stops the movie.

Example

Related topics: abort and pass commands; exit keyword

Lingo example: halt

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether the amount of free memory is less than 50K, and if it is, exits all handlers that called it, and then stops the movie:

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

height of cast (Lingo)

Syntax: `the height of cast` *whichCastmember*

This cast property determines the height in pixels of the cast member specified by *whichCastmember*. The `height of cast` property applies only to bitmap and shape cast members. It does not affect text or button cast members.

The `height of cast` property can be tested, but not set.

Example

Related topics: spriteBox command; the width of cast property; height of sprite and width of sprite sprite properties

Lingo example: height of cast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement assigns the height of cast member 50 to the variable vHeight:

```
put the height of cast 50 into vHeight
```

height of sprite (Lingo)

Syntax: `the height of sprite` *whichSprite*

This sprite property determines the vertical size in pixels of the sprite specified by *whichSprite*. The height applies only to bitmap and shape sprites. It does not affect text or button cast members.

Setting this property does not have any effect on bitmap sprites unless the sprite's stretch property is set to TRUE. In order to set this property with Lingo, the sprite must be a puppet

When you set this property within a script while the playback head is not moving, be sure to use the `updateStage` command to redraw the stage. When you are changing several sprite properties--or several sprites--you only have to use the `updateStage` command once at the end of all the changes.

The `height of sprite` property can be tested and set.

Example

Related topics: height of cast, stretch of sprite, width of cast, and width of sprite sprite properties; spriteBox command

Lingo example: height of sprite

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the height of sprite 10 to 26 pixels:

```
set the height of sprite 10 to 26
```

This statement assigns the height of sprite (i + 1) to the variable vHeight:

```
put the height of sprite (i + 1) into vHeight
```

hilite (Lingo)

Syntax: `hilite` *chunkExpression*

This command highlights (selects) the specified chunk in a text sprite. You can select any chunk that Lingo lets you define, such as a character, word, or line. The color that highlights text is the highlight color set in Color in the Control Panels.

Example

Related topics: char...of, item...of, line...of, and word...of chunk expression keywords; delete command; mouseChar, mouseLine, and mouseWord integer functions; field keyword; selEnd and selStart text property

Lingo example: hilite

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement highlights the fourth word in the text cast member Comments:

```
hilite word 4 of field Comments
```

hilite of cast (Lingo)

Syntax: `the hilite of cast` *whichCastmember*

This button property determines whether a checkbox or radio button sprite is selected.

- When the `hilite of cast` is TRUE, the checkbox or radio button is selected.
- When the `hilite of cast` is FALSE, the checkbox or radio button is deselected.

When *whichCastmember* is a string, it is used as the cast name. When *whichCastmember* is an integer, it is used as the cast number.

The hilite of cast button property can be tested and set. The default value is FALSE.

Example

Related topics: checkBoxAccess and checkBoxType properties

Lingo example: hilite of cast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether the button named 2400 baud is selected and sets the baud rate to 2400 if it is:

```
if the hilite of cast "2400 baud" = TRUE then ¬
   setBaudRate(2400)
```

HMStoFrames (Lingo)

Syntax: `HMStoFrames(`*hms* , *tempo* , *dropFrame* , *fractionalSeconds* `)`

This function converts movie length measured in hours-minutes-seconds to the equivalent number of frames at a specifed tempo.

- The string expression *hms* specifies the time in the form "sHH:MM:SS.FFD", where:

  s is a dash (-) if the time is less than zero, or a space if the time is greater than or equal to zero.

  HH represents number of hours.

  MM represents number of minutes.

  SS represents number of seconds.

  FF represents fraction of a second if fractionalSeconds is TRUE. FF represents frames if fractionalSeconds is FALSE.

  D is the letter "d" if dropFrame is TRUE. D is a space if dropFrame is FALSE.

- The expression *tempo* specifies the tempo in frames per second.

- The *dropFrame* argument is a logical expression,. When TRUE replaces *dropFrame*, it is a drop-frame. When FALSE replaces dropFrame, it is not. When the string hms ends in a "d", the time is treated as a drop-frame, regardless of the value of dropFrame.

- The *fractionalSeconds* argument determines the meaning of the fractional seconds. When it is set to TRUE, the numbers after the seconds specify a fraction of a second, to the nearest hundredth of a second. When it is set to FALSE, the numbers after the seconds specify the number of residual frames.

Related topics: framesToHMS function

Lingo example: HMStoFrames

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement determines the number of frames in a 1-minute, 30.1-second movie when the tempo is 30 frames per second. The dropFrame and fractionalSeconds arguments are both turned off:

```
put HMStoFrames(" 00:01:30.10 ", 30, FALSE, FALSE)

-- 2710
```

idle (Lingo)

See on idle event handler.

if (Lingo)

Syntax: `if` *logicalExpression* `then` *then-statement*

or

`if` *logicalExpression* `then` *then-statement*
`else` *else-statement*
`end if`

or

`if` *logicalExpression* `then`
statement(s)
`end if`

or

`if` *logicalExpression* `then`
statement(s)
`else`
statement(s)
`end if`

or

`if` *logicalExpression1* `then`
statement(s)
`else if` *logicalExpression2* `then`
statement(s)
`else if` *logicalExpression3* `then`
statement(s)
`end if`

The `if...then` structure evaluates the *logicalExpression* specified by *logicalExpression*.

- When the condition is TRUE, Lingo executes the statement(s) that follow then.
- When it is FALSE, Lingo executes the statement(s) following `else`. If no statements follow else, Lingo exits the `if...then` structure.

The `else` portion of the statement is optional. If you need to have more than one *then-statement* or *else-statement* , you must end with the form `end` if.

When you use `else`, it always corresponds to the previous if statement. This means that sometimes you need to include an `else nothing` statement to associate an `else` keyword with the proper `if` keyword.

Example

Lingo example: if

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether the Return key was pressed, and then continues if it was:

```
if the key = RETURN then continue
```

This statement checks whether color QuickDraw software is available on the Macintosh. If it is available, Lingo plays the movie "Color Movie." If the color QuickDraw software isn't available, Lingo plays the movie "Black & White Movie":

```
if the colorQD = TRUE then play "Color Movie"
else play "Black & White Movie"
```

This handler checks whether the "q" key was pressed, and then executes the subsequent statements if it was:

```
on keyDown
   if (the commandDown) and (the key = "q") then
      cleanUp
      quit
   end if
end keyDown
```

ilk (Lingo)

Syntax: `ilk(`*item* `,` *type* `)`

This function checks the type of lists, rects, and points by matching *item* with *type* and returning TRUE (1) or FALSE (0).

item	possible type=return value
linear list	#list=1, #linearlist=1, #propertylist=0, #point=0, #rect=0
property list	#list=1, #linearlist=0, #propertylist=1, #point=0, #rect=0
point	#list=1, #linearlist=1, #propertylist=0, #point=1, #rect=0
rect	#list=1, #linearlist=1, #propertylist=0, #point=0, #rect=1

Example

Lingo example: ilk

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement identifies whether the list named bids is a property list and displays the result in the message window:

```
put ilk(bids, #propList)
```

Because the list is a property list the message window displays 1, which is the numeric equivalent of TRUE.

This statement identifies whether the variable vTotal is a list and displays the result in the message window:

```
put ilk(vTotal, #list)
```

Because the variable is not a list, the message window displays 0, which is the numeric equivalent of FALSE.

importFileInto (Lingo)

Syntax: `importFileInto` *castMember* , *fileName*

This command replaces the content of the cast member specified by *castMember* with the file specified by *fileName*. When the file is not in the same folder or directory as the movie, you must also specify the pathname for the folder or directory

Example

Lingo example: importFileInto

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement replaces the content of the sound cast member Memory with the sound file Wind:

```
importFileInto cast "Memory", "Wind"
```

in (Lingo)

See the number of chars in, number of items in, number of lines in, and number of words in functions.

inflate rect (Lingo)

Syntax: `inflate (`*rectangle* `,` *widthChange* `,` *heightChange* `)`

This command changes the dimensions of the rectangle specified by *rectangle*. The change is relative to the center of the rectangle.

- The *widthChange* parameter specifies the number of the rectangle changes horizontally.
- The *heightChange* parameter specifies how much the rectangle changes vertically.

Values less than 0 for horizontal or vertical reduce the rectangle's size.

Example

Lingo example: inflate rect

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement increases the width of the rectangle by 4 pixels and the height by 2 pixels:

```
put inflate (Rect(10, 10, 20, 20), 2, 1)

-- Rect (8, 9, 22, 21)
```

This statement increases both the height and width of the rectangle by 20 pixels:

```
put inflate (Rect(0, 0, 100, 100), 10, 10)

-- Rect (-10, -10, 110, 110)
```

ink of sprite (Lingo)

Syntax: `the ink of sprite` *whichSprite*

This sprite property determines the ink effect applied to the sprite specified by *whichSprite*.

The following ink effects are available:

0--Copy
1--Transparent
2--Reverse
3--Ghost
4--Not copy
5--Not transparent
6--Not reverse
7--Not ghost
8--Matte
9--Mask
32--Blend
33--Add pin
34--Add
35--Subtract pin
36--Background transparent
37--Lightest
38--Subtract
39--Darkest

In the case of background transparent (ink effect 36), you set the color that becomes transparent by selecting the color from the background color chip in the tools window while the sprite is selected in the score. You can do the same thing by using Lingo to set the `backColor` property, but this is unpredictable when the sprite has more than 1-bit color.

If you set this property within a script while the playback head is not moving, be sure to use the `updateStage` command to redraw the stage. If you change several sprite properties--or several sprites--you need to use only one `updateStage` command at the end of all the changes.

For further information about ink effects, see the *Using Director* manual.

The ink sprite property can be tested and set. To change any sprite property using Lingo, the sprite must be a puppet.

Example

Related topics: backColor of sprite and foreColor of sprite sprite property

Lingo example: ink of sprite

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement changes the variable currentInk to the value for the ink effect of sprite 3:

```
put the ink of sprite 3 into currentInk
```

This statement gives sprite (i + 1) a matte ink effect by setting the ink effect of sprite property to 8, which specifies matte ink:

```
set the ink of sprite (i + 1) to 8
```

inside (Lingo)

Syntax: `inside(`*point*`,` *rectangle* `)`

This function indicates whether the point specified by *point* is within the rectangle specified by *rectangle*.

- When the point is within the rectangle, the `inside` function is TRUE.
- When the point is outside the rectangle, the `inside` function is FALSE.

Example

Related topics: map, mouseH, mouseV, and point functions

Lingo example: inside

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement indicates whether the point Center is within the rectangle Zone and displays the result in the message window:

```
put inside(Center, Zone)
```

installMenu (Lingo)

Syntax: `installMenu` *cast member*

This command installs the menu defined in the text cast member specified by *cast member*. These custom menus appear only while the movie is playing. To remove the custom menus, use the `installMenu` command with no argument, or with 0 as the argument.

For an explanation of how menu items are defined in a text cast member, refer to the menu: keyword.

Example

Related topic: menu keyword

Lingo example: installMenu

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement installs the menu defined in text cast member 37:

```
installMenu 37
```

This statement installs the menu defined in the text cast member named Menubar:

```
installMenu cast "Menubar"
```

This statement disables menus that were installed by the installMenu command:

```
installMenu 0
```

instance (Lingo)

Syntax: `instance` *variable1* `[,` *variable2* `]` `[,` *variable3* `]` *...*

This keyword makes a variable an instance variable--a special kind of variable used with factories. A factory can assign instance variables to specific objects. Instance variables contain unique values for each individual object. The methods of a factory can use the instance variables.

An instance variable is available only to the factory object it is associated with. An instance variable's value is established when an object is created, or when a method is used to change it. Each new object created by a factory has its own set of instance variable values that persist as long as the object persists.

In Director 4, it is recommended that you use lists and parent scripts rather than factories. Lists and parent scripts are a simpler way of achieving the same result.

To use an instance variable within a factory method, you must declare it an instance variable by using the `instance` keyword; otherwise the factory assumes it is a local variable. Variables created inside a handler are assumed to be local, unless otherwise specified to be global or instance variables.

Instance variables need only be declared once in the factory--not in every method as is necessary with global variables.

An instance variable is usually defined in the `mNew` method of a factory. Subsequently, the values of instance variables can be changed by other methods.

Example

Related topics: factory, global, and method keywords

Lingo example: instance

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler uses the instance keyword in the mNew method to declare instance variables in a factory:

```
method mNew parameter1, parameter2
   instance variable1, variable2
   put parameter1 into variable1
   put parameter2 into variable2
end mNew
```

This handler also declares instance variables:

```
method mNew
   global counter
   instance mySpeed, mySprite
   put 0 into mySpeed
   put counter into mySprite
end mNew
```

integer (Lingo)

Syntax: `integer(`*numericExpression* `)`

This function rounds the value of *numericExpression* to the nearest whole integer.

You can force an integer to be a string by using the `string()` function.

Example

Related topics: float and string functions

Lingo example: integer

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement rounds off the number 3.75 to the nearest whole integer:

```
put integer(3.75)

-- 4
```

This statement rounds off the value in parentheses. This provides a usable value for the locH of sprite, which requires an integer:

```
set the locH of sprite 1 ¬

   to integer(0.333 * stageWidth)
```

integerP (Lingo)

Syntax: `integerP(`*expression*`)`

This function indicates whether the expression specified by *expression* is an integer:

- When *expression* can be evaluated to an integer, `integerP` is TRUE (1).
- When *expression* cannot be evaluated to an integer, `integerP` is FALSE (0).

The "P" in integerP stands for "predicate."

Example

Related topics: floatP, objectP, stringP, and symbolP functions

Lingo example: integerP

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether 3 can be evaluated to an integer. Because it is an integer, the message window displays the number 1, which is the numeric equivalent of TRUE:

```
put integerP(3)

-- 1
```

This statement checks whether 3 can be evaluated to an integer. Because 3 surrounded by quotes is a string and therefore cannot be evaluated to an integer, the message window displays the number 0, which is the numeric equivalent of FALSE:

```
put integerP("3")

-- 0
```

This statement checks whether the numerical value of the string in text cast member Entry is an integer, and displays an alert if it isn't.

```
if integerP(value(field "Entry")) = FALSE then ¬

   alert "Please enter an integer."
```

intersect (Lingo)

Syntax: `intersect(`*rectangle1* `,` *rectangle2* `)`

This function determines the rectangle formed where *rectangle1* and *rectangle2* intersect.

Example

Related topics: map, offset, and rect functions

Lingo example: intersect

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement assigns the variable newRectangle the rectangle formed where rectangle toolKit intersects rectangle Ramp:

```
set newRectangle = intersect(vToolKit, vRamp)
```

into (Lingo)

This code fragment occurs in a number of Lingo constructs.

item...of (Lingo)

Syntax: `item` *whichItem* `of` *chunkExpression*

or

`item` *firstItem* `to` *lastItem* `of` *chunkExpression*

This chunk expression keyword specifies an item or a range of items in a chunk expression. An item in this case is any sequence of characters delimited by commas or by the character specified as a delimiter by the `itemDelimiter` global property.

The terms *whichItem* , *firstItem* , and *lastItem* must be integers or integer expressions that refer to the position of items in the chunk.

Chunk expressions refer to any character, word, item, or line in any source of text. Sources of text include text cast members and variables that hold strings.

When the number that specifies the last item is greater than the item's position in the chunk expression, the actual last item is specified instead.

Example

Related topics: char...of, line...of, and word...of chunk expression keywords; number of items in chunk function

Lingo example: item...of

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement determines the third item in the chunk expression that consists of names of colors and displays the result in the message window:

```
put item 3 of "red, yellow, blue green, orange"

-- " blue green"
```

The result is the entire chunk blue green because this is the entire chunk within the commas. Note that the result includes the space before the word "blue."

This statement attempts to determine the third through tenth items in the chunk expression. Because there are only four items in the chunk expression, the fourth item is used instead of the tenth item. The result appears in the message window:

```
put item 3 to 10 of "red, yellow, blue green, orange"

-- " blue green, orange"

put item 10 of "red, yellow, blue green, orange"

-- ""
```

This statement inserts the item Desk as the fourth item in the second line of the text cast member All Bids:

```
put "Desk" into item 4 of line 2 ¬

   of field "All Bids"
```

itemDelimiter (Lingo)

Syntax: `the itemDelimiter`

This property indicates the special character used to separate items.

Use the `itemDelimiter` function to parse filenames by setting `itemDelimiter` to a colon (:) on the Macintosh or a backslash (\) under Windows. Restore `the itemDelimiter` to a comma (,) for normal operation.

The `itemDelimiter` function can be tested and set.

Lingo example: itemDelimiter

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler determines the last component in a Macintosh pathname. The handler first records what the current delimiter is, and then changes the delimiter to a colon (:). When a colon is the delimiter, Lingo can use the last item of to determine the last item in the chunk that makes up a Macintosh pathname. Before exiting, the delimiter is reset to its original value.

```
on getLastComponent pathName

   set oldDelimiter to the itemDelimiter

   set the itemDelimiter = ":"

   set f to the last item of pathName

   set the itemDelimiter to oldDelimiter

   return f

end
```

key (Lingo)

Syntax: `the key`

This function indicates the last character that was typed. (This value is the ANSI value assigned to the key, not the numerical value.)

This function can be used for testing keys within primary event handlers and for navigation/keyboard shortcuts.

You can use `the key` to write handlers that perform certain actions when the user presses specific keys. This is a way to provide keyboard shortcuts and other forms of interactivity for the user. When used in a primary event handler, the actions you specify are the first to be executed.

Example

Related topics: commandDown, controlDown, key, keyCode, and optionDown functions

Lingo example: key

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

These statements have the movie pause when the user presses Return. By setting the keyDownScript property to checkKey, the on StartMovie handler makes the checkKey handler the first event handler executed when a key is pressed. The checkKey handler checks whether the Return key is pressed and pauses the movie if it is:

```
on startMovie

   set the keyDownScript to "checkKey"

end startMovie

on checkKey

   if the key = RETURN then pause

end
```

This keyDown handler checks whether the last key pressed is the Enter key on the number pad, and then calls the handler addNumbers if it is:

```
on keyDown

   if the key = ENTER then addNumbers

end keyDown
```

keyCode (Lingo)

Syntax: `the keyCode`

This function gives the numerical code for the last key pressed. (This keyboard code is the key's numerical value, not the ANSI value.)

You can use the `keyCode` function to detect when the user has pressed the arrow or function keys, which cannot be specified by the `key` function. The character assigned to the key for a given `keyCode` value can vary from keyboard to keyboard. As a result, `the keyCode` can refer to different characters on different computers.

The `keyCode` function can be tested but not set.

Example

Related topics: commandDown, controlDown, key, key, and optionDown functions

Lingo example: keyCode

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler uses the message window to display the appropriate key code each time a key is pressed:

```
on enterFrame

   set the KeyDownScript = "put the keyCode"

end
```

This statement checks whether the Up arrow (whose key code is 126) is pressed, and goes to the previous marker if it is:

```
if the keyCode = 126 then go to marker(-1)
```

keyDown (Lingo)

See on keyDown handler.

keyDownScript (Lingo)

Syntax: `the keyDownScript`

This property specifies the Lingo that is executed when a key is pressed. The Lingo can be a simple statement or a calling script for a handler.

When a key is pressed and the `keyDownScript` is defined, Lingo executes the instructions specified for the `keyDownScript` first. Unless the instructions include the pass command so that the keyDown message can pass on to other objects in the movie, no other keyDown handlers are executed.

Setting the `keyDownScript` property does the same as using the `when keyDown then` command that appeared in earlier versions of Director.

When the instructions you specify for the `keyDownScript` property are no longer appropriate, turn them off by using the statement `set the keyDownScript to EMPTY`.

Related topics: keyUpScript, mouseDownScript, and mouseUpScript properties

Lingo example: keyDownScript

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the keyDownScript to if the key = RETURN then continue. When this is in effect and the movie is paused, the movie always continues whenever the user presses the Return key.

```
set the keyDownScript ¬
   to "if the key = RETURN then continue"
```

keyUpScript (Lingo)

Syntax: `the keyUpScript`

This property specifies the Lingo that is executed when a key is released. The Lingo can be a simple statement or a calling script for a handler.

When a key is released and the `keyUpScript` is defined, Lingo executes the instructions specified for the `keyUpScript` first. Unless the instructions include the pass command so that the `keyUp` message can pass on to other objects in the movie, no other on keyUp handlers are executed.

When the instructions you've specified for `the keyUpScript` property are no longer appropriate, turn them off by using the statement `set the keyUpScript to empty`.

Example

Lingo example: keyUpScript

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the keyUpScript to if the key = RETURN then continue. When this is in effect and the movie is paused, the movie always continues whenever the user presses the Return key.

```
set the keyUpScript ¬

   to "if the key = RETURN then continue"
```

label (Lingo)

Syntax: `label(`*expression*`)`

This function indicates the frame associated with the marker label specified by *expression*. The term *expression* should be a label in the current movie; if it is not, this function returns 0.

Example

Related topics: go and play commands; labelList and marker functions

Lingo example: label

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sends the playback head to the tenth frame after the frame labeled Start:

```
go to label("Start") + 10
```

This statement assigns the frame number of the fourth item in the label list to the variable whichFrame:

```
put label(line 4 of the labelList) into whichFrame
```

labelList (Lingo)

```
the labelList
```

This function gives a listing of the frame labels in the current movie, one label per line.

Example

Related topics: label and marker functions

Lingo example: labelList

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement makes a listing of frame labels the content of the text cast member Key Frames:

```
put the labelList into field "Key Frames"
```

last (Lingo)

Syntax: `the last` *chunk* `in (`*chunkExpression*`)`

This function identifies the last chunk specified by *chunk* of the chunk expression specified by *chunkExpression*.

Chunk expressions refer to any character, word, item, or line in any container of text. Containers include the contents of text cast members; variables that hold strings; and specified characters, words, items, lines, and ranges within containers.

Example

Related topics: char...of and word...of chunk expression keywords

Lingo example: last

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement identifies the last word of the string "Macromedia, the multimedia company" and displays the result in the message window:

```
put the last word of "Macromedia,¬

   the multimedia company"
```

The result is the word "company".

This statement identifies the last character of the string "Macromedia, the multimedia company" and displays the result in the message window:

```
put the last char of "Macromedia,¬

   the multimedia company"
```

The result is the letter "y".

lastClick (Lingo)

Syntax: `the lastClick`

This function gives the time in ticks (60ths of a second) since the mouse button was last pressed.

The `lastClick` can be tested, but not set.

Example

Related topics: lastEvent, lastKey, and lastRoll functions; startTimer command

Lingo example: lastClick

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether it has been ten seconds since the last mouse click, and sends the playback head to the marker No Click if it has:

```
if the lastClick > (10 * 60) then go to "No Click"
```

lastEvent (Lingo)

Syntax: `the lastEvent`

This function gives the time in ticks (60ths of a second) since the last mouse click, mouse roll, or key press occurred.

Example

Related topics: lastClick, lastKey, and lastRoll functions; startTimer command

Lingo example: lastEvent

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether it has been ten seconds since the last mouse click, mouse roll, or key press, and sends the playback head to the marker Help if it has:

```
if the lastEvent > 10 * 60 then go to "Help"
```

lastFrame (Lingo)

Syntax: `the lastFrame`

This property is the number of the last frame in the movie.

The lastFrame property can be tested but not set.

Example

Lingo example: lastFrame

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement displays the number of the last frame of the movie in the message window:

```
put the lastFrame
```

lastKey (Lingo)

Syntax: `the lastKey`

This function gives the time in ticks (60ths of a second) since the last key was pressed.

Example

Related topics: lastClick, lastEvent, and lastRoll functions; startTimer command

Lingo example: lastKey

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether it has been 10 seconds since the last key was pressed, and sends the playback head to the marker "No Key" if it has:

```
if the lastKey > 10 * 60 then go to "No Key"
```

lastRoll (Lingo)

Syntax: `the lastRoll`

This function gives the time in ticks (60ths of a second) since the mouse was last moved.

Example

Related topics: lastClick, lastEvent, and lastKey functions; startTimer command

Lingo example: lastRoll

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether it has been 45 seconds since the mouse was last moved, and sends the playback head to the marker "No Roll" if it has:

```
if the lastRoll > 45 * 60 then go to "No Roll"
```

left of sprite (Lingo)

Syntax: `the left of sprite` *whichSprite*

This sprite property is the left horizontal coordinate of the bounding rectangle of the sprite specified by *whichSprite*.

Sprite coordinates are measured in numbers of pixels, starting with (0,0) at the upper left corner of the stage.

The `left of sprite` property can be tested, but not set. Use the `spriteBox` command to set the left horizontal coordinate of a sprite.

Example

Related topics: bottom of sprite, height of sprite, locH of sprite, locV of sprite, right of sprite, top of sprite, and width of sprite, properties; spriteBox command

Lingo example: left of sprite

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

The following statement determines whether the sprite's left edge is to the left of the stage's left edge. If the sprite's left edge is to the stage's left edge, the script runs the handler offLeftEdge:

```
if the left of sprite 3 < 0 then offLeftEdge
```

This statement measures the left horizontal coordinate of the sprite numbered (i + 1) and assigns the value to the variable named vLowest:

```
put the left of sprite (i + 1) into vLowest
```

length (Lingo)

Syntax: `length(`*string* `)`

This function gives the number of characters in the string specified by *string*. Spaces and control characters like Tab and Return count as characters.

The `length` function can be tested, but not set.

Example

Related topics: chars and offset functions

Lingo example: length

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement displays the number of characters in the string "Macromedia":

```
put length("Macromedia")

-- 10
```

This statement checks whether the content of the text cast member File Name has more than 31 characters and displays an alert if it does:

```
if length(field "File Name") > 31 then ¬

   alert "That file name is too long."
```

line...of (Lingo)

Syntax: `line` *whichLine* `of` *chunkExpression*

or

`line` *firstLine* `to` *lastLine* `of` *chunkExpression*

This chunk expression keyword specifies a line or a range of lines in a chunk expression. A line chunk is any sequence of characters delimited by Returns.

The expressions *whichLine*, *firstLine*, and *lastLine* must be integers that specify a line in the chunk.

Chunk expressions refer to any character, word, item, or line in any source of text. Sources of text include text cast members and variables that hold strings.

Example

Related topics: char...of, item...of, word...of, and chunk expression keywords; number of words in chunk function

Lingo example: line...of

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement assigns the first four lines of the variable vAction to the text cast member To Do:

```
set the text of cast "To Do" = lines 1 to 4 ¬
   of vAction
```

This statement inserts the word "and" after the second word of the third line of the text assigned to the variable vNotes:

```
put "and" after word 2 of line 3 of vNotes
```

lineSize of sprite (Lingo)

Syntax: `the lineSize of sprite` *whichSprite*

This sprite property determines the thickness, in pixels, of the border of the sprite specified by *whichSprite*. The `lineSize of sprite` property applies only to shape sprites. For non-rectangular shapes the border is the edge of the shape, not its bounding rectangle.

The `lineSize of sprite` property can be tested and set. For a sprite property to be set using Lingo, the sprite must be a puppet.

Example

Lingo example: lineSize of sprite

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement displays the thickness of the border of sprite 4:

```
put the lineSize of sprite 4 into thickness
```

This statement sets the thickness of the border of sprite 4 to 3 pixels:

```
set the lineSize of sprite 4 to 3
```

list (Lingo)

Syntax: `list(`*value1* `,` *value2* `,` *value3*... `)`

This function defines a linear list made up of the values specified by *value1*, *value2*, *value3*.... This is an alternative way to create a list.

Example

Lingo example: list

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the variable named designers equal to a linear list that contains the names Gee, Kayne, and Ohashi:

```
set designers = list("Gee", "Kayne", "Ohashi")
```

The result is the list ["Gee", "Kayne", "Ohashi"].

listP (Lingo)

Syntax: `listP(`*item*`)`

This function indicates whether the item specified by *item* is a list.

- When `listP` is TRUE (1), the item specified by item is a list.
- When `listP` is FALSE (0), the item specified by item is not a list.

Example

Related topic: ilk function

Lingo example: listP

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether the list named designers is a list and displays the result in the message window:

```
put listP(designers)
```

The result is 1, which is the numerical equivalent of TRUE.

loaded of cast (Lingo)

Syntax: `the loaded of cast` *whichCastMember*

This cast property specifies whether the cast member specified by *whichCastMember* is loaded into memory.

- When the `loaded of cast` is TRUE, the cast member is loaded into memory.
- When the `loaded of cast` is FALSE, the cast member is not loaded into memory.

The `loaded of cast` property can be tested but not set.

Example

Related topics: size of cast cast property; preLoad and unload commands; ramNeeded function

Lingo example: loaded of cast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether cast member Demo Movie is loaded in memory, and goes to an alternate movie if it isn't:

```
if the loaded of cast "Demo Movie" = FALSE then ¬
go to "Waiting"
```

locH of sprite (Lingo)

Syntax: `the locH of sprite` *whichSprite*

This sprite property is the horizontal position of the specified sprite's registration point. Sprite coordinates are relative to the upper left corner of the stage. See *Using Director* for information about registration points.

The `locH of sprite` property can be tested and set. For a sprite property to be set using Lingo, the sprite must be a puppet.

If you set this property within a script while the playback head is not moving, be sure to use the `updateStage` command to redraw the stage. If you are changing several sprite properties--or several sprites--you need only one `updateStage` command at the end of all the changes.

Example

Related topics: bottom of sprite, height of sprite, left of sprite, locV of sprite, right of sprite, top of sprite, and width of sprite sprite properties; spriteBox and updateStage commands

Lingo example: locH of sprite

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether the horizontal position of sprite 9's registration point is to the right of the right edge of the stage, and moves the sprite's right edge to the edge of the stage if it is:

```
if the locH of sprite 9 > the stageRight then ¬
   set the locH of sprite 9 to the stageRight
```

This statement puts sprite 15 at the same horizontal location as the mouse click:

```
set the locH of sprite (15) to the mouseH
```

locV of sprite (Lingo)

Syntax: `the locV of sprite` *whichSprite*

This sprite property is the vertical position of the specified sprite's registration point. Sprite coordinates are relative to the upper left corner of the stage. See *Using Director* for information about registration points.

The `locV of sprite` property can be tested and set. For a sprite property to be set using Lingo, the sprite must be a puppet.

If you set this property within a script while the playback head is not moving, be sure to use the `updateStage` command to redraw the stage. If you are changing several sprite properties--or several sprites--you need only one `updateStage` command at the end of all the changes.

Example

Related topics: bottom of sprite, height of sprite, left of sprite, locH of sprite, right of sprite, top of sprite, and width of sprite sprite properties; spriteBox and updateStage commands

Lingo example: locV of sprite

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether the vertical position of sprite 9's registration point is to the below the bottom of the stage, and moves the sprite's bottom edge to the bottom of the stage if it is:

```
if the locV of sprite 9 > the stageBottom then set the locV of sprite 9 to
the stageBottom
```

This statement puts sprite 15 at the same vertical location as the mouse click:

```
set the locV of sprite (15) to the mouseV
```

log (Lingo)

Syntax: `log(`*number* `)`

This function calculates the natural logarithm of the number specified by *number* , which must be a decimal number.

Example

Lingo example: log

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement assigns the natural logarithm of 10.5 to the variable vAnswer:

```
set vAnswer = log (10.5)
```

This statement calculates the natural logarithm of the square root of the value vNumber, and then assigns the result to the variable vAnswer:

```
set vAnswer = log (sqrt (vNumber))
```

long (Lingo)

See the date and time functions.

loop (Lingo)

Syntax: `loop`

This keyword refers to `the marker`. The `loop` keyword with the `go to` command is equivalent to the statement `go to marker`.

Example

Lingo example: loop

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler loops between movie in the current frame and the previous marker:

```
on wait

   go loop

end
```

loop of cast (Lingo)

Syntax: `the loop of cast` *castName*

This cast property specifies whether digital video movie cast members are set to loop.

- When loop is set to 1, the digital video movie cast member loops.
- When loop is set to 0, the digital video movie cast member does not loop.

Example

Lingo example: loop of cast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the digital video movie cast member Demo to loop:

```
set the loop of cast Demo to 1
```

machineType (Lingo)

Syntax: `the machineType`

This function indicates the kind of computer that is currently being used. These codes indicate the type of Macintosh computer:

1	Macintosh 512Ke
2	Macintosh Plus
3	Macintosh SE
4	Macintosh II
5	Macintosh IIx
6	Macintosh IIcx
7	Macintosh SE/30
8	Macintosh Portable
9	Macintosh IIci
11	Macintosh IIfx
15	Macintosh Classic
16	Macintosh IIsi
17	Macintosh LC
18	Macintosh Quadra 900
19	PowerBook 170
20	Macintosh Quadra 700
21	Classic II
22	PowerBook 100
23	PowerBook 140
24	Macintosh Quadra 950
25	Macintosh LCIII
27	PowerBook Duo 210
28	Macintosh Centris 650
30	PowerBook Duo 230
31	PowerBook 180
32	PowerBook 160
33	Macintosh Quadra 800
35	Macintosh LC II
42	Macintosh IIvi
46	Macintosh II vx
47	Macintosh Color Classic
48	PowerBook 165c
50	Macintosh Centris 610
52	PowerBook 145
76	Macintosh Quadra 840av
256	IBM PC-type machine

Note: These codes are for general classification purposes only. It is unwise to use them to make assumptions about the performance or screen size of the computer your movie is running on.

Example

Related topics: colorDepth property; colorQD function

Lingo example: machineType

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether the computer is a Macintosh Classic and plays the movie Classic Movie if it is:

```
if the machineType = 15 then play "Classic Movie"
```

map (Lingo)

Syntax: `map(targetRect, sourceRect, destination Rect)`
`map(targetpoint, sourceRect, destination Rect)`

This function is used to position and size a rectangle or point, based on the relationship of a source rectangle to a target.

Example

Lingo example: map

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler modifies the rectangle of sprite n so that it has the same relationship to the dimensions of the stage that sprite 2 has:

```
on scaleMySprite n
   set the stretch of sprite to TRUE
   set the rect of sprite n = ¬
      map(the rect of sprite n, ¬
      the rect of sprite 2, ¬
      the rect of the stage)
   updateStage
end
```

marker (Lingo)

This function returns the frame number of markers before or after the current frame. This can be useful for implementing a "next" or "previous" button, or for setting up an animation loop.

The *integerExpression* can evaluate to any positive or negative integer or zero. For example,

`marker(2)` returns the frame number of the second marker after the current frame;

`marker(1)` returns the frame number of the first marker after the current frame;

`marker(0)` returns the frame number of the current frame, if the current frame is marked, or the frame number of the previous marker if the current frame is not marked;

`marker(-1)` returns the frame number of the first marker before the current frame;

`marker(-2)` returns the frame number of the second marker before the current frame.

Example

Related topics: go command; frame, labelList and label functions

Lingo example: marker

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sends the playback head to the beginning of the current frame if the current frame has a marker:

```
go to marker(0)
```

This statement sets the variable nextMarker equal to the next marker in the score:

```
put marker(1) into nextMarker
```

mAtFrame (Lingo)

Syntax: `method mAtFrame` *frameNumber, subFrameNumber*
`[`*statements*`]`
`end mAtFrame`

This special message is used by Lingo in conjunction with any XObject or factory-produced object that has been assigned to the `perFrameHook` property, as follows:

```
set the perFrameHook to objectName
```

Subsequently, the `mAtFrame` message is automatically sent to the object every time the playback head reaches a new frame, or every time an internal subframe is reached within a visual transition.

The functionality within `mAtFrame` must be supplied by the XObject or factory definition (as opposed to predefined methods). That is why `mAtFrame` is technically called a message, instead of a predefined method.

The `perFrameHook` property is primarily designed for use with XObjects that need to be called at every subframe, such as frame-by-frame video recorders. Factory-produced objects should generally use an on stepMovie handler if they need to be called at every frame.

Related topics: factory and method keywords; perFrameHook property; or on stepMovie movie handler

max (Lingo)

Syntax: `max(list)`

or

`max (value1, value2, value3, ...)`

This function returns the highest value in the specified list, or the highest of a given series of values.

Example

Lingo example: max

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler assigns the variable vWinner the maximum value in the list vBids, which consists of [#Castle:600, #Schmitz:750, #Wang:230]. The result is then inserted in the content of the text cast member Congratulations:

```
on findWinner vBids

   set vWinner = max(vBids)

   set the text of "Congratulations" = ¬

   "You have won, with a bid of $" & vWinner &"!"

end
```

maxInteger (Lingo)

Syntax: `the maxInteger`

This property returns the largest whole number that is supported by the system. On most personal computers, this is 2,147,483,647 (2 to the 31st power, minus 1.)

This can be useful for initializing boundary variables before a loop or for limit testing.

Note: The minimum integer value can be found using the formula `(the maxInteger*(-1))-1`. Higher and lower values can be dealt with using floating-point numbers.

Example

Lingo example: maxInteger

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This example generates a table in the message window, of the maximum decimal value that can be represented by a certain number of binary digits:

```
on showMaxValues
   put 31 into b
   put the maxInteger into v
   repeat while v > 0
      put b && "-" && v
      put b-1 into b
      put v/2 into v
   end repeat
end
```

mci (Lingo)

Syntax: `mci "`*string*`"`

The multimedia extensions for Windows respond to commands sent to the media control interface, or MCI. When you play your Director moives under Windows, the mci command can pass the strings specified by strings to the Windows media control interface.

Strings passed by the `mci` command play only under Windows; they are not executed on the Macintosh. Because the Macintosh does not support the MCI interface, the `mci` command gives you a way to include commands intended for the Windows environment within a movie that also plays on the Macintosh.

Lingo example: mci

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement makes the command play cdaudio from 200 to 600 track 7 play only when the movie plays back under Windows:

```
mci "play cdaudio from 200 to 600 track 7"
```

mDescribe (Lingo)

Syntax: *XObjectName* `(mDescribe)`

This predefined method is used only with XObjects (as opposed to factory-produced objects). The purpose of `mDescribe` is to create a list of methods in the message window. This list contains the names of other methods of the XObject, plus any comments by the programmer of the XObject that document the functionality or syntax of these methods.

You only use this method for authoring. Do not include in scripts within a movie.

Before using `mDescribe` to display an XObject's list of methods, first open the appropriate library using the `openXlib` command. To display information about the XObject, enter the `showXlib` command followed by *XobjectName* `(mDescribe)` in the message window. A display of all open Xlibrary resource files and all XObjects contained in those Xlibraries.

Related topics: mMessageList predefined method; showXLib command

Lingo example: mDescribe

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement displays methods and comments assigned to the fileIO XObject:

```
fileIO(mDescribe)
```

mDispose (Lingo)

Syntax: *object*`(mDispose)`

This predefined method supports factories in earlier versions of Lingo. In Director 4, it is recommended that you use lists and parent lists. They are a simpler way of achieving the same result.

This predefined method is used to destroy the object specified by *object*, which was created earlier with the `mNew` method. It is used to dispose of both factory-produced objects and instances of XObjects. Use it to free up memory when an object is no longer needed.

You do not need to explicitly dispose of child objects created from parent scripts. Lingo disposes of these objects when they are no longer referenced by a variable within the movie.

It is best that you check for previous instances of an object with the same name, and dispose of it before creating new instances of an object using `mNew`. The initialize handler in the following example illustrates this. In this way, if the movie is aborted before the normal `mDispose`, you won't fill up memory by repeatedly creating new objects. This can happen during the development of a project, when you repeatedly stop it before the end, and play it again from the beginning.

If you define an `mDispose` method in a factory, it will be executed instead of the predefined method. The result--which is seldom what you want--is that the object will not really get disposed. If you need to perform various housekeeping actions before disposing, put the routines in a method with another name, like `mRelease`.

Example

Related topics: mNew predefined method

Lingo example: mDispose

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler determines whether the item assigned to the variable myObject is an XObject and disposes of it if it is:

```
on cleanUp

   global myObject

      if objectP(myObject) then myObject(mDispose)

end cleanUp
```

me (Lingo)

Syntax: `me`

This keyword is used within parent scripts as a shorthand means of referring to the script itself.

In earlier versions of Director, the `me` keyword supported factories. In Director 4, it is recommended that you use the `birth` function or lists. They are a simpler way of achieving the same result.

Example

Related topics: birth function; ancestor property

Lingo example: me

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the object myBird1 to the script named Bird. The me keyword accepts the parameter script "Bird" and is used to return that parameter:

```
set myBird1 to birth (script "Bird")
```

This is the birth handler of the Bird script:

```
on birth me

return me

end
```

memorySize (Lingo)

Syntax: `the memorySize`

This function returns the total amount of memory (in bytes) allocated to the program, whether in use or free. It is useful for checking minimum memory requirements. A kilobyte (K) is 1024 bytes. A megabyte (MB) is 1024K.

Under Windows, the value that `the memorySize` returns can include the Windows swap file used for virtual memory.

Example

Related topics: freeBlock and freeBytes, and ramNeeded functions; size of cast cast property

Lingo example: memorySize

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether less than 500K is allocated to the program and displays an alert when it is:

```
if the memorySize < 500 * 1024 then alert ¬
   "There is not enough memory to run this movie."
```

See also freeBlock, freeBytes, and ramNeeded functions; the size of cast cast property

menu (Lingo)

Syntax: `menu:` *menuName*
itemName Å script
itemName Å script
...

or

`menu:` *menuName*
itemName Å script
itemName Å script
...
[*more menus*]

This keyword is used to specify the actual content of custom menus, in conjunction with the `installMenu` command. Menu definitions are typed in the text cast members. You can have a script execute when the user chooses that item by putting the script after the "Å" symbol (press Alt-0197 to create the symbol). A new menu is defined by the subsequent occurrence of the menu: keyword.

When the movie plays on the Macintosh, youou can create hierarchical menus by using XCMDs or simulate them by writing scripts that display a graphic cast member that mimics a submenu.

You can use special characters to define custom menus:

Symbol	Example--Description (Command key)
`Å`	`Open/O Å go to frame "Open"` -- Associates a script with the menu item (Alt-0197)
`@`	`menu: @` -- When the movie plays on the Macintosh, creates the Apple symbol and enables Macintosh menu bar items when you define your own Apple menu
`(`	`Save(` -- Disables the menu item
`(-`	`(-` -- Creates a disabled line in the menu
`Ã`	`ÃEasy Select` -- Checks the menu with a checkmark (create by pressing Alt-0195)
`<B`	`Bold<B` -- Sets the menu item's style to Bold
`<I`	`Italic<I` -- Sets the style to Italic
`<U`	`Underline<U` -- Sets the style to Underline
`<O`	`Outline<O` -- Sets the style to Outline
`<S`	`Shadow<S` -- Sets the style to Shadow
`/`	`Quit/Q` -- Defines a Control-key equivalent

Special symbols should follow the item name, and precede the "Å" symbol. You can also use more than one special character to define a menu item. Using <B<U, for example, sets the style to Bold and Underline.

The menus you create appear the same on the Macintosh and under Windows, except that Windows menus do not support Outline font.

Example

Related topics: installMenu command; name of menu menu property; name of menuItem, number of menuItems, checkMark of menuItem, enabled of menuItem, and script of menuItem menu item properties

Lingo example: menu

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This set of statements specifies the content of a custom File menu:

```
menu: File
Open/O Å go to frame "Open"
Close/W Å go to frame "Close"
   (-
Quit/Q Å go to frame "Quit"
menu: Edit
Undo/Z Å go to frame "Undo"
```

menuItem (Lingo)

See name of menuItem, number of menuItems, checkMark of menuItem, enabled of menuItem, and script of menuItem menu item properties.

method (Lingo)

Syntax: `method methodName [argument1] [, argument2] ...`

This keyword supports factories in earlier versions of Lingo. In Director 4, it is recommended that you use lists and parent lists. They are a simpler way of achieving the same result.

This keyword is used to define a method. A method is a special kind of handler that exists inside a factory script or XObject and that has its own special syntax. It uses Lingo to create expressions that are commands or functions. A method is a script, or series of scripts, that handle different messages (or processes) for objects created by a factory, or XObject.

There are two kinds of objects: internal (created by factories) and external (created by XObjects). Factories and XObjects use methods. The difference is that you define a factory's methods in the movie script or a cast member script, but an XObject's methods are predefined in the XObject itself. To see an XObject's methods, type `XObjectname(mdescribe)` in the message window.

Each object has its own set of messages created by its methods. Messages are the way objects communicate with each other and with the rest of Lingo. Messages are sent by an object's methods and provide all of the necessary functionality for each particular object's task. Methods are associated with the objects created by their factory or XObject. Each object can use all the methods in its factory or XObject.

A method is defined using the `method` keyword:

```
method messagename
```

For ease of reference, it is a good convention to begin the value you substitute for *messagename* with a lowercase m.

Related topics: exit, factory, instance, return keywords; mNew predefined method

mGet (Lingo)

Syntax: *object*`(mGet, whichElement )`

Methods were used for managing arrays in earlier versions of Director. In Director 4, it is recommended that you use lists. They are a simpler way of achieving the same result.

This predefined method (which can only be used with factory-produced objects) retrieves data from an object's internal array. Every object produced by a factory has an associated array capable of storing an arbitrary number of integers, floating-point numbers, strings, objects, or symbols. The elements of the array are numbered 1, 2, 3, ... The `mPut` predefined method is used to assign values to a particular element.

The integer expression *whichElement* specifies which array element the `mGet` method returns. If you retrieve an element that has not been assigned a value with the `mPut` method, the element has the numerical value 0.

Different types of data can be stored in various elements of the same array. You can use the functions `floatP`, `integerP`, `objectP`, `stringP`, and `symbolP` to determine the data type of a particular element.

Example

Note: Different types of data can be stored in various elements of the same array. You can use the functions integerP, floatP, stringP, symbolP, and objectP to determine the data type of a particular element.

Related topic: mPut predefined method

Lingo example: mGet

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

These first three statements use mPut to put data into an internal array. Using 3, 7, and 12 assigns these values to the third, seventh, and twelfth elements of the array:

```
put FactoryName (mNew) into myObject

myObject(mPut, 3, 2 + 2)

myObject(mPut, 7, sqrt(2.0))

myObject(mPut, 12, "hello" && "there")
```

This statement displays the value associated with the third element of the array:

```
put myObject(mGet, 3)
```

The result is 4, which is the equivalent of 2 + 2.

This statement displays the value associated with the seventh element of the array:

```
put myObject(mGet, 7)
```

The result is 1.4142, which is the equivalent of the square root of 2.

This statement displays the value associated with the twelfth element of the array:

```
put myObject(mGet, 12)
```

The result is "hello there", which is the value that was assigned in the first example.

min (Lingo)

Syntax: `min(`*list* `)`

or

`min (`*a1, a2, a3...*`)`

This function specifies the minimum value in the list specified by *list*.

Example

Related topic: max function

Lingo example: min

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler assigns the variable vLowest the minimum value in the list vBids, which consists of [#Castle:600, #Shields:750, #Wang:230]. The result is then inserted in the content of the text cast member Sorry:

```
on findLowest vBids

   set vLowest = min(vBids)

   set the text of "Sorry" = ¬

   "We're sorry, your bid of $" & vLowest && "is not a

   winner!"

end
```

mInstanceRespondsTo (Lingo)

Syntax: *XObject*`(mInstanceRespondsTo, `*message*`)`

This predefined method can only be used with XObjects. It returns a positive integer if an instance of the XObject responds to the specified message, which must be a string or symbol expression. In this case the integer returned is the number of arguments required by the message, plus 1. The method returns 0 if the XObject does not respond to the specified message.

Example

Related topics: mRespondsTo predefined method

Lingo example: mInstanceRespondsTo

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether the SerialPort XObject responds to the message string mWrite.

```
if SerialPort(mInstanceRespondsTo, "mWrite")¬
>0 then
   alert "serialPort XObject support is the¬
   mWrite method"
```

The result is 2; one for the first parameter, plus one.

mMessageList (Lingo)

Syntax: *XObject*`(mMessageList)`

This predefined method can only be used with XObjects. It returns a string that describes the XObject and its methods. The string is the same string that the `mDescribe` method displays in the message window; however, it may be put into fields or variables.

Example

Related topics: mDescribe, mInstanceRespondsTo, mRespondsTo predefined methods

Lingo example: mMessageList

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement displays methods and comments assigned to the fileIO XObject:

```
put fileIO(mMessageList)
```

mName (Lingo)

Syntax: `XObject(`*mName*`)`

or

`XObjectInstance (`*mName*`)`

This predefined method (which can be used only with XObjects and their instances) gives a string that contains the name of the XObject that created the instance.

Example

Related topics: factory function

Lingo example: mName

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

These statements create an instance of the serial port XObject and places it in the variable modemPort. It then displays the name of the XObject instance:

```
put SerialPort(mNew, 0) into modemPort

put modemPort(mName)
```

The result is SerialPort, which is the name of the XObject.

mNew (Lingo)

Syntax: `factory(mNew [,`*argument1*`] [,`*argument2*`] ...`

or

`XObject(mNew [,`*argument1*`] [,`*argument2*`] ...`

This predefined method is used to create factory objects or instances of an external XObject in RAM. To create the instance of a particular class of objects, you assign an object variable to the particular factory or XObject name using the mNew method.

Arguments to the `mNew` method are optional. Of course, a particular XObject may have been written to require a certain number of arguments of a certain type. See its documentation, its example movie, or its `mDescribe` in the message window for this information.

There is no requirement for any particular number of arguments to the mNew method of factory objects. Typically you use the `mNew` method to assign instance variables used throughout the methods of a factory object.

In order to clear the object you create using `mNew` from RAM at the end of the movie, it is a good idea to use the predefined mDispose method for both factory objects and external XObjects.

Before creating new instances of an object using `mNew`, it is also a good idea to check for previous instances of an object that has the same name, and `mDispose` it before you create a new one. In this way, if the movie is aborted before the normal `mDispose`, you won't fill up RAM by repeatedly creating new ones. This can happen during the development of a project, when you repeatedly stop the movie before the end, and play it again from the beginning.

Related topics: factory, method, instance keywords; mDescribe and mDispose predefined methods

Lingo example: mNew

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement creates a new instance of myArrayFactory. The new instance is named myArray:

```
put myArrayFactory(mNew) into myArray
```

This statement creates a new instance of birdFac. The new instance is named bird and has initial instance variables wingCastNum and legCastNum:

```
put birdFac(mNew, wingCastNum, legCastNum) into bird
```

This statement creates a new instance of the XObject PioneerLaserDisc. The new instance is named vDisc:

```
put PioneerLaserDisc(mNew, 1, 9600, 0) into vDisc
```

This handler checks for existing instances of factories and XObjects and disposes of any it finds. It then creates a new instance of the myArrayFactory:

```
on startMovie
   global myObject
      -- check for previous instances:
      if objectP(myObject) then myObject(mDispose)
      -- create a new instance of the object in RAM:
      put myArrayFactory(mNew) into myObject
end startMovie
```

mod (Lingo)

Syntax: *integerExpression1* `mod` *integerExpression2*

This arithmetic operator performs the arithmetic modulus operation on two integer expressions. In this operation, *integerExpression1* is divided by *integerExpression2*. The resulting value of the entire expression is the integer remainder of the division.

This is an arithmetic operator with a precedence level of 4.

Example

Lingo example: mod

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement divides 7 by 4 and then displays the remainder in the message window:

```
put 7 mod 4
```

The result is 3.

This handler sets the ink effect of all odd-numbered sprites to copy, which is the ink effect specified by the number 0. First, the handler checks whether the sprite that has the number in the variable mySprite is an odd-numbered sprite by dividing the sprite number by 2 and then checking whether the remainder is 1. When the remainder is 1, which is the result for an odd-numbered number, the ink effect is set to copy:

```
on setInk
if (mySprite mod 2) = 1 then
      set the ink of sprite mySprite to 0
   else
      set the ink of sprite mySprite to 8
   end if
end setInk
```

modal of window (Lingo)

Syntax: `the modal of window "`*window*`"`

This window property specifies whether movies can respond to events that occur outside the window specified by *window*.

- When the `modal of window` property is TRUE, movies cannot respond to events outside the window.

- When the `modal of window` property is FALSE, movies can respond to events outside the window.

Setting the modal of window to TRUE lets you define that a movie that plays in a window is the only movie that the user can interact with.

Lingo example: modal of window

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement lets movies respond to events outside of the window Demo:

```
set the modal of window "Demo" to FALSE
```

modified of cast (Lingo)

Syntax: `the modified of cast` *castMember*

This function indicates whether the cast member specified by *castMember* has been modified since it was read in from the movie file.

- When `the modified of cast` is TRUE (1), the cast member has been modified since it was read from the movie file.

- When `the modified of cast` is FALSE (0), the cast member has not been modified since it was read from the movie file.

Example

Lingo example: modified of cast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement tests whether the cast member Introduction Text has been modified since it was read from the movie file:

```
put the modified of cast "Introduction Text"
```

The result is 0, which is the numerical equivalent of FALSE.

mouseCast (Lingo)

Syntax: `the mouseCast`

This integer function gives the cast number of the sprite that is under the cursor when the function is called. When the cursor is not over a cast member, it gives the result -1.

This is useful for having the movie perform specific actions when the cursors rolls over a sprite and the sprite uses a certain cast member.

Example

Related topics: castNum of sprite sprite property; mouseChar, mouseItem, mouseLine, mouseWord, and rollOver functions; the number of cast cast property

Lingo example: mouseCast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether the cast member Off Limits is the cast member assigned to the sprite under the cursor and displays an alert if it is. This is one example of how you can specify an action depending on which cast member is assigned to the sprite:

```
if the mouseCast = the number of cast "Off Limits"¬
   then alert "Stay away from there!"
```

This statement assigns the number of the sprite under the cursor to the variable lastCast:

```
put the mouseCast into lastCast
```

mouseChar (Lingo)

Syntax: `the mouseChar`

This integer function, used for text sprites, gives the number of the character that is under the cursor when the function is called. The count is from the beginning of the field. If the mouse is not over a field or is in the gutter of a field, the result is -1.

Example

Related topics: mouseItem, mouseLine and mouseWord functions; char...of chunk expression keyword; the number of chars in chunk function

Lingo example: mouseChar

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement determines whether the cursor is not over a text sprite and changes the content of the text cast member Instructions to "Please point to a character." when it is:

```
if the mouseChar = -1 then ¬
   put "Please point to a character." ¬
   into field "Instructions"
```

This statement assigns the character under the cursor in the specified text field to the variable currentChar:

```
put char (the mouseChar) of field (the mouseCast) ¬
   into currentChar
```

mouseDown (Lingo)

Syntax: `the mouseDown`

This function indicates whether the mouse button is currently being pressed.

- When `the mouseDown` is TRUE, the button is being pressed.
- When `the mouseDown` is FALSE, the button is not being pressed.

Example

Related topics: mouseH, mouseUp, and mouseV functions; on mouseDown and on mouseUp event handlers

Lingo example: mouseDown

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler has the movie beep until the user clicks the mouse button:

```
on enterFrame
   repeat while the mouseDown = FALSE
      beep
   end repeat
```

This statement has Lingo exit the repeat loop or handler it is in when the user clicks the mouse button:

```
if the mouseDown then exit
```

mouseDownScript (Lingo)

Syntax: `the mouseDownScript`

This property specifies the Lingo that is executed when the mouse button is pressed. The Lingo can be a simple statement or a calling script for a handler.

When the mouse button is pressed and the `mouseDownScript` is defined, Lingo executes the instructions specified for the `mouseDownScript` first. Unless the instructions include the pass command so that the mouseDown message can pass on to other objects in the movie, no other on mouseDown handlers are executed.

Setting the `mouseDownScript` property does the same as using the when `keyDown` then command that appeared in earlier versions of Director.

When the instructions you've specified for the `mouseDownScript` property are no longer appropriate, turn them off by using the statement set the `mouseDownScript` to empty.

The `mouseDownScript` property can be tested and set, and the default value is EMPTY, which means that the `mouseDownScript` has no Lingo at all assigned to it.

Example

Related topics: dontPassEvent command; mouseUpScript property; on mouseDown and on mouseUp event handlers

Lingo example: mouseDownScript

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the mouseDownScript to if the mouseDown then go to next. When this is in effect and the user clicks the mouse button, the playback head always jumps to the next marker in the movie:

```
set the mouseDownScript ¬

   to "if the mouseDown then go to next"
```

This statement sets the mouseDownScript so that if the user clicks anywhere on the stage, the computer beeps. When this is in effect and the user clicks anywhere on the stage, the computer beeps:

```
set the mouseDownScript ¬

   to "if the clickOn = 0 then beep"
```

mouseH (Lingo)

Syntax: `the mouseH`

This function indicates the horizontal position of the mouse cursor. The value of `mouseH` is the number of pixels the cursor is from the left edge of the stage. Positions to the left of the stage are given as negative values.

The `mouseH` function is useful for moving sprites to the horizontal position of the mouse cursor and checking whether the cursor is within a region of the stage. Using `mouseH` and `mouseV` functions together, you can determine the cursor's exact location.

The `mouseH` function can be tested but not set.

Example

Related topics: locH of sprite and locV of sprite sprite properties; mouseV function

Lingo example: mouseH

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler moves sprite 10 to the mouse cursor location and updates the stage when the user clicks the mouse button:

```
on mouseDown

   set the locH of sprite 1 to the mouseH

   set the locV of sprite 1 to the mouseV

   updateStage

end
```

Example:
This statement tests whether the cursor is more than ten pixels to the right or left of a starting point and sets the variable Far to TRUE if it is:

```
if abs(the mouseH - startH) > 10 then ¬

   put TRUE into draggedEnough
```

See also locH and locV sprite properties; mouseV function

mouseItem (Lingo)

Syntax: `the mouseItem`

This integer function gives the number of the item that is under the pointer when the function is called and the cursor is over a text sprite. (An item is any sequence of characters delimited by commas.) Counting starts at the beginning of the field. If the mouse is not over a field, the result is -1.

Example

Related topics: item...of chunk expression keyword; mouseChar, mouseLine, and mouseWord functions; the number of items in chunk function

Lingo example: mouseItem

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement determines whether the cursor is over a text sprite and changes the content of the text cast member Instructions to "Please point to an item." when it is not:

```
if the mouseItem = -1 then ¬
   put "Please point to an item." ¬
   into field "Instructions"
```

This statement assigns the item under the cursor in the specifed text field to the variable currentItem:

```
put item (the mouseItem) of field (the mouseCast) ¬
   into currentItem
```

mouseLine (Lingo)

Syntax: `the mouseLine`

This integer function gives the number of the line under the pointer when the function is called and the cursor is over a text sprite. Counting starts at the beginning of the field. Lines are delimited by carriage returns, not soft line breaks. When the mouse is not over a text sprite, the result is -1.

Example

Related topics: mouseChar, mouseItem, and mouseWord functions; line...of chunk expression keyword; the number of lines in chunk function

Lingo example: mouseLine

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement determines whether the cursor is over a text sprite and changes the content of the text cast member Instructions to "Please point to a line." when it is not:

```
if the mouseLine = -1 then ¬
   put "Please point to a line." ¬
   into field "Instructions"
```

This statement assigns the number of the item under the cursor in the specifed text field to the variable currentLine:

```
put line (the mouseLine) of field (the mouseCast) ¬
   into currentLine
```

mouseUp (Lingo)

Syntax: `the mouseUp`

This function indicates whether the mouse button is being pressed.

- The `mouseUp` function is TRUE when the mouse button is not being pressed.
- The `mouseUp` function is FALSE when the mouse button is being pressed.

Example

Related topics: mouseDown, mouseH, and mouseV functions; on mouseDown and on mouseUp event handlers

Lingo example: mouseUp

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler has the movie beep until the user clicks the mouse button:

```
on enterFrame
   repeat while the mouseUp = FALSE
      beep
   end repeat
end enterFrame
```

This statement has Lingo exit the repeat loop or handler it is in when the user clicks the mouse button:

```
if the mouseUp then exit
```

mouseUpScript (Lingo)

Syntax: `the mouseUpScript`

This property determines the Lingo that is executed when the mouse button is released. The Lingo can be a simple statement or a statement that calls a handler.

When a key is released and the `mouseUpScript` is defined, Lingo executes the instructions specified for the `mouseUpScript` first. Unless the instructions include the pass command so that the `mouseUp` message can pass on to other objects in the movie, no other `mouseUp` handlers are executed.

When the instructions you've specified for the `mouseUpScript` property are no longer appropriate, turn them off by using the statement `set the mouseUpScript to empty`.

```
when mouseUp then nothing
```

Setting the `mouseDownScript` property does the same as using the `when keyDown then` command that appeared in earlier versions of Director.

The `mouseUpScript` property can be tested and set. The default value is EMPTY.

Example

Related topics: dontPassEvent command; mouseDownScript property; on mouseDown and on mouseUp event handlers

Lingo example: mouseUpScript

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the mouseUpScript to continue. When this is in effect and the movie is paused, the movie always continues whenever the user releases the mouse button:

```
set the mouseUpScript to "continue"
```

This statement has the movie beep when the user releases the mouse button after clicking anywhere on the stage:

```
set the mouseUpScript ¬
   to "if the clickOn = 0 then beep"
```

mouseV (Lingo)

Syntax: `the mouseV`

This function indicates the vertical position of the mouse cursor. The value of `mouseV` is the number of pixels the cursor is from the top of the stage.

The `mouseV` function is useful for moving sprites to the vertical position of the mouse cursor and checking whether the cursor is within a region of the stage. Using `mouseH` and `mouseV` functions together, you can identify the cursor's exact location.

Example

Related topics: mouseH function; locH of sprite and locV of sprite sprite properties

Lingo example: mouseV

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler moves sprite ten to the mouse cursor location and updates the stage when the user clicks the mouse button:

```
on mouseDown
   set the locH of sprite 10 to the mouseH
   set the locV of sprite 10 to the mouseV
   updateStage
end
```

This statement tests whether the cursor is more than ten pixels above or below a starting point and sets the variable vFar to TRUE if it is:

```
if abs(the mouseV - startV) > 10 then ¬
   put TRUE into draggedEnough
```

mouseWord (Lingo)

Syntax: `the mouseWord`

This integer function gives the number of the word under the cursor when the function is called and when the cursor is over a text sprite. Counting starts from the beginning of the field. When the mouse is not over a field, the result is -1.

Example

Related topics: mouseChar, mouseItem, and mouseLine functions; the number of words in in chunk function; word...of chunk expression keyword

Lingo example: mouseWord

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement determines whether the cursor is over a text sprite and changes the content of the text cast member Instructions to "Please point to a word." when it is not:

```
if the mouseWord = -1 then ¬
   put "Please point to a word." ¬
   into field "Instructions"
```

This statement assigns the number of the word under the cursor in the specified text field to the variable currentWord:

```
put word (the mouseWord) of field (the mouseCast) ¬
   into currentWord
```

move cast (Lingo)

Syntax: `move cast` *whichCastmember* `[, cast` *whichLocation*`]`

This command moves the cast member specified by *whichCastmember* to a different location in the cast window.

- Using the `move` cast command without the optional parameter, the cast member moves to the first empty location in the cast window.

- Including the `cast` *whichLocation* parameter in the move cast command moves the cast member to the location specified by *whichLocation*.

Director does not automatically update the cast member numbers assigned to sprites when you move cast members in the cast window.

Example

Lingo example: move cast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement moves cast member Shrine to the first empty location in the cast window:

```
move cast "Shrine"
```

This statement moves cast member Shrine to location 20 in the cast window:

```
move cast "Shrine", cast 20
```

moveableSprite of sprite (Lingo)

Syntax: `the moveableSprite of sprite` *whichSprite*

This sprite property indicates whether a sprite is moveable.

- When the sprite can be moved by the user, `the moveableSprite of sprite` is TRUE (1).
- When the sprite cannot be moved by the user, `the moveableSprite of sprite` is FALSE (0).

To use Lingo to make a text sprite moveable, the sprite must first be a puppet sprite.

You can also make a sprite moveable by using the Moveable option in the score. However, controlling whether a sprite is moveable by using Lingo lets you turn this condition on and off as situations in the movie require. For example, referring to the "Mechanical Simulation" sample movie, you could let the user drag parts from the toolkit but make them unmoveable after they are on the pegboard by turning `moveableSprite of sprite` on and off at the appropriate times.

Setting `the moveableSprite of sprite` property lets you control whether sprites are moveable from other scripts.

The `moveableSprite of sprite` property can be tested and set.

Example

Related topic: puppetSprite command

Lingo example: moveableSprite of sprite

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler first makes the sprite a puppet and then makes it moveable:

```
on spriteMove
   puppetSprite 5, TRUE
   set the moveableSprite of sprite 5 to TRUE
end
```

This statement checks whether a sprite is moveable and displays a message if it isn't:

```
if the moveableSprite of sprite 13 = FALSE ¬
   then set the text of cast "Notice" to ¬
   "You can't drag this item by using the mouse."
```

moveToBack (Lingo)

Syntax: `moveToBack window "`*whichWindow*`"`

This command moves the window specified by *whichWindow* behind all other windows.

Example

Lingo example: moveToBack

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement moves the window Demo Window behind all other windows:

```
set myWindow=getat(the windowList, 1,1)

moveToBack window myWindow
```

moveToFront (Lingo)

Syntax: `moveToFront window "`*whichWindow*`"`

This command moves the window specified by *whichWindow* in front of all other windows.

Example

Lingo example: moveToFront

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement moves the window Demo Window in front of all other windows:

```
set myWind=getat(the windowList, 1,1)

moveToFront window myWind
```

movie (Lingo)

Syntax: `the movie`

This string function returns the name of the currently open movie.

Example

Related topics: go and play commands; pathName function

Lingo example: movie

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement assigns the name of the current movie to the text field Current Movie:

```
put the movie into field "Movie Name"
```

movieFileFreeSize (Lingo)

Syntax: `the movieFileFreeSize`

This function returns the amount of unused space in the current movie in bytes.

Movies saved with the Save And Compact command or the Save As command do not have any unused space. This function will return 0.

movieFileSize (Lingo)

Syntax: `the movieFileSize`

This function returns the size of the current file in bytes.

movieName (Lingo)

Syntax: `the movieName`

This function indicates the simple name of the current movie. The `movieName` function is equivalent to the movie function.

Example

Related topics: movie, moviePath, pathName functions

Lingo example: movieName

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement displays the name of the current movie in the message window:

```
put the movieName
```

moviePath (Lingo)

Syntax: `the moviePath`

This function indicates the pathname of the folder that the current movie is located in. The `moviePath` function always gives the pathname in the format used by whichever platform Director is running on. The `moviePath` function is equivalent to the `pathName` function.

Example

Related topics: movie, movieName, and pathName functions

Lingo example: moviePath

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement displays the pathname of the current movie's folder:

```
put the moviePath
```

movieRate of sprite (Lingo)

Syntax: `the movieRate of sprite` *channelNumber*

This sprite property controls the rate at which a digital video movie in a specific channel plays. The movie rate is a value specifying the playback of the digital video movie. A value of 1 is normal forward play, -1 is reverse, 0 is stop.

Example

Lingo example: movieRate of sprite

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the rate for a digital video movie in sprite channel 9 to normal playback speed:

```
set the movieRate of sprite 9 to 1
```

This statement has the digital video movie in sprite channel 9 play in reverse:

```
set the movieRate of sprite 9 to -.1
```

movieTime of sprite (Lingo)

Syntax: `the movieTime of sprite` *channelNumber*

This sprite property determines the current time of a digital video movie playing in the channel specified by *channelNumber*. The value of `the movieTime` is measured in ticks.

The `movieTime of sprite` property can be tested and set.

Example

Lingo example: movieTime of sprite

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement displays the current time of the digital video movie in channel 9 in the message window:

```
put the movieTime of sprite 9
```

This statement sets the current time of the digital video movie in channel 9 to the value in the symbol #Poster:

```
set the movieTime of sprite 9 to #Poster
```

mPerform (Lingo)

Syntax: *object* `(mPerform, ` *message* `[, ` *argument1*`][, ` *argument2*`]...)`

This predefined method only works with XObjects and factory objects. Factories were supported in earlier versions of Director. In Director 4, it is recommended that you use list and parent scripts instead of factories. They are a simpler method of achieving the same result.

This predefined method is similar to the Lingo `do` command, which executes a Lingo statement stored as a string. However, `mPerform` invokes a particular method of the specified object by sending that message to the object indirectly.

This is accomplished as follows: The first argument to `mPerform` is a required argument called a "message expression." This expression can be either in the form of either a string or symbol. This message specifies the name of the method to be invoked by the `mPerform` message.

Optional additional arguments, which can be any data type, constant, or property used in the method to be invoked, follow this required first argument.

Typically, the object name is specified by use of the me keyword, since the typical use of `mPerform` is within a factory method that invokes one of several other methods.

A powerful use for `mPerform` is to eliminate a lot of `if...then` conditional tests within methods that call other methods.

Example

Related topics: factory, me, and method keywords

Lingo example: mPerform

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement creates an instance named modemPort of the SerialPort XObject:

```
put SerialPort(mNew, 0) into modemPort
```

These statements invoke the mWriteChar method with the argument charNum:

```
modemPort(mPerform, "mWriteChar", charNum)
```

```
modemPort(mWriteChar, charNum)
```

mPut (Lingo)

Syntax: *object*`(mPut, whichElement, expression)`

This predefined method, was used for managing arrays in earlier versions of Director. In Director 4, it is recommended that you use lists to manage arrays. Lists are a simpler means of achieving the same result.

This predefined method, which can only be used with factory-produced objects, puts data into an object's internal array. Every object produced by a factory has an associated array capable of storing an arbitrary number of integers, floating-point numbers, strings, objects, or symbols. The elements of the array are numbered 1, 2, 3, ... The `mGet` predefined method is used to retrieve values from a particular element.

The integer expression *whichElement* specifies which array element the `mPut` method assigns. The value of *expression* is assigned to the specified element.

Example

Related topics: mGet predefined method

Lingo example: mPut

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

These first three statements use mPut to put data into an internal array. Using 3, 7, and 12 assigns these values to the third, seventh, twelfth elements of the array:

```
myObject(mPut, 3, 2 + 2)

myObject(mPut, 7, sqrt(2.0))

myObject(mPut, 12, "hello" && "there")
```

This statement displays the value associated with the third element of the array:

```
put myObject(mGet, 3)
```

The result is 4.
Example:
This statement displays the value associated with the seventh element of the array:

```
put myObject(mGet, 7)
```

The result is 1.4142, which is the square root of 2.

This statement displays the value associated with the twelfth element of the array:

```
put myObject(mGet, 12)
```

The result is the string "hello there".

mRespondsTo (Lingo)

Syntax: *XObjectInstance* `(mRespondsTo, message)`

This predefined method, which can only be used with instances of XObjects, returns a positive integer when *XObjectInstance* responds to the specified message, which must be a string or symbol expression. In this case the integer returned is the number of arguments required by the message, plus 1. The method returns 0 if *XObjectInstance* does not respond to the specified message.

Example

Related topics: mInstanceRespondsTo predefined method

Lingo example: mRespondsTo

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

These statements create an instance of the XObject SerialPort and checks whether it responds to the message string mWrite.

```
put SerialPort(mNew, 0) into modemPort

put modemPort(mRespondsTo, "mWrite")
```

multiSound (Lingo)

Syntax: `the multiSound`

This system property is TRUE when the computer supports more than one sound channel. (A PC must have a multichannel sound card for the `multiSound` property to be TRUE.)

Example

Lingo example: multisound

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement plays the sound file Music in sound channel 2 if the computer supports more than one sound channel:

```
if the multiSound sound playFile 2, "Music"
```

name of cast (Lingo)

Syntax: `the name of cast` *whichCastmember*

This cast property determines the name of the specified cast member.

- When *whichCastmember* evaluates to a string, it is used as the cast name.

- When *whichCastmember* evaluates to an integer, it is used as the cast number. (The cast identifiers A11 through H88 evaluate to integer cast numbers.)

The name is a descriptive string assigned by the user. Setting this property is equivalent to entering a name in the Cast Member Info dialog box.

The `name` cast property can be tested and set.

Related topics: the number of cast cast property

Lingo example: name of cast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement changes the name of cast member named On to Off:

```
set the name of cast "On" to "Off"
```

This statement sets the name of cast member 15 to Background Sound:

```
set the name of cast 15 to "Background Sound"
```

This statement sets the variable itsName to the name of the cast member that follows the cast member whose number is equal to the variable i:

```
put the name of cast (i + 1) into itsName
```

name of menu (Lingo)

Syntax: `the name of menu` *whichMenu*

This menu property returns a string containing the name of the specified menu. The expression *whichMenu* can evaluate to either a menu number or a menu name.

The `name of menu` property can be tested but cannot be set directly. Use the `installMenu` command to set up a custom menu bar.

Example

Related topics: number of menus property; name of menuItem menu item property

Lingo example: name of menu

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement assigns the name of menu number 1 to the variable firstMenu:

```
put the name of menu 1 into firstMenu
```

The following handler returns a list of menu names, one per line:

```
on menuList
   put EMPTY into list
   repeat with i = 1 to the number of menus
      put the name of menu i & RETURN after list
   end repeat
   return list
end
```

name of menuItem (Lingo)

Syntax: `the name of menuItem` *whichItem* `of menu` *whichMenu*

This menu item property determines the text that appears in the menu item specified by *whichItem* in the menu specified by *whichMenu*. The *whichItem* expression can be either a menu item name or a menu item number; *whichMenu* can be either a menu name or a menu number.

The `name of menuItem` property can be tested and set.

Example

Related topics: name of menu item property; number of menuItems property

Lingo example: name of menuItem

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the variable itemName to the name of the eighth item in the Edit menu:

```
put the name of menuItem 8 of menu "Edit" ¬
   into itemName
```

This statement has a specific filename follow the term Open in the File menu:

```
set the name of menuItem "Open" of menu fileMenu ¬
   to "Open" & fileName
```

next (Lingo)

Syntax: `next`

This keyword refers to the next marker in the movie. The `next` keyword is equivalent to the phrase `the marker (+ 1)`.

Example

Related topics: loop and previous keywords

Lingo example: next

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sends the playback head to the next marker in the movie:

```
go next
```

next repeat (Lingo)

Syntax: `next repeat`

This keyword causes Lingo to go to the next step in a repeat loop in a script. This is different from the exit repeat keyword.

This repeat loop displays only odd numbers in the message window:

```
repeat with i = 1 to 10

   if (i mod 2) = 0 then next repeat

   put i

end repeat
```

not (Lingo)

Syntax: `not` *logicalExpression*

This logical operator performs a logical negation on a logical expression.

- When the expression specified by *logicalExpression* is TRUE, the result is FALSE (0).
- When the expression specified by *logicalExpression* is FALSE, the result is TRUE (1).

This is a logical operator with a precedence level of 5.

Example

Related topics: and and or logical operators

Lingo example: not

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement determines whether 1 is not less than 2:

```
put not (1 < 2)
```

Because 1 is less than 2, the result is 0, which indicates that the expression is FALSE.

This statement determines whether 1 is not greater than 2:

```
put not (1 > 2)
```

Because 1 is not greater than 2, the result is 1, which indicates that the expression is TRUE.

This handler sets the checkMark of menuItem for the item Bold in the Style menu to the opposite of its current setting:

```
on resetMenuItem
   set the checkMark of menuItem "Bold" ¬
      of menu "Style" to not (the checkMark ¬
      of menuItem "Bold" of menu "Style")
end
```

nothing (Lingo)

Syntax: `nothing`

This command does nothing at all. It is useful for making the logic of an if...then statement more obvious. Also, a nested `if...then...else` statement that contains no explicit command for the `else` clause may require `else nothing`. Otherwise, Lingo interprets the `else` clause as part of the preceding `if`.

Example

Related topic: if...then keywords

Lingo example: nothing

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

The nested if...then...else statement in this handler uses the nothing command to satisfy the statement's else clause:

```
on mouseDown
   if the clickOn = 1 then
      if the moveable of sprite 1 = TRUE ¬
      then set the text of cast "Notice" = ¬
      "Drag the ball"
      else nothing
   else set the text of cast "Notice" = ¬
      "Click again"
   end if
end mouseDown
```

This handler has the movie do nothing as long as the mouse button is being pressed:

```
on mouseDown
   repeat while the stillDown
      nothing
   end repeat
end mouseDown
```

number of cast (Lingo)

Syntax: `the number of cast` *whichCastmember*

This cast property indicates the cast number of the cast member specified by *whichCastmember*.

- When *whichCastmember* is a string, the string is used as the cast member name.
- When *whichCastmember* is an integer, the integer is used as the cast member number.
- When *whichCastmember* is an octal number (octal numbers were used in earlier versions of Director), *whichCastmember* is replaced with the decimal equivalent of the octal number. The first cast member, A11 becomes cast number 1; A12 becomes cast number 2; and so on.

The `number of cast` property can be tested, but not set.

Related topics: castNum of sprite sprite property; number of castmembers property

Lingo example: number of cast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement assigns the cast number of the cast member Power Switch to the variable whichCastmember:

```
put the number of cast "Power Switch" into ¬

   whichCastmember
```

This statement assigns the cast member Red Balloon to sprite 1:

```
set the castNum of sprite 1 ¬

   to the number of cast "Red Balloon"
```

number of castmembers (Lingo)

Syntax: `the number of castMembers`

This property indicates the number of the last cast member, including the cast members in the shared cast, in the current movie. Some of the cast member slots may be empty, so the actual number of cast members may be fewer than the number of cast members value.

The `number of castMembers` property can be tested, but not set.

Example

Related topics: the number of cast cast property

Lingo example: number of castmembers

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

The following handler returns a string containing a list of all the cast member names, one per line:

```
on castList
   put EMPTY into list
   repeat with i = 1 to the number of castMembers
      put the name of cast i & RETURN after list
      end repeat
   return list
end
```

number of chars in (Lingo)

Syntax: `the number of chars in` *chunkExpression*

This chunk function returns a count of the characters in a chunk expression.

Chunk expressions refer to any character, word, item, or line in any container of text. Containers include fields (text cast members) and variables that hold strings, and specified characters, words, items, lines, and ranges in containers.

Spaces and control characters such as Tab and Return count as characters.

Example

Note: Spaces count as characters. So do control characters like `TAB` and `RETURN`.

Related topics: length function; number of items in, number of lines in, number of words in in chunk functions; char...of keyword

Lingo example: number of chars in

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement displays the number of characters in the string "Macromedia, the multimedia company" in the message window:

```
put the number of chars ¬
   in "Macromedia, the multimedia company"
```

The result is 33.

This statement sets the variable charCounter to the number of characters in the ith word in the string Names:

```
put the number of chars in word i of "Names" into
   charCounter
```

number of items in (Lingo)

Syntax: `the number of items in` *chunkExpression*

This chunk function returns a count of the items in a chunk expression. An item chunk is any sequence of characters delimited by commas.

Chunk expressions refer to any character, word, item, or line in any container of text. Containers include fields (text cast members) and variables that hold strings, and specified characters, words, items, lines, and ranges in containers.

Example

Related topics: item...of chunk expression keyword, number of chars in, number of lines in, number of words in in chunk functions

Lingo example: number of items in

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement displays the number of items in the string "Macromedia, the multimedia company" in the message window:

```
put the number of items ¬
   in "Macromedia, the multimedia company"
```

The result is 2.

This statement sets the variable itemCounter to the number of items in the field Names:

```
put the number of items in field "Names" into
   itemCounter
```

number of lines in (Lingo)

Syntax: `the number of lines in` *chunkExpression*

This chunk function returns a count of the lines in a chunk expression.

Chunk expressions are used to refer to any character, word, item, or line in any container of text. Containers include fields (text cast members) and variables that hold strings, and specified characters, words, items, lines, and ranges in containers.

Example

Related topics: line...of chunk expression keyword; number of chars in, number of items in, number of words in in chunk functions

Lingo example: number of lines in

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement displays the number of lines in the string "Macromedia, the multimedia company" in the message window:

```
put the number of lines ¬
   in "Macromedia, the multimedia company"
```

The result is 1.

This statement sets the variable lineCounter to the number of lines in the field Names:

```
put the number of lines in field "Names" into
   lineCounter
```

number of menuItems (Lingo)

Syntax: `the number of menuItems of menu` *whichMenu*

This menu property indicates the number of menu items in the custom menu specified by *whichMenu*. The *whichMenu* parameter can be a menu name or a menu number.

The `number of menuItems` menu property can be tested but not set directly. Use the `installMenu` command to set up a custom menu bar.

Example

Related topics: installMenu command; number of menus property

Lingo example: number of menuItems

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the variable fileItems to the number of menu items in the custom File menu:

```
put the number of menuItems of menu "File" ¬
   into fileItems
```

This statement sets the variable itemCount to the number of menu items in the custom menu 3:

```
put the number of menuItems of menu 3 into itemCount
```

number of menus (Lingo)

Syntax: `the number of menus`

This menu property indicates the number of menus installed in the current movie.

The `number of menus` menu property can be tested, but not set. Use the `installMenu` command to set up a custom menu bar.

Example

Related topics: installMenu command; number of menuItems property

Lingo example: number of menus

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement determines whether there are any custom menus installed in the movie and installs the menu Menubar if no menus are already installed:

```
if the number of menus = 0 then ¬

   installMenu (the number of cast "Menubar")
```

This statement has the message window display the number of menus that are in the current movie:

```
put the number of menus
```

number of words in (Lingo)

Syntax: `the number of words in` *chunkExpression*

This function tells how many words are in the chunk expression specified by *chunkExpression*.

Chunk expressions refer to any character, word, item, or line in any container of text. Containers include fields (text cast members) and variables that hold strings, and specified characters, words, items, lines, and ranges in containers.

Example

Related topics: number of chars in, number of items in, number of lines in chunk functions; word...of chunk expression keyword

Lingo example: number of words in

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement has the message window display the number of words in the string "Macromedia, the multimedia company":

```
put the number of words ¬
   in "Macromedia, the multimedia company"
```

The result is 4.

This handler reverses the order of words in the string specified by the argument wordList:

```
on reverse wordList
   put EMPTY into list
   repeat with i = 1 to the number of words ¬
      in wordList
      put word i of wordList & " " before list
   end repeat
   delete char (the number of chars in list) of list
   return list
end reverse wordList
```

numToChar (Lingo)

Syntax: `numToChar(`*integerExpression*`)`

This function gives a string containing the single character whose ASCII sequence number is the value of *integerExpression*. It is useful for interpreting data from outside sources that are presented as numbers rather than as characters.

Example

Related topics: charToNum function

Lingo example: numToChar

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement has the message window display the character whose ASCII number is 65:

```
put numToChar(65)
```

The result is the letter "A."

objectP (Lingo)

Syntax: `objectP(`*expression*`)`

This function indicates whether the expression specified by *expression* is an object produced by a parent script, factory, or XObject.

- When `objectP` is TRUE, the expression is such an object.
- When `objectP` is FALSE, the expression is not such an object.

The "P" in objectP stands for "predicate."

It is good practice to use `objectP` to determine which items are XObjects when you create XObjects by using `mNew` or disposing of XObjects by using `mDispose`.

Example

Related topics: floatP, integerP, stringP, and symbolP functions; mDispose and mNew predefined methods

Lingo example: objectP

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether modemPort is an XObject and displays the result in the message window:

```
put objectP(modemPort)
```

This handler checks whether externalFile is an XObject and disposes of it if it is:

```
on stopMovie   if objectP(externalFile) then ¬
   externalFile(mDispose)end stopMovie
```

of (Lingo)

The word of is part of many Lingo properties, such as the `foreColor of sprite`, `the number of cast`, `the name of menu`, and so on.

offset (Lingo)

Syntax: `offset(`*stringExpression1*`, `*stringExpression2*`)`

This function tells the number of the position in *stringExpression2* where the first character of *stringExpression1* first occurs.

- When *stringExpression1* is found in *stringExpression2*, the result is the number that indicates the position of the first occurrence.

- When *stringExpression1* is not found in *stringExpression2*, the result is 0.

Lingo counts spaces as characters in both strings. The string comparison is not sensitive to case or diacritical marks. For example, Lingo considers "a" and "Å" the same character.

Example

Related topics: chars and length functions; contains and starts comparison operators

Lingo example: offset

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement has the message window display the beginning position of the string "media" within the string "Macromedia":

```
put offset("media","Macromedia")
```

The result is 6.

This statement has the message window display the beginning position of the string "Micro" within the string "Macromedia":

```
put offset("Micro", "Macromedia")
```

The result is 0, because "Macromedia" doesn't contain the string "Micro".

offset rect (Lingo)

Syntax: `offset (`*rectangle, horizontalChange, verticalChange* `)`

This function yields a rectangle that is offset from the rectangle specified by *rectangle*. The horizontal offset is the value specified by *widthChange*; the vertical offset is the value specified by *heightChange*.

- When *heightChange* is greater than zero, the offset is toward the top of the stage; when heightChange is less than zero, the offset is toward the bottom of the stage.

- When *widthChange* is greater than zero, the offset is toward the right of the stage; when *heightChange* is less than zero, the offset is toward the left of the stage.

The values for *heightChange* and *widthChange* are in pixels.

Lingo example: offset rect

This statement sets the variable newRect to a rectangle that is 100 pixels to the right and 150 pixels above the rectangle named oldRect:

```
put offset(oldRect, 100, 150) into newRect
```

on (Lingo)

Syntax: `on` *handlerName* `[`*argument1* `] [, ` *arg2* `] [, ` *arg3* `] ...`
`[`*statements* `]`
`end` *handlerName*

This keyword indicates the beginning of a handler. Handlers are collections of Lingo statements that you can execute by simply using the handler name. A handler can accept arguments as input values and return a value as a function result.

Handlers can be defined in score scripts, movie scripts, and scripts of cast members. A handler in a script of a cast member can only be called by other handlers in the same script. A handler in a score script or movie script can be called from anywhere.

You can use the same handler in more than one movie by putting the handler's script in the shared cast.

Related topics: on enterFrame, on exitFrame, on idle, on keyDown, on keyUp, on mouseDown, on mouseUp, on startMovie, on stepMovie, and on stopMovie event handlers

on enterFrame (Lingo)

Syntax:
```
on enterFrame
    statement(s)
end enterFrame
```

This event handler contains statements that are executed each time the playback head enters the frame that the `on enterFrame` handler is attached to. The `on enterFrame` handler is equivalent to the `on stepMovie` handler used in earlier versions of Director.

The `on enterFrame` event handler is a good place for Lingo that you want executed once at every new frame.

Place `on enterFrame` handlers in frame scripts or movie scripts.

- When you want to assign the handler to an individual frame, put the handler in the frame script.

- When you want to assign the handler to every frame unless you explicitly instruct the movie otherwise, put the `on enterFrame` handler in a movie script. The handler then executes every time the playback head enters a frame unless the frame script has its own `on enterFrame` handler. When the frame script has its own `on enterFrame` handler, the `on enterFrame` handler in the frame script overrides the one in the movie script.

Example

Related topics: on exitFrame, on idle, on keyDown, on keyUp, on mouseDown, on mouseUp, on startMovie, on stepMovie, and on stopMovie event handlers

Lingo example: on enterFrame

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler turns off the puppet condition for sprites 1 through 5 each time the playback head enters the frame:

```
on enterFrame  repeat with i = 1 to 5           puppetSprite i, FALSE   end repeatend
```

on exitFrame (Lingo)

Syntax:
```
on exitFrame
    statement(s)
end exitFrame
```

This event handler contains statements that are activated each time the playback head exits the frame that the `on exitFrame` handler is attached to. The `on exitFrame` handler is a useful place for Lingo that resets conditions that are no longer appropriate after leaving the frame.

Place `on exitFrame` handlers in frame scripts or movie scripts.

- When you want to assign the handler to an individual frame, put the handler in the frame script.

- When you want to assign the handler to every frame unless explicitly instructed otherwise, put the handler in a movie script. The `on exitFrame` handler then executes every time the playback head exits the frame unless the frame script has its own `on exitFrame` handler. When the frame script has its own `on exitFrame` handler, the `on exitFrame` handler in the frame script overrides the one in the movie script.

Example

Related topics: on enterFrame, on idle, on keyDown, on keyUp, on mouseDown, on mouseUp, on startMovie, on stepMovie, and on stopMovie event handlers

Lingo example: on exitFrame

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler turns off all puppetSprite conditions when the playback head exits the frame:

```
on exitFrame   repeat with i = 48 down to 1           set the puppet of
sprite i = FALSE  end repeatend
```

This handler sends the playback head to a specified frame if the value in the variable vTotal exceeds 1000 when the playback head exits the frame:

```
on exitFrame

   if vTotal > 1000 then go to frame "Finished"

end
```

on idle (Lingo)

Syntax:
```
on idle
    statement(s)
end idle
```

This event handler contains statements that are executed whenever the movie has no other events to handle.

This is a useful location for Lingo statement that you want to execute as frequently as possible. Some common cases are updating values in global variables and displays that tell current movie conditions.

Because statements in `on idle` handlers run frequently, it is good practice to avoid placing Lingo that takes a long time to process in an `on idle` handler.

Note: This is a good place for the Lingo statements that you want executed as frequently as possible.

Related topics: on enterFrame, on exitFrame, on keyDown, on keyUp, on mouseDown, on mouseUp, on startMovie, on stepMovie, and on stopMovie event handlers

Lingo example: on idle

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler updates the time being displayed in the movie whenever there are no other events to handle:

```
on idle

   put the short time into field "Time"

end
```

on keyDown (Lingo)

Syntax:
```
on keyDown
    statement(s)
end
```

This event handler contains statements that are activated when a key is pressed.

When a key is pressed, Lingo searches these locations in order for an `on keyDown` handler: primary event handler, editable text sprite script, script of a text cast member, frame script, and movie script. (For sprites and cast members, `on keyDown` handlers work only for editable text. A `keyDown` on a different type of cast member, such as a bitmap, has no effect.)

Lingo stops searching when it reaches the first location that has an `on keyDown` handler, unless the handler includes the pass command to explicitly pass the `keyDown` message on to the next location.

The `on keyDown` event handler is a good place to put Lingo that implements keyboard shortcuts or other interface features that you want to have occur when the user presses keys.

Where you place an `on keyDown` handler can affect when it runs.

- When you want the handler to apply to a specific editable text sprite, put the handler in a sprite script.
- When you want the handler to apply to an editable text cast member in general, put the handler in a script of the cast member.
- When you want the handler to apply to an entire frame, put the handler in a frame script.
- When you want the handler to apply throughout the entire movie, put the handler in a movie script.

You can override an `on keyDown` handler by placing an alternate `on keyDown` handler in a location that Lingo checks before it gets to the handler you want to override. For example, you can override an `on keyDown` handler assigned to a cast member by placing an `on keyDown` handler in a sprite script.

Example

Related topics: on enterFrame, on exitFrame, on idle, on keyUp, on mouseDown, on mouseUp, on startMovie, on stepMovie, and on stopMovie event handlers

Lingo example: on keyDown

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler checks whether the Return key was pressed and sends the playback head to another frame if it was:

```
on keyDown

   if the key = RETURN then go to frame "AddSum"

end keyDown
```

on keyUp (Lingo)

Syntax:
```
on keyUp
    statement(s)
end
```

This event handler contains statements that are activated when a key is released. The `on keyUp` handler is similar to the `on keyDown` handler.

When a key is released, Lingo searches these locations in order for an `on keyUp` handler: primary event handler, editable text sprite script, script of a text cast member, frame script, and movie script. (For sprites and cast members, `on keyUp` handlers only for editable text. A `keyUp` on a different type of cast member, such as a bitmap, has no effect.)

Lingo stops searching when it reaches the first location that has an `on keyUp` handler, unless the handler includes the pass command to explicitly pass the `keyUp` message on to the next location.

The `on keyUp` event handler is a good place to put Lingo that implements keyboard shortcuts or other interface features that you want to have occur when the user releases keys.

Where you place an `on keyUp` handler can affect when it runs.

- When you want the handler to apply to a specific editable text sprite, put the handler in a sprite script.
- When you want the handler to apply to an editable text cast member in general, put the handler in a script of the cast member.
- When you want the handler to apply to an entire frame, put the handler in a frame script.
- When you want the handler to apply throughout the entire movie, put the handler in a movie script.

You can override an `on keyUp` handler by placing an alternate `on keyUp` handler in a location that Lingo checks before it gets to the handler you want to override. For example, you can override an `on keyUp` handler assigned to a cast member by placing an `on keyUp` handler in a sprite script.

Related topics: on enterFrame, on exitFrame, on idle, on keyDown, on mouseDown, on mouseUp, on startMovie, on stepMovie, and on stopMovie event handlers

Lingo example: on keyUp

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler checks whether the Return key was released and sends the playback head to another frame if it was:

```
on keyUp

   if the key = RETURN then go to frame "AddSum"

end keyUp
```

on mouseDown (Lingo)

Syntax:
```
on mouseDown
    statement(s)
end
```

This event handler contains statements that are activated when the mouse button is pressed.

When the mouse button is pressed, Lingo searches these locations in order for an `on mouseDown` handler: primary event handler, sprite script, script of a cast member, frame script, and movie script. Lingo stops searching when it reaches the first location that has an `on mouseDown` handler, unless the handler includes the pass command to explicitly pass the `mouseDown` message on to the next location.

When you want to specify an fine

The `on mouseDown` event handler is a good place to put Lingo that flashes images, triggers sound effects, or makes sprites move when the user presses the mouse button.

Where you place an `on mouseUp` handler can affect when it runs.

- When you want the handler to apply to a specific sprite, put the handler in a sprite script.
- When you want the handler to apply to a cast member in general, put the handler in a script of the cast member.
- When you want the handler to apply to an entire frame, put the handler in a frame script.
- When you want the handler to apply throughout the entire movie, put the handler in a movie script.

You can override an `on mouseUp` handler by placing an alternate `on mouseUp` handler in a location that Lingo checks before it gets to the handler you want to override. For example, you can override an `on mouseUp` handler assigned to a cast member by placing an `on mouseDown` handler in a sprite script.

Example

Related topics: on enterFrame, on exitFrame, on idle, on keyDown, on keyUp, on mouseUp, on startMovie, on stepMovie, and on stopMovie event handlers

Lingo example: on mouseDown

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler checks whether the user clicks anywhere on the stage and sends the playback head to another frame if he or she does:

```
on mouseDown

   if the clickOn = 0 then go to frame "AddSum"

end mouseDown
```

This handler, assigned to a sprite script, plays a sound when the sprite is clicked:

```
on mouseDown   play "Crickets"end
```

on mouseUp (Lingo)

Syntax:
```
on mouseUp
    statement(s)
end mouseUp
```

This event handler contains statements that are activated when the mouse button is released.

When the mouse button is released, Lingo searches these locations in order for an `on mouseUp` handler: primary event handler, sprite script, script of a cast member, frame script, and movie script. Lingo stops searching when it reaches the first location that has an `on mouseUp` handler, unless the handler includes the `pass` command to explicitly pass the `mouseUp` message on to the next location.

An `on mouseUp` event handler is a good place to put Lingo that changes the appearance of objects--such as buttons--after they are clicked. You can do this by switching the cast member assigned to the sprite after the sprite is clicked and the mouse button is released. The sprite's different appearance indicates that the sprite has already been clicked.

Where you place an `on mouseup` handler can affect when it runs.

- When you want the handler to apply to a specific sprite, put the handler in a sprite script.
- When you want the handler to apply to a cast member in general, put the handler in a script of the cast member.
- When you want the handler to apply to an entire frame, put the handler in a frame script.
- When you want the handler to apply throughout the entire movie, put the handler in a movie script.

You can override an `on mouseup` handler by placing an alternate `on mouseup` handler in a location that Lingo checks before it gets to the handler you want to override. For example, you can override an `on mouseup` handler assigned to a cast member by placing an `on mouseup` handler in a sprite script.

Example

Related topics: on enterFrame, on exitFrame, on idle, on keyDown, on keyUp, on mouseDown, on startMovie, on stepMovie, and on stopMovie event handlers

Lingo example: on mouseUp

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

mouse button after clicking the sprite:

```
on mouseUp  puppetSprite 10, TRUE   set the castNum of sprite 10 to
"Dimmed"end
```

on startMovie (Lingo)

Syntax:
```
on startMovie
     statement(s)
end startMovie
```

This event handler contains statements that are activated after the movie preloads cast members but before the movie starts playing, regardless of where the playback head is.

An `on startMovie` handler is a good place to put Lingo that opens resource files, creates global variables, initializes variables, plays a sound while the rest of the movie is loading into memory, and checks and adjusts to computer conditions such as color depth.

Example

Related topics: on enterFrame, on exitFrame, on idle, on keyDown, on keyUp, on mouseDown, on mouseUp, on stepMovie, and on stopMovie event handlers

Lingo example: on startMovie

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler creates global variables and opens two resource files when the movie starts:

```
on startMovie
   global currentScore
   set currentScore = 0
end
```

on stepMovie (Lingo)

Syntax:
```
on stepMovie
     statement(s)
end stepMovie
```

This handler contains statements that are executed each time the playback head enters a new frame. This handler, which was used in earlier versions of Director, has the same result as the `on enterFrame` handler. The Allow Outdated Lingo checkbox must be selected to be able to use on stepMovie syntax in movies created in earlier versions of Director.

Related topics: on enterFrame, on exitFrame, on idle, on keyDown, on keyUp, on mouseDown, on mouseUp, on startMovie, and on stopMovie event handlers ; perFrameHook property

on stopMovie (Lingo)

Syntax:
```
on stopMovie
     statement(s)
end stopMovie
```

This event handler contains statements that are activated when the movie stops playing.

An `on stopMovie` handler is a good place to put Lingo that performs "cleanup" tasks--such as closing resource files, clearing global variables, erasing text fields, and disposing of objects--when the movie is finished.

Example

Related topics: on enterFrame, on exitFrame, on idle, on keyDown, on keyUp, on mouseDown, on mouseUp, on startMovie, and on stepMovie event handlers

Lingo example: on stopMovie

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler clears global variables and closes two resource files when the movie stops:

```
on stopMovie
   set gCurrentScore = 0
   closeResFile "Special Fonts"
   closeResFile "Special Cursors"
end
```

open (Lingo)

Syntax: `open [`*whichDocument* `with]` *whichApplication*

This command launches the application specified by the string *whichApplication*. By specifying *whichDocument*, you can specify a document that the application opens at the same time. When either is in a different folder than the current movie, you must specify the pathname.

If you are running MultiFinder (on a Macintosh), there must be enough memory to run both Macromedia Director and the other application at the same time.

Example

Related topics: openDA, openResFile, and openXlib commands

Lingo example: open

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement opens the MacWrite application:

```
open "MacWrite"
```

This statement opens the MacWrite application, which is in the folder Applications on the drive specifed by the variable myDrive, and the document named storyboards:

```
open storyboards with myDrive & "Applications:" ¬
   & MacWrite
```

open window (Lingo)

Syntax: `open window "`*whichWindow*`"`

This command opens a window that can play a Director movie and brings it to the front of the stage. The window is specified by *whichWindow* and must a have movie already assigned to it before you can use the `open window` command.

Example

Related topics: close window command

Lingo example: open window

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement opens the window Panel and brings it to the front:

```
open window "Panel"
```

openDA (Lingo)

Syntax: `openDA` *DAname*

On the Macintosh, this command opens the desk accessory specified by *DAName*, which is the menu item name or an expression that yields the menu item name of any desk accessory installed in the Macintosh. The `openDA` command has no effect under Windows.

Example

Related topics: closeDA command

Lingo example: openDA

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement opens the Calculator desk accessory:

```
openDA "Calculator"
```

openResFile (Lingo)

Syntax: `openResFile` *whichFile*

On the Macintosh, this command opens the resource file specified by the string *whichFile*. When the file is in a different folder than the current movie, *whichFile* must specify a pathname. The `openResFile` command has no effect under Windows.

In earlier versions of Director, this command was necessary to make additional fonts and cursors available in your movies. However, you can now provide custom cursors by importing the cursor as a cast member and using the cursor property.

When the file is already open, `openResFile` has no effect. It is good practice to close any open file as soon as you are finished using it.

Do not use `openResFile` to open another application. Its code resources interfere with Director's. Use a resource mover like ResEdit to move the resources you need to a separate resource file.

Related topics: closeResFile and showResFile commands; cursor property

Lingo example: openResFile

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement opens the resource file Special Fonts:

```
openResFile "Special Fonts"
```

This statement opens the resource file Special Icons, which is in another folder:

```
openResFile somePath &"Special Icons"
```

openXlib (Lingo)

Syntax: `openXlib` *whichFile*

This command opens the Xlibrary file specified by the string expression *whichFile*. If the file is in a different folder than the current movie, *whichFile* must include the pathname.

It is good practice to close any file you have opened as soon as you are finished using it. Do not use `openResFile` to open another application, because its code resources interfere with Director's. When the file is already open, `openXlib` has no effect.

The `openXlib` command also opens HyperCard XCMDs and XFCNs so that you can use them with Director. When you need to use an XCMD from more than one application in a movie, use this command to open a link to the HyperCard stack, rather than install the XCMD in both places with ResEdit. When you do that, a resource conflict that results in a system beep occurs.

Xlibrary files contain XObjects as XCOD resources. Unlike `openResFile`, `openXlib` makes these XObjects known to Director. Using the mNew predefined method, you can then create instances of the XObjects in memory.

Under Windows, the .DLL extension is optional.

Example

Related topics: closeXlib and showXlib commands

Lingo example: openXlib

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement opens the Xlibrary file VideoDisc Xlibrary:

```
openXlib "VideoDisc Xlibrary"
```

This statement opens the Xlibrary file XObjects, which is in a different folder than the current movie:

```
openXlib "My Drive:New Stuff:Transporter XObjects"
```

optionDown (Lingo)

Syntax: `the optionDown`

This function determines whether the Option key on the Maciatosh or the Alt key on the PC is being pressed.

- When the Option key or Alt key is being pressed, `the optionDown` is TRUE.
- When the Option key or Alt key is not being pressed, `the optionDown` is FALSE.

Example

Related topics: controlDown, commandDown, key, and shiftDown functions

Lingo example: optionDown

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler checks whether the Option key or Alt key is being pressed and calls the handler named doOptionKey if it is:

```
on keyDown

   if the optionDown then doOptionKey(the key)

end
```

or (Lingo)

Syntax: *logicalExpression1* `or` *logicalExpression2*

This operator performs a logical OR operation on two logical expressions.

- When either expression or both expressions are TRUE, the result is TRUE (1).
- When both expressions are FALSE, the result is FALSE (0).

This is a logical operator with a precedence level of 4.

Example

Related topics: and and not logical operators

Lingo example: or

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement has the message window display whether at least one of the expressions 1 < 2 and 1 > 2 is TRUE:

```
put 1 < 2 or 1 > 2
```

Because the first expression is TRUE, the result is 1, which is the numerical equivalent of TRUE.

This statement checks whether the contents of the text cast member named field are either AK or HI, and displays an alert if they are:

```
if field "State" = "AK" or field "State" = "HI" ¬
   then alert "You're off the map!"
```

palette of cast (Lingo)

Syntax: `the palette of cast` *whichCastMember*

This cast property determines which palette is associated with the cast member specified by *whichCastmember*. This property applies to bitmap cast members only.

- When the palette number is a positive number, it refers to another cast member.
- When the palette number is a negative number, it refers to a built-in palette.

The `palette of cast` property can be tested and set.

Example

Lingo example: palette of cast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement displays the palette assigned to the cast member Leaves in the message window:

```
put the palette of cast "Leaves"
```

param (Lingo)

Syntax: `param(`*parameter*`)`

This function gives the value of a parameter that is an argument to a handler. The variable *parameter* represents the parameter's position in the list.

Example

Related topic: paramCount function

Lingo example: param

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler calculates the average value of a list of parameters:

```
on avg first, second, third
   set n = paramCount()
   set sum = 0.0
      repeat with i = 1 to n
         set sum = param(i) + sum
      end repeat
   return sum/n
end
```

This statement passes the handler three values and displays the result in the message window:

```
put avg(1,2,3)
-- 2.0
```

paramCount (Lingo)

Syntax: `the paramCount`

This function determines the number of parameters sent to the current handler.

Example

Lingo example: paramCount

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler sets the variable named counter to the number of parameters that were sent to the current handler. In this case, the handler was sent the parameters (1, 2, 3):

```
set counter = paramCount()
```

pass (Lingo)

Syntax: `pass`

This command passes an event message to the next location in the message hierarchy. Otherwise, an event message stops at the first location that contains a handler for the event.

Passing an event message to other locations in the message hierarchy lets you execute more than one handler for a given event.

Example

Related topic: dontPassEvent command

Lingo example: pass

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

Used together, these handlers are both activated by a mouseUp event because the first handler contains a pass command.

This on mouse Up handler attached to sprite 3 executes the handler and then passes the mouseUp message on:

```
on mouseUp
   if sprite 3 intersects sprite 4 ¬
      then set the text of cast 10 = ¬
      "You placed it correctly"
   pass
end
```

This on mouseUp handler in the frame script executes because the on mouseUp handler assigned to the sprite script contains the pass command:

```
on mouseUp
   go to "Next test"
end
```

pasteClipBoardInto (Lingo)

Syntax: `pasteClipBoardInto cast` *whichCastMember*

This command pastes the contents of the Clipboard into the cast member specified by *whichCastMember*. When you paste into an occupied cast window slot, the old cast member is completely erased. For instance, pasting a bitmap into a text cast member makes the bitmap the cast member and erases the text cast member.

You can paste any item that is in a format that Director can use as a cast member. When you copy text from another application, the text's formatting is not retained.

The `pasteClipBoardInto cast` command provides a convenient way to copy objects from other movies and from other applications into the cast window.

Example

Lingo example: pasteClipBoardInto

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement pastes the contents of the Clipboard into the bitmap cast member Shrine:

```
pasteClipBoardInto cast Shrine
```

pathName (Lingo)

Syntax: `the pathName`

This function returns a string containing the full pathname of the folder in which the current movie is located.

Example

Related topics: movie function

Lingo example: pathName

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether the pathname contains the term System and has the computer beep if it does:

```
if the pathName contains "System" then beep
```

pause (Lingo)

Syntax: `pause`

This command causes the playback head to halt. Typically, you would put the `pause` command in the script channel of a frame, and then assign `continue` or `go` commands to one or more sprite scripts in that frame.

In many cases, using `pause` is recommended over looping on the same frame, or looping between two frames. This is because a pause uses much less processor time than repeatedly moving the playback head to the beginning of the frame. Some exceptions to this general rule are when you are moving sprites or are using the `perFrameHook`, so keeping the playback head going to the same frame is required.

The `pause` command is useful for halting the movie while a menu is displayed or for letting the user look at a screen as long as she or he wants.

Example

Related topics: continue command; pauseState function

Lingo example: pause

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

The following on mouseUp handler for a button alternately pauses and continues the animation, like the pause button on a videocassette recorder:

```
on mouseUp
   if the pauseState = TRUE then
      continue
   else
      pause
   end if
end
```

pausedAtStart (Lingo)

Syntax: `the pausedAtStart of cast` *whichDVMovie*

This digital video cast property specifies whether the Paused at Start checkbox in the Digital Video Cast Member Info dialog box is checked or not.

- When the `pausedAtStart of cast` property is TRUE, the Paused at Start checkbox is checked.
- When the `pausedAtStart of cast` property is FALSE, the Paused at Start checkbox is not checked.

The `pausedAtStart of cast` property can be tested and set.

Example

Lingo example: pausedAtStart

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement turns on the Paused at Start checkbox in the Digital Video Cast Member Info dialog box for the digital video movie Rotating Chair:

```
set the pausedAtStart of cast "Rotating Chair" = TRUE
```

pauseState (Lingo)

Syntax: `the pauseState`

This function returns TRUE when the movie is currently paused.

Example

Related topics: pause and continue commands

Lingo example: pauseState

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether the movie is currently paused and has the movie continue if it is:

```
if the pauseState = TRUE then continue
```

perFrameHook (Lingo)

Syntax: `the perFrameHook`

The perFrameHook property designates an object (created by either a factory or an XObject) that is called every frame (or subframe) with a special message called `mAtFrame`. You specify what routines and procedures are used in `mAtFrame`.

The `perFrameHook` property was required in earlier versions of Director. However, you can now achieve the same results by placing Lingo that you want to execute at every frame in an `on enterFrame` handler.

The `perFrameHook` can be used to call a certain set of procedures (using `mAtFrame`) each frame. Without the `perFrameHook`, you would have to type this set of procedures (using a handler) into the script channel of every single frame in which you wanted it to occur. With the perFrameHook, you need only set the proper object to the perFrameHook once and the procedures (contained in the `mAtframe` method) will be executed at every frame. When you no longer want to use the `perFrameHook`, set it to 0 to turn it off. The `perFrameHook` is especially useful when recording animations frame-per-frame to videotape.

At every frame, the `perFrameHook` object is sent the `mAtFrame` message. Therefore, you must create a factory that defines an `mAtFrame` method (in the same factory that creates the object you set to the `perFrameHook`).

When recording frame-per-frame to videotape, you can define two arguments for `mAtFrame` that specify the frame and subframe (subframes occur during transitions; each change during the transition is a subframe):

```
method mAtFrame frame, subframe
```

The frame argument is sent for each frame and the subframe argument is sent for each subframe. You can name the arguments whatever you like, if you prefer not to use frame or subframe. You can also define additional arguments for `mAtFrame`, whether you are recording frame-per-frame to videotape or not.

The `perFrameHook` is primarily designed to be used with XObjects that have an `mAtFrame` argument. If you do use the `perFrameHook` with a factory, do not use the `updateStage` command or set the text property of a sprite. Otherwise, unexpected results could occur.

Example

Related topics: factory and method keywords

Lingo example: perFrameHook

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This factory lets you create objects that display the current frame and subframe numbers (when a transition is occurring) in the message window:

```
factory myPerFrameHook

method mAtFrame n,sub

put "at frame " & n & ", subframe " & sub

end mAtFrame
```

pi (Lingo)

Syntax: `pi()`

This function gives the value of ¹, the ratio of a circle's circumference to its diameter. The value of ¹ is given as a floating-point number to the number of decimal places set by the `floatPrecision` property.

Example

Lingo example: pi

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement uses the pi function as part of an equation for calculating the area of a circle:

```
set vArea = pi()*power(vRadius,2)
```

picture of cast (Lingo)

Syntax: `the picture of cast` *whichCastmember*

This cast property determines the image displayed by a bitmap or PICT cast member.

The `picture of cast` property can be tested and set.

Example

Related topics: type of sprite property

Lingo example: picture of cast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the variable named pict to the image in the cast member named Sunset:

```
put the picture of cast "Sunset" into pict
```

pictureP (Lingo)

Syntax: `pictureP(`*castMember*`)`

This function determines whether the cast member specified by *castMember* is a bitmap data type.

- When the cast member is a bitmap data type, `pictureP` is TRUE (1).
- When the cast member is not a bitmap data type, `pictureP` is FALSE (0).

Example

Lingo example: pictureP

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement has the message window display whether the cast member Shrine is a bitmap:

```
put pictureP("Shrine")
```

The result is 1, which is the numerical equivalent of TRUE.

play (Lingo)

Syntax: `play [frame]` *whichFrame*
`play movie` *whichMovie*
`play frame` *whichFrame* `of movie` *whichMovie*

This command causes the playback head to jump to the specified frame of the specified movie. The expression *whichFrame* can be either a string marker label or an integer frame number. The expression *whichMovie* must be a string that specifies a movie file. When the movie is in another folder, *whichMovie* must specify a pathname.

The `play` command is similar to the `go to` command, but with the `play` command, when the sequence being played is over, the playback head automatically returns to the frame where the `play` command was called. If the `play` command is issued from a frame script, the playback head returns to the next frame; if the `play` command comes from a sprite script or handler, the playback head returns to the same frame. A sequence is over when the playback head reaches the end of the movie, or the `play done` command is given.

The `play` command can also be used for playing several movies from a single handler. The handler is suspended while each movie plays, but resumes when the movie is over. Contrast this with a series of `go` commands that, when called from a handler, play the first frame of each movie. The handler is not suspended while the movie plays but immediately continues executing.

Example

Related topics: go and play done commands; marker function

Lingo example: play

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement moves the playback head to the marker named blink:

```
play "blink"
```

This statement moves the playback head to the next marker:

```
play marker(1)
```

This statement moves the playback head to a separate movie:

```
play movie "My Drive:More Movies:" & newMovie
```

play done (Lingo)

Syntax: `play done`

This command indicates that the sequence being played is complete when the current movie or sequence was started using the `play` or `go to` commands. The play done command causes the playback head to return to where the sequence was started from, either a frame in the same movie or a frame in the another separate. When the `play` command is issued from a frame script, the playback head returns to the next frame; if the `play` command is issued from a sprite script, the playback head returns to the same frame.

Related topic: play command

point (Lingo)

Syntax: `point(`*horizontal*`,` *vertical*`)`

This function yields a point that has the horizontal coordinate specified by *horizontal* and the vertical coordinate specified by *vertical*.

Example

Related topic: rect function

Lingo example: point

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the variable lastLocation to the point (250, 400):

```
put point(250, 400) into lastLocation
```

power (Lingo)

Syntax: `power(`*base*`, `*exponent*`)`

This function calculates the value of the number specified by base to the exponent specified by exponent.

Example

Lingo example: power

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the variable vResult to the value of 4 to the third power:

```
set vResult to power(4,3)
```

preLoad (Lingo)

Syntax: `preLoad`
`preLoad toFrameNum`
`preLoad fromFrame, toFrameNum`

This command preloads cast members in the specified frame or range of frames into memory. Preloading stops when memory is full or when all of the specified cast members have been preloaded.

When used without arguments, the `preLoad` command causes a preload of all cast members used from the current frame to the last frame of a movie.

When used with one argument, *toFrame*, the preLoad command causes a preload of all cast members used in the range of frames from the current frame to the frame *toFrame*, as specified by frame number or label name.

When used with two arguments, *fromFrame* and *toFrame*, the `preLoad` command causes a preload of all cast members used in the range of frames from the frame *fromFrame* to the frame *toFrame*, as specified by frame number or label name.

The `preLoad` command also returns the number of the last frame successfully loaded. To access this value, use the `result` function.

Related topics: preLoadCast command

Lingo example: preLoad

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement preloads the cast members used up from the current frame to the frame that has the next marker:

```
preLoad marker (1)
```

This statement preloads the cast members used up from frame to the frame 10 to frame 50:

```
preLoad 10, 50
```

preLoad of cast (Lingo)

Syntax: `the preLoad of cast` *castMember*

This digital video cast property determines whether the digital video cast member specified by *castMember* has been preloaded into memory.

- When the digital video cast member has been preloaded into memory, `the preLoad of cast` is TRUE.

- When the digital video cast member has not been preloaded into memory, `the preLoad of cast` is TRUE.

Example

Lingo example: preLoad of cast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement has the message window display whether the digital video movie Rotating Chair has been preloaded into memory:

```
put the preLoad of cast "Rotating Chair"
```

preLoadCast (Lingo)

Syntax: `preLoadCast`
`preLoadCast` *`CastNumber`*
`preLoadCast` *`fromCastNumber`*`, ` *`toCastNumber`*

This command preloads cast members. Preloading stops when memory is full or when all of the specified cast members have been preloaded.

When used without arguments, the `preLoadCast` command preloads all cast members in the movie.

When used with the *castNumber* argument, the `preLoadCast` command preloads that cast member.

When used with the arguments *fromCastNumber* and *toCastNumber*, the `preLoadCast` command preloads all cast members in the range specified by the cast member numbers or names.

The `preLoadCast` command returns the cast member number of the last cast member successfully loaded. To obtain this value, use the `result` function.

Example

Lingo example: preLoadCast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement preloads cast member 20:

```
preLoadCast 20
```

This statement preloads cast member Shrine and the ten cast members after it in the cast window:

```
set logo to the number of cast "Shrine"preLoadCast logo, logo + 10
```

preLoadEventAbort (Lingo)

Syntax: `the preLoadEventAbort`

This property specifies whether pressing keys or clicking the mouse can stop preloading of cast members.

- When the `preLoadEventAbort` property is TRUE, pressing keys or clicking the mouse can stop preloading of cast members.

- When the `preLoadEventAbort` property is FALSE, pressing keys or clicking the mouse cannot stop preloading of cast members.

The default value is FALSE. The setting of this property affects the current movie.

The `preLoadEventAbort` property can be tested and set.

Example

Related topics: preLoad and preLoadCast commands

Lingo example: preLoadEventAbort

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement lets the user stop preloading of cast members by pressing keys or clicking the mouse:

```
set the preLoadEventAbort = TRUE
```

preLoadRAM (Lingo)

Syntax: `the preLoadRAM`

This property specifies the amount of RAM that can be used for preloading a digital video movie. It can be set and tested.

This is useful for managing memory, so that preloaded digital video cast members are not given more than a certain limit of memory, and other types of cast members can still be preloaded. When the value is set to FALSE, all available memory can be used for preloading digital video cast members.

Example

Related topics: loop and next keywords

Lingo example: preLoadRAM

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement allocates the amount of RAM available for preloading three times the size of the cast member Interview:

```
set the preLoadRAM to 3 * (the size of cast ¬   "Interview")
```

previous (Lingo)

See go previous

printFrom (Lingo)

Syntax: `printFrom` *fromFrame* `[,` *toFrame*`]` `[,` *reduction*`]`

This command prints whatever is displayed on the stage in each frame starting at the frame specified by *fromFrame*. Optionally you can supply the *toFrame*, and the reduction (100, 50, or 25 percent).

When printing at less than 100 percent, the document prints as a bitmap, so text does not print as sharply as it would at full size.

Example

Lingo example: printFrom

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement prints what is on the stage in every frame starting at frame 1:

```
printFrom 1
```

This statement prints what is on the stage in every frame from frame 10 to frame 20. The reduction is 50 percent:

```
printFrom 10, 20, 50
```

property (Lingo)

Syntax: `property [property1][, property2][, property3] [...]`

This keyword declares that the properties specified by *property1*, *property2*, and so on are property variables. Property variables, which are used in parent scripts, serve the same purpose as instance variables in factories and XObjects.

You declare property variables at the beginning of the parent script. You can access them from outside the parent script by using the `the` operator.

Example

Related topics: ancestor property

Lingo example: property

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement allows each child object created from a single parent script to have its own location and velocity setting:

```
property location, velocity
```

puppet of sprite (Lingo)

Syntax: `the puppet of sprite` *whichSprite*

This sprite property determines whether the sprite specified by the integer expression *whichSprite* is a puppet.

A puppet sprite is controlled by Lingo instead of the score. For example, Lingo can switch the cast member assigned to a sprite or turn on and off whether the sprite is moveable. For more information on using puppets, see "Using Puppets" in Chapter 4 in the *Using Lingo* manual.

The sprite channel must contain a sprite before you can make the channel a puppet.

Making the sprite channel a puppet lets you control any of the sprite properties--such as `castNum of sprite`, `locH of sprite`, and `width of sprite`--from Lingo.

Setting the puppet of sprite property is equivalent to using the `puppetSprite` command. For example, the statement:

```
set the puppet of sprite 1 to TRUE
```

has the same effect as:

```
puppetSprite 1, TRUE
```

The `puppetSprite` property can be tested and set. The default value is FALSE.

Example

Related topics: puppetSprite command

Lingo example: puppet of sprite

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement makes the sprite numbered i + 1 a puppet:

```
set the puppet of sprite (i + 1) to TRUE
```

This statement records whether sprite 5 is a puppet by assigning the value of the puppet of sprite to the variable. When sprite 5 is a puppet, isPuppet is set to TRUE. When sprite 5 is not a puppet, isPuppet is set to FALSE:

```
put the puppet of sprite 5 into isPuppet
```

puppetPalette (Lingo)

Syntax: `puppetPalette` *whichPalette* `[,` *speed*`] [,` *nFrames*`]`

This command causes the palette channel to act as a puppet. When the palette channel is a puppet, Lingo can override the palette setting in the palette channel of the score and assign palettes to the movie.

The `puppetPalette` command sets the current palette to the palette cast member specified by the expression *whichPalette*. If *whichPalette* evaluates to a string, it specifies the cast name of the palette. If *whichPalette* evaluates to an integer, it specifies the cast number of the palette.

Optionally, you can fade in the palette by replacing *speed* with an integer expression, with 1 being slowest and 60 being fastest. You can also fade in the palette over several frames by replacing *nFrames* with an integer expression for the number of frames.

A puppet palette remains in effect until you turn it off with the command `puppetPalette 0`. No subsequent palette changes in the score are obeyed when the puppet palette is in effect.

Example

Lingo example: puppetPalette

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement makes Rainbow the movie's palette:

```
puppetPalette "Rainbow"
```

This statement makes Grayscale the movie's palette. The transition to the Grayscale palette occurs over a time setting of 15 and between frames labeled Gray and Color:

```
puppetPalette customPalette, 15, ¬
   label("Gray") - label("Color")
```

puppetSound (Lingo)

Syntax: `puppetSound` *whichCastMember*
`puppetSound 0`

This command makes the sound channel a puppet and plays the sound cast member specified by *whichCastMember*. When the sound is a puppet, Lingo can override any sounds assigned in the first sound channel of the score.

The `puppetSound` command starts playing the specified sound. To play a sound stored in the cast, replace *whichCastMember* with the name of the sound cast member. The sound will not start playing until the playback head moves or until the `updateStage` command is executed.

The statement `puppetSound 0` stops a sound from playing. It also turns off the puppet status of the sound and returns control of the sound to the sound channel in the score. Use `puppetSound` to restore control of the sound channel to the score.

Puppet sounds can be useful for playing a sound while a different movie is being loaded into memory.

Example

Related topics: sound fadeIn, sound fadeOut, sound playFile, sound stop commands

Lingo example: puppetSound

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement plays the sound Wind under control of Lingo:

```
puppetSound "Wind"
```

puppetSprite (Lingo)

Syntax: `puppetSprite` *whichSprite, state*

This command controls whether the sprite specified by *whichSprite* is a puppet. When a sprite is a puppet, any sprite property can be controlled by Lingo instead of the score. For example, Lingo can switch the cast member assigned to a sprite or turn on and off whether the sprite is moveable.

- When *state* is TRUE, Lingo controls the sprite and the score is ignored.

- When *state* is FALSE, the sprite is controlled by the score.

The initial properties of the puppet are taken from whatever sprite is in the channel when the `puppetSprite` command is executed. Subsequent control of the sprite properties through Lingo can change these properties.

The channel must contain a sprite when you use the `puppetSprite` command.

You must provide the command `puppetSprite` *whichSprite*, `FALSE` when you are finished with your puppet; otherwise unpredictable results can occur when the playback head returns to sprites in frames that aren't intended to be puppets.

For more information on using puppets, see "Using Puppets" in Chapter 4 of *Using Lingo*.

Example

Note: You must provide the command `puppetSprite` *whichSprite*, `false` when you are finished with your puppet, otherwise unpredictable results can occur when the playback head returns to sprites in frames not intended to be puppets.

Related topics: backColor of sprite, bottom of sprite, castNum of sprite, constraint of sprite, cursor, foreColor of sprite, height of sprite, ink of sprite, left of sprite, lineSize of sprite, locH of sprite, locV of sprite, puppet of sprite, right of sprite, stretch of sprite, top of sprite, type of sprite, and width of sprite sprite properties; puppetSprite property

Lingo example: puppetSprite

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement makes the sprite in channel 15 a puppet:

```
puppetSprite 15, TRUE
```

This statement removes the puppet condition from the sprite in the channel numbered i + 1:

```
puppetSprite i + 1, FALSE
```

puppetTempo (Lingo)

Syntax: `puppetTempo` *framesPerSecond*

This command causes the tempo channel to act as a puppet. When the tempo channel is a puppet, Lingo can override the tempo setting in the score and change the tempo assigned to the movie.

The `puppetTempo` command sets the tempo to the number of frames specified by *framesPerSecond*. The maximum frames per second is 60.

You do not need to turn off the puppet tempo condition to have subsequent tempo changes in the score take effect.

Example

Lingo example: puppetTempo

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement set the movie's tempo to 30 frames per second:

```
puppetTempo 30
```

This statement increases the movie's old tempo by ten frames per second:

```
puppetTempo oldTempo + 10
```

puppetTransition (Lingo)

Syntax: `puppetTransition` *whichTransition* `[,` *time*`]` `[,` *chunkSize* ¬ `[,` *changeArea*`]`

This command performs the transition specified by *whichTransition* between the current frame and the next frame.

Replace *whichTransition* with one of the following values:

Code	Transition
01	Wipe right
02	Wipe left
03	Wipe down
04	Wipe up
05	Center out, horizontal
06	Edges in, horizontal
07	Center out, vertical
08	Edges in, vertical
09	Center out, square
10	Edges in, square
11	Push left
12	Push right
13	Push down
14	Push up
15	Reveal up
16	Reveal up, right
17	Reveal right
18	Reveal down, right
19	Reveal down
20	Reveal down, left
21	Reveal left
22	Reveal up, left
23	Dissolve, pixels fast *
24	Dissolve, boxy rectangles
25	Dissolve, boxy squares
26	Dissolve, patterns
27	Random rows
28	Random columns
29	Cover down
30	Cover down, left
31	Cover down, right
32	Cover left
33	Cover right
34	Cover up
35	Cover up, left
36	Cover up, right
37	Venetian blinds
38	Checkerboard
39	Strips on bottom, build left
40	Strips on bottom, build right
41	Strips on left, build down
42	Strips on left, build up

43 Strips on right, build down
44 Strips on right, build up
45 Strips on top, build left
46 Strips on top, build right
47 Zoom open
48 Zoom close
49 Vertical blinds
50 Dissolve, bits fast *
51 Dissolve, pixels *
52 Dissolve, bits *

Transitions marked with an asterisk (*) in the table will not work on monitors that are set to 32 bits. Instead, the stage changes instantly from one screen to the next.

Replace *time* with the number of 1/4 seconds used to complete the transition. The minimum is 0; the maximum is 120 (30 seconds). Replace *chunkSize* with the number of pixels in each chunk of the transition. The minimum is 1; the maximum is 128. Smaller chunk sizes give smoother transitions but are slower.

There is not a direct relationship between a low time and a fast transition. The actual speed of the transition depends on the relation of *chunkSize* and *time*. As an example, if the *chunkSize* is one pixel, the transition takes a long time no matter how low the time, because the Macintosh has to do a lot of work. To make transitions occur faster you should use a larger chunk size, instead of setting a shorter time.

Replace *changeArea* with a value that determines whether the transition occurs only in the changing area. The `changeArea` is an area within which sprites have changed.

- To have the transition occur only in the areas that change, replace *changeArea* with TRUE, which is the default setting.

- To have the transition occur over the entire stage, replace *changeArea* with FALSE.

Example

Lingo example: puppetTransition

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement performs a wipe from right transition. Because no value is specified for changeArea, the transition occurs only on the changing area, which is the default:

```
puppetTransition 1
```

This statement performs a wipe from right transition that lasts 1 second, has a chunk size of 20, and occurs over the entire stage:

```
puppetTransition 2, 4, 20, FALSE
```

purgePriority (Lingo)

Syntax: `the purgePriority of cast` *whichCastMember*

This cast property specifies the purge priority of the cast member specified by *whichCastMember*.

Cast members' purge priorities determine the priority that Director follows when choosing which cast members to delete from memory when memory is full. The higher the purge priority, the more likely that the cast member is deleted. The following `purgePriority` settings are available:

0--Never purge
1--Purge last
2--Purge next
3--Purge normal

Setting `purgePriority` for cast members is useful for managing memory when the size of the movie's cast exceeds the available memory. As a general rule, you can minimize pauses while the movie loads cast members by assigning a low purge priority to cast members that are frequently used in the course of the movie. This reduces the number of times that Director reloads the cast member when the movie plays.

The Normal setting allows Director to purge cast members from memory at random. The Next, Last, and Never settings allow you some control over purging.

Note: If you set too many cast members to Last(1), your movie might run out of memory and perform poorly. If you set too many cast memebers to Never (0), Director might quit without warning.

Example

Lingo example: purgePriority

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the purge priority of cast member Background to 2, which makes it one of the next cast members to be purged when memory is needed:

```
set the purgePriority of cast "Background" to 2
```

put (Lingo)

Syntax: `put` *expression*

This command evaluates the expression specified by *expression* and displays the result in the message window. This can be used as a debugging tool by tracking the values of variables as the movie plays.

Example

Related topics: put...after, put...before, and put...into commands

Lingo example: put

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement displays the time in the message window:

```
put the time
-- "9:10 AM"
```

This statement displays the value assigned to the variable vBid in the message window:

```
put vBid
-- "Johnson"
```

put...after (Lingo)

Syntax: `put` *expression* `after` *chunkExpression*

This command evaluates a Lingo expression, converts the value to a string, and inserts the resulting string after a specified chunk in a text container. (If *chunkExpression* specifies a nonexistent target chunk, the string value is inserted as appropriate into the text container.) The previous contents of the container remain.

Chunk expressions can refer to any character, word, item, or line in any container of text. Containers include fields (text cast members) and variables that hold strings, and specified characters, words, items, lines, and ranges in containers.

Example

Related topics: char...of, item...of, line...of, put...before, put...into, and word...of chunk expression keywords

Lingo example: put...after

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement adds the string "fox dog cat" after the contents of the variable animalList:

```
put "fox dog cat" after animalList
```

put...before (Lingo)

Syntax: `put` *expression* `before` *chunkExpression*

This command evaluates a Lingo expression, converts the value to a string, and inserts the resulting string before a specified chunk in a text container. (If *chunkExpression* specifies a nonexistent target chunk, the string value is inserted as appropriate into the text container.) The previous contents of the container remain.

Chunk expressions can refer to any character, word, item, or line in any container of text. Containers include fields (text cast members) and variables that hold strings, and specified characters, words, items, lines, and ranges in containers.

Example

Related topics: char...of, item...of, line...of, put...after, put...into, and word...of chunk expression keywords

Lingo example: put...before

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

These statements set the variable named animalList to the string "fox dog cat" and then insert the word elk before the second word of the list:

```
put "fox dog cat" into animalListput "elk " before word 2 of animalList
```

The result is the string "fox elk dog cat".

Note: In the second statement, there is an intentional space between the word elk and the second quote mark. If it were missing, the resulting string would be "fox elkdog cat".

These statements set the field named Price to the value of 20.00 plus 7.25, and then insert a dollar sign before the number:

```
put (20.00 + 7.25) into field "Price"put "$" before field "Price"
```

The result in field Price is "$27.25".

put...into (Lingo)

Syntax: `put` *expression* `into` *variable*
`put` *expression* `into` *chunkExpression*

This command has two different usages.

The first usage evaluates a Lingo expression and stores its value in a local, global, property or instance variable. The value can be an integer, a floating-point number, a string, an object, or a symbol; it resides unchanged in the variable.

The second usage evaluates a Lingo expression, converts the value to a string, and uses the resulting string to replace a specified chunk in a text container. (If *chunkExpression* specifies a nonexistent target chunk, the string value is inserted as appropriate into the text container.)

Chunk expressions can refer to any character, word, item, or line in any container of text. Containers include fields (text cast members) and variables that hold strings, and specified characters, words, items, lines, and ranges in containers.

Note: In Lingo, you can use set...to and set...= as well as put...into for variable assignments. However, since HyperTalk only allows set to be used with properties, its use with variables is not recommended.

Example

Related topics: char...of, item...of, line...of, put...after, put...before, and word...of chunk expression keywords; set to command

Lingo example: put...into

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the variable x to the square root of 2:

```
put sqrt(2.0) into x

The result is 1.4142.
```

quickTimePresent (Lingo)

Syntax: `the quickTimePresent`

This function determines whether QuickTime is available on the current computer.

- When QuickTime is present, the `quickTimePresent` function is TRUE (1).
- When QuickTime is not present, the `quickTimePresent` function is FALSE (0).

Example

Lingo example: quickTimePresent

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement determines whether QuickTime is available and plays the QuickTime movie Rotating Chair if it is:

```
if the quickTimePresent = 1 then ¬
   play "Rotating Chair"
```

quit (Lingo)

Syntax: `quit`

This command exits from Director or a projector.

- On the Macintosh, the quit command exits from Director or a projector to the Finder.
- Under Windows, the quit command exits from Director or a projector to the Windows Program Manager.

Example

Related topics: restart and shutDown commands

Lingo example: quit

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement has the computer exit to the Program Manager when the user presses Control-q:

```
if the key = "q" and the controlDown then quit
```

QUOTE (Lingo)

Syntax: `QUOTE`

This character constant represents the quote character. It is needed to refer to the literal quote character in a string, since the quote character itself is used by Lingo scripts to delimit strings.

Example

Lingo example: QUOTE

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement inserts quote characters in the string of text:

```
put "Can you spell" && QUOTE & "Macromedia" ¬
   & QUOTE & "?"
```

The result is quotes around the word Macromedia, as in the following string:

```
Can you spell "Macromedia"?
```

ramNeeded (Lingo)

Syntax: `ramNeeded (`*firstFrame, lastFrame*`)`

This function determines, in bytes, the memory needed to display a range of frames. For example, you can test the size of frames containing 32-bit artwork. If the `ramNeeded` is larger than the `freeBytes`, then go to frames containing 8-bit artwork. Divide by 1024 to convert bytes to kilobytes (K).

Example

Related topics: freeBytes function; size of cast cast property

Lingo example: ramNeeded

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the variable frameSize to the number of kilobytes needed to display frames 100 to 125 of the movie:

```
put ramNeeded (100, 125) into frameSize
```

This statement determines whether the memory needed to display frames 100 to 125 is more than the available memory and branches to a movie using cast members that have lower color depth if it is:

```
if ramNeeded (100, 125) > freeBytes then ¬
   play frame "8-bit"
```

random (Lingo)

Syntax: `random(`*integerExpression*`)`

This function returns a random integer from 1 to the value specified by *integerExpression*.

The `random` function is useful when you want to randomly vary values in a movie. Some possible uses are varying the path through a game, assigning random numbers, or changing the color or position of sprites.

Example

Lingo example: random

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement assigns random values to the variable diceRoll:

```
put random(6) + random(6) into diceRoll
```

This statement randomly changes the foreground color of sprite 10:

```
set the foreColor of sprite 10 = random(255)
```

This handler randomly chooses which of two movie segments to play in the "Noh Tale":

```
on selectMovie
   if random(2) = 2 then play frame "11a"
   else
      play frame "11-b" of movie "NTBRANCH.DIR"
   end if
end
```

randomSeed (Lingo)

Syntax: `the randomSeed`

This property specifies seed for generating random numbers. Using the same seed produces the same sequence of random numbers

The `randomSeed` property can be tested and set.

Example

Lingo example: randomSeed

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement displays the random seed number in the message window:

```
put the randomSeed
```

rect (Lingo)

Syntax: `rect(`*left*`, `*top*`, `*right*`, `*bottom*`)`
`rect(`*point1*`, `*point2*`)`

This function has two uses:

- When you use four arguments, the `rect` function defines a rectangle that has the sides specified by *left*, *top*, *right*, and *bottom*. The *left* and *right* values specify numbers of pixels from the left edge of the stage. The *top* and *bottom* values specify numbers of pixels from the top of the stage.

- When you use two arguments, the rect function defines a rectangle that encloses the points specified by *point1* and *point2*.

Example

Related topics: point function

Lingo example: rect

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the variable newArea to a rectangle whose left side is at 100, top is at 150, right side is at 300, and bottom is at 400 pixels:

```
put rect(100, 150, 300, 400) into newArea
```

This statement sets the variable newArea to the rectangle defined by the points firstPoint and secondPoint. The coordinates of firstPoint are (100, 150); the coordinates of secondPoint are (300, 400). Note that this statement creates the same rect as the rectangle created in the previous example:

```
put rect(firstPoint, secondPoint)
```

rect of cast (Lingo)

Syntax: `the rect of cast` *whichCastMember*

This cast property indicates the left, top, right, and bottom paint window coordinates of the rectangle of a cast member. The coordinates are returned as a rect.

The `rect of cast` property can be tested but not set.

Example

Lingo example: rect of cast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement displays the coordinates of bitmap cast member 20:

```
put the rect of cast 20
```

rect of window (Lingo)

Syntax: `the rect of window` *whichWindow*

This window property determines the left, top, right, and bottom coordinates of the window specified by *whichWindow*. The coordinates are given as a rect.

The `rect of window` property can be tested and set.

Example

Lingo example: rect of window

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement displays the coordinates of the window Control Panel:

```
put the rect of window "Control Panel"
```

regPoint (Lingo)

Syntax: `the regPoint of cast` *whichCastMember*

This cast property specifies the paint window registration point of a bitmap cast member. The registration points are listed as horizontal and vertical coordinates in a point that has the form `point` (`horizontal`, `vertical`).

The `regPoint of cast` property can be tested and set.

Example

Related topic: rect of cast property

Lingo example: regPoint

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement displays the registration points of the bitmap cast member Desk in the message window:

```
put the regPoint of cast "Desk"
```

This statement changes the registration points of the bitmap cast member Desk to the values in the list:

```
set the regPoint of cast "Desk" = ¬
   point (300, 400)
```

repeat while (Lingo)

Syntax: `repeat while` *testCondition*
[statements...]
`end repeat`

This keyword structure repeatedly executes the statements as long as the condition specified by *testCondition* is TRUE. Some possible uses for this structure are for Lingo that continues to read text strings until the end of a file is reached, checks items until the end of a list is reached, or repeatedly performs an action until the user clicks or releases the mouse button.

Example

Related topics: exit, exit repeat, and repeat with keywords

Lingo example: repeat while

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler starts the timer counting, resets the timer to 0, and then has the timer count up to 60 ticks:

```
on countTime
   startTimer
   repeat while the timer < 60
      -- waiting for timer
   end repeat
end countTime
```

repeat with (Lingo)

Syntax: `repeat with` *counter* `=` *start* `to` *finish*
[statements...]
`end repeat`

This keyword structure executes the Lingo specified by *statements* the number of times specified by *counter*. The value of *counter* is the difference between the value specified by *start* and the value specified by *finish*. The counter is incremented by 1 each time Lingo goes through the repeat loop.

The `repeat with` structure is useful for repeatedly applying the same effect to a series of puppets or calculating a series of numbers, such as a number to some exponent.

Example

Related topics: exit, exit repeat, and repeat while keywords

Lingo example: repeat with

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

The following handler turns sprites 1 through 30 into puppets:

```
on puppetize
   repeat with channel = 1 to 30
      puppetSprite channel, TRUE
   end repeat
end
```

repeat with...down to (Lingo)

Syntax: `repeat with` *variable* `=` *startValue* `down to` *endValue*

This keyword counts down by increments of 1 from *startValue* to *endValue*.

Example

Lingo example: repeat with...down to

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler contains a repeat loop that counts down from 20 to 15:

```
on countDown
   repeat with i = 20 down to 15
   set the castNum of sprite 6 to (10 + i)
   updateStage
end repeat
```

repeat with i in list (Lingo)

Syntax: `repeat with` *variable* `in` *someList*

This keyword assigns successive values from the specified list to the variable.

Example

Lingo example: repeat with i in list

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement displays four values in the message window:

```
repeat with x in [1, 2, 3, 4]
   put x
end repeat
```

restart (Lingo)

Syntax: `restart`

This command restarts the computer or exits Windows.

- On the Macintosh, the `restart` command is equivalent to choosing Restart in the Finder's Special menu.
- Under Windows, the `restart` command is equivalent to choosing Exit Windows from the Program Manager's File menu.

Example

Related topics: quit and shutDown commands

Lingo example: restart

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement restarts Windows when the user presses Control-r:

```
if the key = "r" and the commandDown then restart
```

result (Lingo)

Syntax: `the result`

This function gives the value of the return expression in the last handler executed.

The `result` function is useful for obtaining values from movies that are playing in windows and tracking Lingo's progress by displaying results of handlers in the message window as the movie plays.

Example

Related topic: return keyword

Lingo example: result

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

The following handler returns a random roll for two dice:

```
on diceRoll

   return random(6) + random(6)

end diceRoll
```

The two statements

```
diceRollput the result into roll
```

are equivalent to

```
put diceRoll() into roll
```

Note that

```
put diceRoll into roll
```

does not call the handler because there are no parentheses following diceRoll; diceRoll here is considered a variable reference.

RETURN constant (Lingo)

Syntax: `RETURN`

This character constant represents the return key.

Example

Lingo example: RETURN constant

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement has a paused movie continue when the user presses the Return key:

```
if the key = RETURN then continue
```

This statement uses the Return character constant to insert a return between two lines in an alert:

```
alert "Last line in the file." & RETURN & ¬
   "Click OK to exit."
```

Under Windows, writing to a text file requires an additional line feed character at the end of each line. This statement creates a two-character string named CRLF that provides the additional line feed:

```
put Return&numToChar(10) into CRLF
```

return (Lingo)

Syntax: `return` *expression*

This keyword is used in handlers and methods that return values. It returns the value of *expression* and exits from a handler or method. The expression can be an integer, floating-point number, string, object, or symbol.

When calling a handler or method that serves as a user-defined function and has a return value, you must use parentheses around the argument list. This is necessary even when there are no arguments, as in the `diceRoll` function handler discussed under the entry "`the result`".

Example

Related topics: result keyword

Lingo example: return

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

The following handler returns the greater of two expressions:

```
on max a, b
   if a > b then
      return a
   else
      return b
   end if
end max
```

If 3 and 7 were used for a and b, the result would be as follows:

```
put max(3, 7)-- 7
```

right of sprite (Lingo)

Syntax: `the right of sprite` *whichSprite*

This sprite property indicates the number of pixels that the right edge of the sprite specified by *whichSprite* is from the left edge of the stage.

The `right of sprite` property can be tested, but not set directly. The right horizontal coordinate of a sprite can be set using the `spriteBox` command.

Example

Note: Sprite coordinates are expressed relative to the upper-left corner of the Stage.

Related topics: bottom of sprite, height of sprite, left of sprite, locH of sprite, locV of sprite, top of sprite, and width of sprite sprite properties; spriteBox command

Lingo example: right of sprite

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement calls the handler offRightEdge when the right edge of sprite 3 is past the right edge of the stage:

```
if the right of sprite 3 > (the stageRight ¬
   - the stageLeft) then offRightEdge
```

rollOver (Lingo)

Syntax: `rollOver(`*whichSprite*`)`

This function indicates whether the cursor is currently over the bounding rectangle of the sprite specified by *whichSprite*.

- When `rollOver` is TRUE (1), the cursor is currently over the sprite.
- When `rollOver` is FALSE (0), the cursor is not currently over the sprite.

The `rollOver` function is typically used in frame scripts. It is useful for creating handlers that perform an action when the user places the cursor over a specific sprite or simulating additional sprite channels by splitting the stage into regions that send the playback head to a different frame that subdivides the region for the available sprite channels.

Related topics: mouseCast function

Lingo example: rollOver

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement changes the content of text cast member Message to "This is the place." when the cursor is over sprite 6:

```
if rollOver(6) then ¬

put "This is the place." into field "Message"
```

This handler sends the playback head to different frames when the cursor is over certain sprites on the stage. The three sprites in this case could be invisible rectangles in different parts of the stage. Putting additional subdivisions within each of the frames lets you work with more sprites than there are available channels:

```
on enterFrame

   if rollOver(1) then go to frame "Left"

   if rollOver(2) then go to frame "Middle"

   if rollOver(3) then go to frame "Right"

end enterFrame
```

romanLingo (Lingo)

Syntax: `the romanLingo`

This property specifies whether Lingo uses a single-byte or double-byte interpreter.

- When `the romanLingo` is TRUE, Lingo uses a single-byte interpreter.
- When `the romanLingo` is FALSE, Lingo uses a double-byte interpreter.

The Lingo interpreter is faster with single-byte character sets. For example, some versions of Macintosh system software, such as Japanese, use a double-byte character set. U.S. system software uses a single-byte character set. Normally, `the romanLingo` is set when starting up Director and is determined by the local version of Macintosh system software.

If you are using a non-roman script system but don't use any double-byte characters in your script, set this property to TRUE to get faster execution of your Lingo scripts.

Example

Lingo example: romanLingo

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the romanLingo to TRUE, which has Lingo use a single-byte character set:

```
set the romanLingo to TRUE
```

saveMovie (Lingo)

Syntax: `saveMovie [`*pathname*`:`*filename*`]`

This command saves the current movie. Including the optional parameter saves the movie to the file specified by *pathname*:*filename*.

Example

Lingo example: saveMovie

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement saves the current movie to the file Update:

```
saveMovie "Update"
```

scoreColor (Lingo)

Syntax: `the scoreColor of sprite` *whichSprite*

This sprite property indicates the score color assigned to the sprite specified by *whichSprite*. The possible values correspond to color chips 0 to 5 in the current palette.

You can use this property to identify groups of cells in the score by assigning them the same score color. You can then write handlers that apply to the items in a group by first testing for items that have a specific score color assigned.

The `scoreColor of sprite` property can be tested but not set.

Example

Lingo example: scoreColor

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement has the message window display the value for the score color assigned to sprite 7:

```
put the scoreColor of sprite 7
```

script of menuItem (Lingo)

Syntax: `the script of menuItem` *whichItem* `of menu` *whichMenu*

This menu item property determines which Lingo statement is executed when the specified menu item is selected. The *whichItem* expression can be either a menu item name or a menu item number; the *whichMenu* expression can be either a menu name or a menu number.

When the menu is installed, the script is set to the text following the "Å" character in the menu definition.

The script property can be tested and set.

Example

Related topics: checkMark of menuItem and enabled of menuItem properties; installMenu command; menu keyword

Lingo example: script of menuItem

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement makes the handler named goHandler the handler that is executed when the user chooses the command Go from the custom menu Control:

```
set the script of menuItem "Go" of menu ¬
   "Control" to "goHandler"
```

scriptNum of sprite (Lingo)

Syntax: `scriptNum of sprite` *whichSprite*

This sprite property indicates the number of the script assigned to the sprite specified by *whichSprite*.

The `scriptNum of sprite` property can be tested, but not set.

Example

Lingo example: scriptNum of sprite

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement displays the number of the script attached to sprite 4:

```
put the scriptNum of sprite 4
```

scriptText of cast (Lingo)

Syntax: `the scriptText of cast` *whichCastMember*

This cast property indicates the text of the script, if any, assigned to the cast member specified by *whichCastMember*.

The scriptText of cast property can be tested and set.

Example

Lingo example: scriptText of cast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement makes the contents of text cast member 20 the script of cast member 30:

```
set the scriptText of cast 30 = the text of cast 20
```

searchCurrentFolder (Lingo)

Syntax: `the searchCurrentFolder`

This function determines whether Director searches the movie's current folder first when searching for filenames.

- When `the searchCurrentFolder` function is TRUE (1), Director searches movie's current folder first when resolving filenames.

- When `the searchCurrentFolder` function is FALSE (0), Director does not search the movie's current folder first when resolving filenames.

The `searchCurrentFolder` function can be tested and set.

Example

Lingo example: searchCurrentFolder

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement has the message window display whether the searchCurrentFolder function is on:

```
put the searchCurrentFolder
```

The result is the number 1, which is the numeric equivalent of TRUE.

This statement sets the searchCurrentFolder function to TRUE, which has Director search the movie's current folder first when resolving filenames:

```
set the searchCurrentFolder to TRUE
```

searchPath (Lingo)

Syntax: `the searchPath`

This function provides a list of the pathnames that are searched when Director resolves filenames. When Director cannot find the file in the current folder, it searches for it in the folders listed in the `searchPath`. The list of pathanmes to search is initially empty each time you launch Director. You must re-create the list of folders that are searched each time you launch Director.

The `searchPath` function can be tested but not set directly. Use the `append` command to add pathnames to the list.

Example

Related topic: searchCurrentFolder function

Lingo example: searchPath

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement displays the pathnames that Director searches when resolving filenames:

```
put the searchPath
```

selection (Lingo)

Syntax: `the selection`

This function returns a string containing the highlighted portion of the currently editable text field. It is useful for testing what a user has selected in a text field.

The `selection` function only determines which text is selected; you cannot used `the selection` to select text.

Example

Related topics: selStart and selEnd properties

Lingo example: selection

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether any text is selected and displays the alert "Please select a word." if none is:

```
if the selection = EMPTY then ¬
   alert "Please select a word."
```

selEnd (Lingo)

Syntax: `the selEnd`

This text property specifies the ending character of a selection. It is used with `the selStart` to determine a selection from the currently editable text, counting from the beginning character.

The `selEnd` text property can be tested and set, and the default value is 0.

Example

Related topics: editableText of sprite and hilite commands; selection function; selStart and text of cast properties

Lingo example: selEnd

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

These statements select "cde" from the text "abcdefg":

```
set the selStart to 3set the selEnd to 5
```

This statement calls the handler noSelection when the selEnd is the same as the selStart:

```
if the selEnd = the selStart then noSelection
```

This statement makes a selection 20 characters long as long as there are 20 characters to select:

```
set the selEnd to the selStart + 20
```

selStart (Lingo)

Syntax: `the selStart`

This text property specifies the starting character of a selection. It is used with `the selEnd` to determine a selection from the currently editable text, counting from the beginning character.

The `selStart` text property can be tested and set. The default value is 0.

Example

Related topics: editableText of sprite and hilite commands; selection function; selEnd and text of cast properties

Lingo example: selStart

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

These statements select "cde" from the text "abcdefg":

```
set the selStart to 3set the selEnd to 5
```

This statement calls the handler noSelection when the selEnd is the same as the selStart:

```
if the selEnd = the selStart then noSelection
```

This statement makes a selection 20 characters long as long as there are 20 characters to select:

```
set the selEnd to the selStart + 20
```

set...to, set...= (Lingo)

Syntax: `set the` *property* `to` *expression*
`set the` *property* `=` *expression*
`set` *variable* `to` *expression*
`set` *variable* `=` *expression*

This command evaluates the expression specified by *expression* and puts the result into the *property* specified by property or the variable specified by *variable*.

Example

Related topics: property and instance keywords

Lingo example: set...to, set...=

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the ink effect for sprite 3 to the ink effect specified by the number 8:

```
set the ink of sprite 3 to 8
```

This statement sets the soundEnabled property to the opposite of its current state. When the soundEnabled is TRUE (the sound is on), this statement turns it off. When the soundEnabled is FALSE (the sound is off), this statement turns it on.

```
set the soundEnabled = not (the soundEnabled)
```

This statement sets the variable named vowels to the string "aeiou":

```
set vowels to "aeiou"
```

setaProp (Lingo)

Syntax: `setaProp` *list*, *property*, *newValue*

This command replaces the value assigned to *property* with the value specified by *newValue* in the list specified by *list*. When the property is not already in the list, Lingo adds the new property and value. This command allows you to create duplicate property names.

Example

Lingo example: setaProp

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

These statements create a property in the given list, and then change that property:

```
set x = [#a:1]

setaProp x, #a, 2

put x

-- [#a:2]
```

setAt (Lingo)

Syntax: `setAt` *list*, *orderNumber*, *value*

This command replaces the item specified by *orderNumber* with the *value* specified by value in the list specified by *list*.

- When *orderNumber* is greater than the number of items in a linear list, the list is expanded with blank entries to provide the number of places specified by *orderNumber*.

- When *orderNumber* is greater than the number of items in a property list, an error alert occurs.

Example

Lingo example: setAt

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler assigns a name to the list [12, 34, 6, 7, 45], replaces the fourth item in the list with the value 10, and then displays the result in the message window:

```
on enterFrame

   set vNumbers = [12, 34, 6, 7, 45]

   setAt vnumbers, 4, 10

   put vNumbers

end enterFrame
```

When the handler runs, the message window displays the following:

```
 [12, 34, 6, 10, 45]
```

setCallBack (Lingo)

Syntax: `setCallBack` *XCMDname, value*

This command specifies how Lingo handles unsupported callbacks from the HyperTalk XCMD or XFCN specified by *XCMDname*.

- When *value* is TRUE (1), unsupported callbacks from the specified XCMD or XFCN cause a generic alert to be displayed.

- When *value* is FALSE (0), unsupported callbacks from the specified XCMD or XFCN are ignored.

- When *value* is an object created from a factory, unsupported callbacks from the specified XCMD or XFCN cause various messages to be sent to the object.

Example

Lingo example: setCallBack

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement has Lingo ignore unsupported callbacks from the SuperDuperXCMD command:

```
setCallBack superDuperXCMD, 0
```

setProp (Lingo)

Syntax: `setProp` *list*, *property*, *newValue*

This command replaces the value assigned to property with the value specified by `newValue` in the list specified by list. This command is similar to the `setaProp` command, except that this command gives an error when the property is not already in the list.

Example

Related topics: setaProp command

Lingo example: setProp

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement changes the age property of property list x to 11:

```
set x = [#age:10, #sex:0]

setProp x, #age, 11
```

shiftDown (Lingo)

Syntax: `the shiftDown`

This function indicates whether the user is pressing the Shift key.

- When `the shiftDown` is TRUE, the user is pressing the Shift key.
- When `the shiftDown` is FALSE, the user is not pressing the Shift key.

Example

Related topics: commandDown, controlDown, key, and optionDown functions

Lingo example: shiftDown

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether the Shift key is being pressed and calls the handler doShiftKey if it is:

```
if the shiftDown then doShiftKey (the key)
```

short (Lingo)

See the date and time functions.

showGlobals (Lingo)

Syntax: `showGlobals`

This command displays all global variables and factories, including XObjects, in the message window. It is useful for debugging scripts.

Related topics: clearGlobals and showLocals commands; global keyword

showLocals (Lingo)

Syntax: `showLocals`

This command displays all local variables in the message window. This command can only be used within handlers, parent script, or factory methods.

Local variables in handlers are abandoned after the handler executes. This command is useful for debugging scripts.

Related topics: clearGlobals and showGlobals commands; global keyword

showResFile (Lingo)

Syntax: `showResFile [`*whichFile*`]`

This command, when used on the Macintosh, displays a list in the message window of resources in the resource file specified by the string *whichFile*. The file must already be open. If the resource file is in a different folder than the current movie, *whichFile* must specify a pathname. If no file is specified, all open resource files are listed. Under Windows, the `showResFile` command has no effect.

There may be many open resource files, and the listing may be very long. To cancel the listing, press the mouse button.

Example

Related topics: closeResFile, openResFile, openXlib, and showXlib commands

Lingo example: showResFile

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement displays the resource file Special Fonts:

```
showResFile "Special Fonts"
```

showXlib (Lingo)

Syntax: `showXlib [`*Xlibfilename*`]`

This command creates a message window display of all XObjects in *Xlibfilename* (it must be open), or all open Xlibraries if no file is specified. If the file is in another folder than the current movie, specify the pathname.

Xlibrary files are resource files that contain XCOD (XObjects) resources on the Macintosh or .DLLs under Windows. Because the type of Xlibrary files on the Macintosh and under Windows differs, the list of files that the showXlib command generates can be different on different platforms.

The `mDescribe` method displays on line documentation for an XObject.

To use `mDescribe`:

1. Type showXlib in the message window and press Return.
   This displays all open Xlibrary resource files and all XObjects contained in those Xlibraries.
2. Using the list of XObjects displayed in the message window, type `XObjectName(mDescribe)` and press Return.
   This displays the on-line documentation for that XObject.

Example

Related topics: closeXlib and openXlib commands

Lingo example: showXlib

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement displays the XObjects in the VideoDisc Library:

```
showXlib "VideoDisc Xlibrary"
```

shutDown (Lingo)

Syntax: `shutDown`

This command has different effects on the Macintosh and under Windows.

- On the Macintosh, the `shutDown` command closes all open applications and turns the computer off.
- Under Windows, the `shutDown` command exits Director or the projector and then exits Windows.

Example

Related topics: quit and restart commands

Lingo example: shutDown

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether the user has pressed Control-q and exits Windows if he or she has:

```
if the key = "q" and the commandDown then shutDown
```

sin (Lingo)

Syntax: `sin(`*angle*`)`

This function calculates the sine of the specified angle. The angle must be expressed in radians as a floating-point number.

Example

Lingo example: sin

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

The following statement calculates the sine of pi/2:

```
sin (pi()/2.0) = 1
```

size of cast (Lingo)

Syntax: `the size of cast` *castName*

This cast property permits you to learn the size, in bytes, of a specific cast member number or name. Divide bytes by 1024 to convert to kilobytes.

Example

Lingo example: size of cast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement displays the size of the Shrine cast member in the message window:

```
put the size of cast "Shrine" into field "How Big"
```

sort (Lingo)

Syntax: `sort` *list*

This command puts the items in the list specified by *list* into alphanumeric order.

- When the list is a linear list, the list is sorted by values.
- When the list is a property list, the list is sorted alphabetically by properties.

Once a list is sorted, it maintains its sort order even when you add new variables using the `add` command.

Example

Lingo example: sort

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement puts the list Values, which consists of [#a: 1, #d: 2, #c: 3], into alphanumeric order. The result appears below the statement:

```
put values

-- [#a: 1, #d: 2, #c: 3]

sort Values

put Values

--[#a: 1, #c: 3, #d: 2]
```

soundBusy (Lingo)

Syntax: `soundBusy(`*whichChannel*`)`

This function determines whether a sound is playing in the sound channel specified by *whichChannel*.

- When a sound is playing in the specified sound channel, the `soundBusy` function is TRUE (1).
- When no sound is playing in the specified sound channel, the `soundBusy` function is FALSE (0).

Make sure that you allow enough time for the sound to start playing before using `soundBusy` to check the sound channel.

Example

Related topics: sound playFile and sound stop commands

Lingo example: soundBusy

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether a sound is playing in sound channel 1 and loops in the frame if it is. This would allow the sound to finish before the playback head goes to another frame:

```
if soundBusy(1) then go to the frame
```

sound close (Lingo)

Syntax: `sound close` *soundChannel*

This command stops the sound playing in and then closes the sound channel specified by *soundChannel*.

Example

Lingo example: sound close

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement stops any sound playing in and closes sound channel 1:

```
sound close 1
```

soundEnabled (Lingo)

Syntax: `the soundEnabled`

This property determines whether the sound is on or off. TRUE means that the sound is on.

The `soundEnabled` property can be tested and set; and the default value is TRUE. When you set this property to FALSE, the volume setting of the sound is not changed but you do not hear the sound.

Example

Related topics: soundLevel, volume of sound, volume of sprite properties

Lingo example: soundEnabled

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the soundEnabled property to the opposite of its current setting. It turns the sound on if it is off and turns it off if it is on:

```
set the soundEnabled to not (the soundEnabled)
```

sound fadeIn (Lingo)

Syntax: `sound fadeIn` *whichChannel*
`sound fadeIn` whichChannel, ticks

This command fades in a sound in the specified sound channel over a period of frames or ticks.

- When ticks is specified, then the fade in occur s evenly over that period of time.

- When ticks is not specified, the default number of ticks is calculated as 15 * (60 / (Tempo setting)) based on the Tempo setting for the first frame of the fade in.

The fade in continues at a predetermined rate until the number of ticks has elapsed, or the sound in the specified channel changes.

Example

Related topics: sound fadeOut command

Lingo example: sound fadeIn

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement fades in the sound in channel 1 over 5 seconds:

```
sound fadeIn 1, 5 * 60
```

sound fadeOut (Lingo)

Syntax: `sound fadeOut` *whichChannel*
`sound fadeOut` *whichChannel, ticks*

This command fades out a sound in the specified sound channel over a period of frames or ticks.

- When ticks is specified, then the fade out occurs evenly over that period of time.

- When ticks is not specified, the default number of ticks is calculated as 15 * (60 / (Tempo setting)) based on the Tempo setting for the first frame of the fade out.

The fadeout continues at a predetermined rate until the number of ticks has elapsed, or the sound in the specified channel changes.

Example

Related topics: sound fadeIn command

Lingo example: sound fadeOut

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement fades out the sound in channel 1 over 5 seconds:

```
sound fadeOut 1, 5 * 60
```

soundLevel (Lingo)

Syntax: `the soundLevel`

This property determines the volume level of the sound that is played through the computer's speaker. Settings range from 0 (no sound) to 7 (maximum sound volume).

It is better to use `the volume of sound` property instead of controlling the sound volume on a channel-by-channel basis.

The `soundLevel` property can be tested and set. The default value is 7.

Example

Related topics: soundEnabled property; volume of sound property

Lingo example: soundLevel

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the variable oldSound equal to the current sound level:

```
put the soundLevel into oldSound
```

This statement sets the sound level to 5:

```
set the soundLevel to 5
```

sound of cast (Lingo)

Syntax: `the sound of cast` *castMember* `to` *onOrOff*

This cast property controls the audio output of the digital video cast member specified by *castMember*.

- When `the sound of cast` is set to 1, sound for the cast member is turned on.
- When `the sound of cast` is set to 0, sound for the cast member is turned off.

Example

Lingo example: sound of cast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement turns on the sound for the cast member Movie Clip:

```
set the sound of cast "Movie Clip" to 1
```

sound playFile (Lingo)

Syntax: `sound playFile` *whichChannel*, *whichFile*

This command plays the AIFF or WAVE sound located at *whichFile* in the sound channel specified by *whichChannel*.

When the sound file is in a different folder than the movie, *whichFile* must specify the full pathname to the file.

The `sound playFile` command streams files from disk rather than playing them from RAM the way Director plays sound cast members. As a result, using the `sound playFile` command when playing digital video or when loading cast members into memory can cause conflicts when the computer tries to read the disk in two places at once.

On the Macintosh, this command requires System 6.0.7 or later to work; otherwise the sound playback will not occur.

Example

Related topics: sound stop command

Lingo example: sound playFile

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement plays the file named Thunder in channel 1:

```
sound playFile 1, "Thunder"
```

This statement plays the file named Thunder in channel 3:

```
sound playFile 3, the pathName &"Thunder"
```

sound stop (Lingo)

Syntax: `sound stop` *whichChannel*

This command stops the playing of the sound playing in the specified channel.

Example

Related topics: soundBusy function

Lingo example: sound stop

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether a sound is playing in sound channel 1 and stops the sound if it is:

```
if soundBusy(1) then sound stop 1
```

sourceRect (Lingo)

Syntax: `the sourceRect of window` *whichWindow*

This window property specifies the coordinates of the rectangle that the movie in the window specified by *whichWindow* was originally created for.

Example

Lingo example: sourceRect

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement displays the original coordinates of the movie Panel in the message window:

```
put the sourceRect of "Panel"
```

sprite (Lingo)

Syntax: `the` *property* `of sprite` *whichSprite*

This keyword tells Lingo that the value specified by *whichSprite* is a sprite number. It is used with every sprite property.

A sprite is an occurrence of a cast member in an animation channel of the score.

Example

Related topics: cast keyword; puppetSprite command

Lingo example: sprite

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the variable named horizontal to the locH of sprite 1:

```
put the locH of sprite 1 into horizontal
```

This statement turns on the puppet condition for the sprite that has sprite number i + 1:

```
set the puppet of sprite (i + 1) to TRUE
```

sprite...intersects (Lingo)

Syntax: `sprite` *sprite1* `intersects` *sprite2*

This operator compares the position of two sprites. It is true if the bounding rectangle of *sprite1* touches the bounding rectangle of *sprite2*.

If both sprites have matte ink, their actual outlines are used, not the bounding rectangles. A sprite's outline is defined by the non-white pixels that make up its border.

This is a comparison operator with a precedence level of 5.

Example

Related topics: sprite within comparison operator

Lingo example: sprite...intersects

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether two sprites intersect and changes the contents of the text cast member Notice to "You placed it correctly." if they do:

```
if sprite i intersects j then ¬put "You placed it correctly." into field
"Notice"
```

sprite...within (Lingo)

Syntax: `sprite` *sprite1* `within` *sprite2*

This comparison operator compares the position of two sprites. It is true if the bounding rectangle of *sprite1* is entirely inside the bounding rectangle of *sprite2*.

If both sprites have matte ink, their actual outlines are used, not the bounding rectangles. A sprite's outline is defined by the non-white pixels that make up its border.

This is a comparison operator with a precedence level of 5.

Example

Related topics: sprite intersects comparison operator

Lingo example: sprite...within

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether two sprites intersect and calls the handler doInside if they do:

```
if sprite 3 within 2 boundary ¬
   then doInside
```

spriteBox (Lingo)

Syntax: `spriteBox` *whichSprite*`,` *left*`,` *top*`,` *right*`,` *bottom*

This command sets the bounding rectangle coordinates of the puppet sprite specified by the integer expression *whichSprite*. The `spriteBox` command gives you a way to set the left, top, right, and bottom sprite properties of a sprite directly without having to convert it into `locH`, `locV`, `width`, and `height`. This is useful because the left, top, right, and bottom sprite properties cannot be set directly.

This command works only on puppet sprites. For bitmap sprites, the `stretch of sprite` property must be TRUE to use this command.

A sprite's coordinates change based on their registration points. For bitmap sprites, it may be necessary to move the registration points in order to obtain proper results.

Example

Related topics: bottom of sprite, height of sprite, left of sprite, rect of cast, right of sprite, stretch of sprite, top of sprite, and width of sprite sprite properties; puppetSprite and updateStage command

Lingo example: spriteBox

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the coordinates of sprite 3's bounding rectangle to 50, 50, 200, and 250:

```
spriteBox 3, 50, 50, 200, 250
```

This statement sets the bounding rectangle of the sprite whose number is mySprite to the starting values and the current cursor location. This creates a rectangle that stretches from the specifed point to the mouse cursor:

```
spriteBox mySprite, ¬
   startH, startV, the mouseH, the mouseV
```

sqrt (Lingo)

Syntax: `the sqrt(`*number*`)`
`the sqrt of` *number*

This function yields the square root of the number specified by *number*.

- When *number* is a floating-point number, the result is a floating- point number.
- When *number* is an integer, the result is rounded to the nearest integer.

Example

Related topic: floatPrecision property

Lingo example: sqrt

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement displays the square root of 3.0 in the message window:

```
put sqrt(3.0)

-- 1.7321
```

stage (Lingo)

Syntax: `the stage`

This system property is used to refer to the main movie in commands and functions that relate to movies in a window. This is useful when using the `tell` command to send a message to the main movie from a child movie.

Example

Lingo example: stage

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement causes the main movie to stop animating:

```
tell the stage to pause
```

This statement displays the current setting of the stage:

```
put the rect of the stage
--rect (0, 0, 640, 480)
```

stageBottom (Lingo)

Syntax: `the stageBottom`

This function--along with `the stageLeft`, `the stageRight`, and `the stageTop` --indicates where the stage is positioned on the desktop. It returns the bottom vertical coordinate of the stage, relative to the upper left corner of the main screen. The height of the stage in pixels is given by `the stageBottom - the stageTop`.

The `stageBottom` function can be tested but not set.

Example

Note: Sprite coordinates are expressed relative to the upper-left corner of the Stage.

Related topics: stageLeft, stageRight, and stageTop functions; locH of sprite and locV of sprite sprite properties

Lingo example: stageBottom

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

These two statements position sprite 3 a distance of 50 pixels from the bottom edge of the stage:

```
put the stageBottom - the stageTop into ¬
   stageHeight
set the locV of sprite 3 to stageHeight - 50
```

stageColor (Lingo)

Syntax: `the stageColor`

This property determines the color of the movie background.

The value of the `stageColor` ranges from 0 to 255 for 8-bit color, or from 0 to 15 for 4-bit color. You can click a color in the color palette to see that color's index number in the lower left corner of the window. Setting the `stageColor` in a Lingo script is equivalent to choosing the stage color from the pop-up palette in the panel window.

The stageColor property can be tested and set. The default value is 0 (white).

Example

Related topics: backColor of sprite and foreColor of sprite properties

Lingo example: stageColor

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the variable oldColor to the index number of the current stage color:

```
put the stageColor into oldColor
```

This statement sets the stage color to the color assigned to chip 249 on the current palette:

```
set the stageColor to 249
```

stageLeft (Lingo)

Syntax: `the stageLeft`

This function--along with `the stageRight`, `the stageTop`, and `the stageBottom`--indicates where the stage is positioned on the desktop. It equals the left horizontal coordinate of the stage, relative to the upper left corner of the main screen. When the stage is flush with the left side of the main screen, this coordinate is zero.

The `stageLeft` function can be tested but not set.

Sprite coordinates are expressed relative to the upper-left corner of the Stage.

Related topics: stageBottom, stageRight, and stageTop, functions; locH of sprite and locV of sprite sprite properties

Lingo example: stageLeft

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether the left edge of the stage is beyond the left edge of the screen and calls the handler leftMonitorProcedure if it is:

```
if the stageLeft < 0 then leftMonitorProcedure
```

stageRight (Lingo)

Syntax: `the stageRight`

This function--along with `the stageLeft`, `the stageTop`, and `the stageBottom`--indicates where the stage is positioned on the desktop. It returns the right horizontal coordinate of the stage, relative to the upper left corner of the main screen's desktop. The width of the stage in pixels is given by `the stageRight - the stageLeft`.

The `stageRight` function can be tested but not set.

Example

Related topics: stageBottom, stageLeft, and stageTop functions; locH of sprite and locV of sprite sprite properties

Lingo example: stageRight

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

These two statements position sprite 3 a distance of 50 pixels from the right edge of the stage:

```
put the stageRight - the stageLeft into stageWidthset the locH of sprite 3 to
stageWidth - 50
```

stageTop (Lingo)

Syntax: `the stageTop`

This function--along with `the stageBottom`, `the stageLeft`, and `the stageRight`--indicates where the stage is positioned on the desktop. It returns the top vertical coordinate of the stage, relative to the upper left corner of the main screen's desktop. If the stage is in the upper left corner of the main screen, this coordinate is zero.

The `stageTop` function can be tested but not set.

Example

Related topics: stageBottom, stageLeft, and stageRight functions; locH of sprite and locV of sprite sprite properties

Lingo example: stageTop

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether the top of the stage is beyond the top of the screen and calls the handler upperMonitorProcedure if it is:

```
if the stageTop < 0 then upperMonitorProcedure
```

startMovie (Lingo)

See on startMovie movie handler.

starts (Lingo)

Syntax: *string1* `starts` *string2*

This comparison operator compares two strings.

- When *string1* starts with *string2*, the condition is TRUE (1).
- When *string1* does not start with *string2*, the condition is FALSE (0).

The string comparison is not sensitive to case or diacritical marks; "a" and "Å" are considered the same.

This is a comparison operator with a precedence level of 1.

Example

Related topics: contains comparison operator

Lingo example: starts

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement has the message window display whether the word Macromedia starts with the string Macro:

```
put "Macromedia" starts "Macro"
```

The result is 1, which is the numerical equivalent of TRUE.

startTime of sprite (Lingo)

Syntax: `the startTime of sprite` *spriteNumber*

This sprite property sets the beginning of a digital video movie in the specified sprite channel. The value of `the startTime` is measured in ticks.

When a digital video movie is played, `the startTime` determines where playback begins.

Example

Related topics: duration of cast property; movieRate of sprite and movieTime of sprite sprite properties

Lingo example: startTime of sprite

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement has the digital video movie in sprite channel 5 start playing at 100 ticks into the movie:

```
set the startTime of sprite 5 to 100
```

startTimer (Lingo)

Syntax: `startTimer`

This command sets the timer property to zero.

Example

Related topics: lastClick, lastEvent, lastKey, and lastRoll functions; timeoutLength, timeoutMouse, timeoutPlay, timeoutScript, and timer properties

Lingo example: startTimer

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler set the timer to zero when a key is pressed:

```
on keyDown

   startTimer

end keyDown
```

stillDown (Lingo)

Syntax: `the stillDown`

This function indicates whether the user is pressing the mouse button.

- When the user is pressing the mouse button, `the stillDown` is TRUE.
- When the user is not pressing the mouse button, `the stillDown` is FALSE.

This function is useful within a `mouseDown` script to trigger certain events only after the `mouseUp`.

Lingo cannot test `the stillDown` when it is used inside a repeat loop. Use `the mouseDown` function inside of repeat loops instead.

Related topics: mouseDown function

Lingo example: stillDown

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether the mouse button is being pressed and calls the handler dragProcedure if it is:

```
if the stillDown then dragProcedure
```

stopTime of sprite (Lingo)

Syntax: `the stopTime of sprite` *whichSprite*

This property specifies the end of a digital video movie that has the sprite number specified by *spriteNumber*. The value of the `stopTime` is measured in ticks.

When a digital video movie is played, the `stopTime` is where playback halts or loops if the loop property is turned on.

Example

Related topics: movieRate of sprite, movieTime of sprite, and startTime of sprite properties

Lingo example: stopTime of sprite

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement has the digital video movie in sprite channel 5 stop playing at 1500 ticks into the digital video movie:

```
set the stopTime of sprite 5 to 1500
```

stretch of sprite (Lingo)

Syntax: `the stretch of sprite` *whichSprite*

This sprite property determines whether the bitmap sprite specified by *whichSprite* can be stretched by using the `spriteBox` command or `the width of sprite` and `height of sprite` properties. If it is True, the bitmap sprite can be stretched.

The `stretch of sprite` property can be tested and set, and the default value is FALSE. When FALSE, the bitmap sprite always stays at its default or normal size.

The `stretch of sprite` property applies to bitmap, digital video, and picture cast members, but not to shape, text, or button cast members. Shapes can be stretched at any time by setting their `height of sprite` and `width of sprite` properties, regardless of the setting of their `stretch` property. Text and button cast members cannot be stretched in any case.

Director requires much more processor time to draw stretched sprites than regular sprites, which can affect movie performance.

In order to have its properties set using Lingo, the sprite must be a puppet.

Example

Related topics: spriteBox and updateStage commands; height of sprite and width of sprite properties

Lingo example: stretch of sprite

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether sprite 3 is stretchable and sets the sprite's width to 10 pixels if it is:

```
if the stretch of sprite 3 = TRUE then ¬  set the width of sprite 3 to 10
```

string (Lingo)

Syntax: `string(`*expression*`)`

This function converts an integer, floating-point, or symbol expression to a string.

Example

Related topics: value function

Lingo example: string

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement adds 2 + 2 and has the message window display the results:

```
put string(2 + 2)
```

This statement converts the number 123 to a string:

```
put string(123)

-- "123"
```

stringP (Lingo)

Syntax: `stringP(`*expression*`)`

This function determines whether the expression specified by *expression* is a string.

- When the expression is a string, the result is TRUE.
- When the expression is not a string, the result is FALSE.

The "P" in stringP stands for predicate.

Example

Related topics: floatP, ilk, integerP, objectP, and symbolP functions

Lingo example: stringP

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether "3" is a string:

```
put stringP("3")
```

The result is 1, which is the numeric equivalent of TRUE.

This statement checks whether the floating-point number 3.0 is a string:

```
put stringP(3.0)?

-- 0
```

Because 3.0 is a floating-point number and not a string, the result is 0, which is the numeric equivalent of FALSE.

switchColorDepth (Lingo)

Syntax: `the switchColorDepth`

This property determines whether Director automatically switches the color depth when loading a movie.

- When `the switchColorDepth` is TRUE, Director switches the monitor(s) that the stage occupies to the color depth of the movie that is being loaded.

- When `the switchColorDepth` is FALSE, Director leaves the color depth of the monitor(s) unchanged when a movie is loaded.

When `the switchcolorDepth` is TRUE, nothing happens until a new movie is loaded.

Setting the monitor's color depth to that of the movie is good practice.

- When the monitor's color depth is set below that of the movie, resetting it to the color depth of the movie (assuming that the monitor can provide that color depth) helps maintain the movie's original appearance.

- When the monitor's color depth is higher than that of the movie, reducing the color depth lets you use the minimum amount of memory to play movies. At minimum memory, loading cast members is more efficient and animation can occur faster.

The `switchColorDepth` property can be tested and set. The default value is the setting for the Switch Monitor's Color Depth to Match Movie's checkbox in the Preferences dialog box.

Example

Related topics: colorDepth property; colorQD function

Lingo example: switchColorDepth

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the variable named switcher to the current setting of switchColorDepth:

```
put the switchColorDepth into switcher
```

This statement checks whether the current color depth is 8-bit and turns the switchColorDepth property on if it is:

```
if the colorDepth = 8 then ¬
   set the switchColorDepth to TRUE
```

symbolP (Lingo)

Syntax: `symbolP(`*expression*`)`

This function determines whether the expression specified by *expression* is a symbol.

- When the expression is a symbol, the result is TRUE.
- When the expression is not a symbol, the result is FALSE.

The "P" in symbolP stands for predicate.

Example

Lingo example: symbolP

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether #3 is a string:

```
put stringP(#3)
```

TAB (Lingo)

Syntax: `TAB`

This character constant represents the Tab key.

Example

Related topics: BACKSPACE, EMPTY, RETURN character constants

Lingo example: TAB

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether the character typed is the Tab character and calls the handler doNextField if it is:

```
if the key = TAB then doNextField
```

This statement advances or retreats the playback head on Tab and Shift-Tab:

```
if the key = TAB then
   if the shiftDown then
      go the frame -1
   else
      go the frame +1
   end if
end if
```

tan (Lingo)

Syntax: `tan(`*angle*`)`

This function yields the tangent of the specified angle. The angle must be expressed in radians as a floating-point number. A radian is an arc in a circle, equal in length to the radius. It is 57.295 degrees. There are 2¹ or 6.2833 radians in a circle.

Example

Lingo example: tan

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

The following function yields the tangent of pi()/4:

```
tan (pi()/4.0) = 1
```

tell (Lingo)

Syntax: `tell` *object* `to` *statement*

or

```
tell object
    statement(s)
end tell
```

The `tell` command is useful for allowing movies to interact. It can be used within a main movie to send a message to a movie playing in a window, or to send a message from a movie playing in a window to the main movie. For example, the `tell` command can let a button in a control panel call a handler in a movie playing in a window. The movie playing in a window could react to the first movie handler by executing the handler. The movie playing in the window could interact with the main movie by sending some value back to the movie.

When you use the `tell` command to send a message to a movie playing in a window, it is important to use the full path name or the window number (in the `windowList`) as the object name. Because opening and closing windows may change the order of the `windowList`, it is a good idea to store the full path name as a global variable. When you do, you can close the window `the stopMovie` handler of the main movie.

When you use the tell command to call a handler in another movie, make sure that you do not have a handler by the same name in the same script in the local movie. If you do, the local script will be called. This applies only to handlers in the same script in which you are using the tell command.

Example

Lingo example: tell

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement has the window Control Panel instruct the movie Simulation to branch to another frame:

```
tell window "Simulation" to go to frame "Save"
```

These statements instruct the movie playing in the childMovie window to continue playing:

```
global childMovie

put the pathName & "Simulation" into childMovie

open window childMovie

tell window childMovie to continue
```

In this set of statements, the tell command sends a series of instructions to the movie playing in the childMovie window. Note that a multiple-line tell command resembles a handler. It needs an end statement:

```
global childMovie

tell window childMovie

   go to frame 5

   set the stageColor to 100

   set the castNum of sprite 4 to 45

   updateStage

end tell
```

In this example, the tell command instructs the movie playing in the childMovie window to execute the calcBalance handler, to put the result into the myBalance global variable, and to display the result in the message window:

```
global childMovie

tell window childMovie to calcBalance

put the result into myBalance

put myBalance

-- $17,300
```

When you use the tell command to send a message from a movie playing in a window to the main movie, use the system propertythe stage as the object name, as in this example:

```
tell the stage to go to frame "Main menu"
```

text of cast (Lingo)

Syntax: `the text of cast` *whichCastMember*

This cast property determines the string that is the text contained in the text cast member specified by *whichCastMember*.

The `text of cast` property is useful for displaying messages and recording what the user types.

The `text` cast property can be tested and set.

Note that any Lingo change to the text of a cast member removes any special formatting you have applied to individual words or lines. Altering `the text of cast` reapplies global formatting.

Example

Related topics: selEnd and selStart text properties;

Lingo example: text of cast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement places the phrase "Thank you." in the empty text cast member Response:

```
if the text of cast "Response" = EMPTY then ¬
   set the text of cast "Response" to "Thank You."
```

This statement sets the text of cast member Notice to "You have made the right decision."

```
set the text of cast "Notice" = "You have ¬
   made the right decision!"
```

textAlign of field (Lingo)

Syntax: `the textAlign of field` *whichField*
`the textAlign of field` *whichField*

This text property determines the alignment used to display text within the specified text cast member.

The value of the property is a string consisting of one of the following: "left," "center," or "right." The parameter *whichField* can be either a cast name or a cast number.

The `textAlign of field` property can be tested and set.

The text cast member must contain text, if only a space, to use the `textAlign of field` property. It has no effect on a cast member that contains no text.

Example

Note: This property requires that the text castmember already contain text, if only a space. It will not affect a castmember that contains no text.

Related topics: text of cast property; textFont of field, textHeight of field, textSize of field, and textStyle of field text properties; & (ampersand) and && (double ampersand) text operators

Lingo example: textAlign of field

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the variable named alignment to the current textAlign of field setting for the text cast member Rokujo Speaks:

```
put the textAlign of field "Rokujo Speaks" into ¬
   alignment
```

This repeat loop consecutively sets the textAlign of the text cast member Rove to left, center, and then right.

```
repeat with i = 1 to 3
   set the textAlign of field "Rove" ¬
   to word i of "left center right"
end repeat
```

textFont of field (Lingo)

Syntax: `the textFont of field` *whichField*

This text property determines the type of font used to display the specified text cast member. The parameter *whichField* can be either a cast member name or number.

The `textFont of field` text property can be set, affecting every line in the text field. When tested, it returns the height of the first line of text.

The text cast member must contain text, if only a space, to use the `textFont of field` property. It has no effect on a cast member that contains no text.

Example

Related topics: text of cast property; textAlign of field, textHeight of field, textSize of field, and textStyle of field text properties

Lingo example: textFont of field

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the variable named oldFont to the current textFont of field setting for the text cast member Rokujo Speaks:

```
put the textFont of field "Rokujo Speaks" into oldFont
```

textHeight of field (Lingo)

Syntax: `the textHeight of field` *whichField*

This text property determines the line spacing used to display the specified text cast member. The parameter *whichField* can be either a cast member name or number.

The `textHeight of field` property can be tested and set.

The text cast member must contain text, if only a space, to use `the textHeight of field` property. It has no effect on a cast member that contains no text.

Example

Related topics: text of cast property; textAlign of field, textFont of field, textSize of field, and textStyle of field text properties

Lingo example: textHeight of field

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

```
put the textHeight of field "Rokujo Speaks" into ¬
   oldHeight
```

textSize of field (Lingo)

Syntax: `the textSize of field` *whichField*

This text property determines the size of the font used to display the specified text cast member. The parameter *whichField* can be either a cast member name or number.

The `textSize` text property can be tested and set.

The text cast member must contain text, if only a space, to use `the textSize of field` property. It has no effect on a cast member that contains no text.

Example

Note: This property requires that the text castmember already contain text, if only a space. It will not affect a castmember that contains no text.

Related topics: text of cast property; textAlign of field, textFont of field, textHeight of field, and textStyle of field text properties

Lingo example: textSize of field

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the variable named oldSize to the current textSize of field setting for the text cast member Rokujo Speaks:

```
put the textSize of field "Rokujo Speaks" into ¬
   oldSize
```

textStyle of field (Lingo)

Syntax: `the textStyle of field` *whichField*

This text property determines the styles applied to the font used to display the specified text cast member.

The value of the property is a string of styles delimited by commas. Lingo uses a font that is a combination of the styles in the string. The available styles are plain, bold, italic, underline, shadow, outline, condense, and extend. In addition, you can use the word normal to remove all of the styles that are currently applied. The parameter *whichField* can be either a cast member name or number.

The text cast member must contain text, if only a space, to use `the textStyle of field` property. It has no effect on a cast member that contains no text.

The `textStyle of field` text property can be tested and set.

The text cast member must contain text, if only a space, to use `the textStyle of field` property. It has no effect on a cast member that contains no text.

Example

Related topics: text of cast property; textAlign of field, textFont of field, textHeight of field, and textSize of field text properties

Lingo example: textStyle of field

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the variable named oldStyle to the current textStyle of field setting for the text cast member Rokujo Speaks:

```
put the textStyle of field "Rokujo" into oldStyle
```

This statement sets the textStyle of field setting for the text cast member Rokujo Speaks to bold italic:

```
set the textStyle of field "Poem" to "bold, italic"
```

the (Lingo)

Syntax: `the` *property*

All Lingo properties and many sprite properties and functions require the keyword the to precede the property. This distinguishes the property from a variable or object name.

Properties have "super-global" scope, which means they are available within handlers and methods even without a global declaration. Like global variables, Lingo system properties are available between different movies in the same presentation (unless they are changed by system events). Movie properties would change when a new movie is loaded.

ticks (Lingo)

Syntax: `the ticks`

This function returns the current time in ticks (1/60th of a second). Counting ticks begins from the time the computer is started.

Example

Related topics: time and timer functions

Lingo example: ticks

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement converts ticks to minutes by dividing the number of ticks by 60 twice and then sets the variable minutesOn to the result:

```
put the ticks/60/60 into minutesOn
```

time (Lingo)

Syntax:
```
the time
the short time
the long time
the abbreviated time
the abbrev time
the abbr time
```

This function returns the current time in the system clock as a string in one of three formats: short, long, or abbreviated. If no format is specified, the default is short. The abbreviated format can also be referred to as `abbrev` and `abbr`. In the United States, the short and abbreviated formats are the same.

Example

Note: The three time formats vary, depending on the country for which your System file was designed. The examples above are for the United States.

Related topics: date function

Lingo example: time

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

These statements have the message window display the time in different formats. Possible results appear below each statement:

```
put the abbreviated time"1:30 PM"

put the long time"1:30:24 PM"

put the short time"1:30 PM"
```

timeoutKeyDown (Lingo)

Syntax: `the timeoutKeyDown`

When this property is TRUE, `keyDown` events set the `timeoutLapsed` property to zero.

The `timeoutKeyDown` property can be tested and set. The default value is TRUE.

Example

Related topic: keyDownScript property

Lingo example: timeoutKeyDown

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the variable timing to the value of the timeoutKeyDown:

```
put the timeoutKeyDown into timing
```

This statement turns off the timeoutKeyDown:

```
set the timeoutKeyDown to FALSE
```

timeoutLapsed (Lingo)

Syntax: `the timeoutLapsed`

This property indicates the number of ticks elapsed since the last timeout. A timeout event occurs when the `timeoutLapsed` property reaches the time specified by the `timeoutLength` property.

The `timeoutLapsed` property can be tested, but not set directly in Lingo.

Example

Lingo example: timeoutLapsed

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the text of field Countdown to the value of the timeoutLapsed property. (Dividing the timeOutLapsed by 60 converts it to seconds):

```
put the timeoutLapsed / 60 into field "Countdown"
```

timeoutLength (Lingo)

Syntax: `the timeoutLength`

This property determines the number of ticks before a timeout event occurs. A timeout occurs when the `timeoutLapsed` property reaches the time specified by the `timeoutLength` property.

The `timeoutLength` property can be tested and set. The default value is 10,800 ticks, which is 3 minutes.

Example

Lingo example: timeoutLength

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the timeOutLength to 10 seconds:

```
set the timeoutLength to 10 * 60
```

timeoutMouse (Lingo)

Syntax: `the timeoutMouse`

This property determines whether `mouseDown` events reset the `timeoutLapsed` property to zero. When this property is TRUE, `mouseDown` events reset the `timeoutLapsed` property.

The `timeoutMouse` property can be tested and set. The default value is TRUE.

Example

Related topics: mouseDownScript and mouseUpScript properties

Lingo example: timeoutMouse

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement records the current setting of the timeOutMouse by setting the variable named timing to the timeOutMouse.

```
put the timeoutMouse into timing
```

This statement sets the timeoutMouse property to FALSE. The result is that the timeoutLapsed property keeps its current value when the mouse button is pressed:

```
set the timeoutMouse to FALSE
```

timeoutPlay (Lingo)

Syntax: `the timeoutPlay`

This property determines whether the `timeoutLapsed` property is reset to zero when a movie is played. When `timeoutPlay` is TRUE, playing a movie resets the `timeoutLapsed` property to zero. This allows timeouts to occur only when the animation is paused.

The `timeoutPlay` property can be tested and set. The default value is FALSE.

Example

Lingo example: timeoutPlay

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the timeoutPlay to TRUE, which has Lingo reset the timeoutLapsed property to zero when a movie is played:

```
set the timeoutPlay to true
```

timeoutScript (Lingo)

Syntax: `the timeoutScript`

This property determines the string that is executed as a Lingo statement when a timeout occurs.

Setting the `timeOutScript` property is equivalent to executing a when `timeOut then` command that was used in earlier versions of Director.

When the event script you've assigned is no longer appropriate, turn it off with the statement `set the timeOutScript to EMPTY`.

The `timeOutScript` property can be tested and set. The default value is EMPTY.

Example

Lingo example: timeoutScript

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the timeoutScript to a calling script for the handler timeoutProcedure:

```
set the timeoutScript to "timeoutProcedure"
```

timer (Lingo)

Syntax: `the timer`

This property is a free-running timer that counts time in ticks (60ths of a second). It has nothing to do with the `timeOutScript`. It is only for convenience in timing certain events. The `startTimer` command zeroes the value of the timer property.

The `timer` property is useful for setting up delays within handlers. (The `delay` command works only in frame scripts.) For example, you can use `the timer` to synchronize pictures to a sound track by inserting a delay that makes the movie wait until a sound is finished.

Related topics: lastClick, lastEvent, lastKey, and lastRoll functions; startTimer command

Lingo example: timer

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler creates a 1 second delay:

```
on countTime   startTimer     repeat while the timer < 60         nothing
   end repeatend countTime
```

This statement sets the variable startTicks to the current value of the timer:

```
set the timer = startTicks
```

title of window (Lingo)

Syntax: `the title of window` *whichWindow*

This window property is the title of the window specified by *whichWindow*. This is different than the filename of the movie assigned to the window.

The `title of window` property can be tested and set.

Example

Lingo example: title of window

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement makes Action View the title of window X:

```
set the title of window "X" to "Action View"
```

titleVisible of window (Lingo)

Syntax: `the titleVisible of window` *whichWindow*

This window property specifies whether the window specified by *whichWindow* displays the window title in the window's title bar.

The `titleVisible of window` property can be tested and set.

Example

Lingo example: titleVisible of window

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement display the title of the window Control Panel by setting the window's titleVisible property to TRUE:

```
set the titleVisible of "Control Panel" to TRUE
```

to (Lingo)

The word `to` occurs in a number of Lingo constructs.

See char...of, item...of, line...of, and word...of chunk expression keywords; repeat with, and set...to/set...= commands

top of sprite (Lingo)

Syntax: `the top of sprite` *whichSprite*

This sprite property returns the top vertical coordinate of the bounding rectangle of the sprite specified by *whichSprite*. The coordinate is the number of pixels from the upper left corner of the stage.

The `top of sprite` property can be tested, but not set directly. The top vertical coordinate of a sprite can be set with the `spriteBox` command.

Example

Related topics: bottom of sprite, height of sprite, locH left of sprite, of sprite, locV of sprite, right of sprite, and width of sprite sprite properties; spriteBox command

Lingo example: top of sprite

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether the top of sprite 3 is above the top of the stage and calls the handler offTopEdge if it is:

```
if the top of sprite 3 < 0 then offTopEdge
```

trace (Lingo)

Syntax: `the trace =` *trueOrFalse*

This property specifies whether the movie's trace function is on or off.

- When `the trace` is TRUE (1), the trace function is on.
- When `the trace` is FALSE (0), the trace is off.

Example

Lingo example: trace

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement turns the trace function on:

```
set the trace = TRUE
```

traceLoad (Lingo)

Syntax: `the traceLoad`

This property specifies the amount of information that is displayed about cast members as they are loaded. The possible values for `the traceLoad` property have the following effect:

0--Displays no information

1--Displays cast members' names

2--Display cast members' names, number of the current frame, movie name, and file seek offset

The `traceLoad` property can tested and set.

Example

Lingo example: traceLoad

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement has the movie display the names of cast members as they are loaded:

```
set the traceLoad to 1
```

traceLogFile (Lingo)

Syntax: `the traceLogFile`

This property specifies the name of the file that the message window display is written to. You can close the file by setting the `traceLogFile` property to
EMPTY ("").

Example

Lingo example: traceLogFile

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement has Lingo write the display of the message window to the file Messages:

```
set the traceLogFile = "Messages"
```

This statement closes the file that the message window display is being written to:

```
set the traceLogFile = ""
```

trails of sprite (Lingo)

Syntax: `the trails of sprite` *whichSprite*

This property turns the trails ink effect on and off for the sprite specified by *whichSprite*. In order to set this property, the sprite must have the `puppetSprite` property set to TRUE before setting the `trails` property. Set `the trails` to 0 to turn trails off; set `the trails` to 1 to turn trails on.

To erase trails:

- Animate another sprite across these pixels.
- Use a transition.

Related topics: directToStage cast property

Example

Lingo example: trails of sprite

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the trails on for sprite 7:

```
set the trails of sprite 7 to 1
```

TRUE (Lingo)

Syntax: `TRUE`

This logical constant represents the value of a logically true expression, such as 2 < 3. It has a numerical value of 1.

Example

Related topics: FALSE logical constant

Lingo example: TRUE

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement turns on the soundEnabled property by setting it to TRUE:

```
set the soundEnabled to TRUE
```

type of sprite (Lingo)

Syntax: `the type of sprite` *whichSprite*

This sprite property determines the type of the sprite specified by *whichSprite*. The type can be a bitmap, a shape, a text field, or a button. This command is useful with puppet sprites, for example, to change a shape sprite into a bitmap prior to replacing it with a bitmapped cast member, or to replace one button sprite with another type, or to make it invisible on the stage.

The sprite types are as follows:

0	inactive sprite (turns the sprite off)
1	bitmap
2	rectangle
3	rounded rectangle
4	oval
5	line topleft to bottomright
6	line bottomleft to topright
7	text
8	button
9	checkbox
10	radio button
16	Undetermined. Use castType property to examine the type of cast member associated with the sprite.

Before setting a sprite to type 1 [bitmap],set the `stretch of sprite` property of the sprite to FALSE. This prevents it from stretching to the size of the previous sprite.

When you set this property within a script while the playback head is not moving, be sure to use the command `updateStage` to redraw the stage. When you are changing several sprite properties--or several sprites--you only have to use one `updateStage` command at the end of all the changes.

The `type of sprite` property can be tested and set.

Example

Related topic: stretch of sprite property

Lingo example: type of sprite

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the type of sprite 4 to text:

```
set the type of sprite 4 to 7
```

This statement turns off sprite 1:

```
set the type of sprite 1 to 0
```

union rect (Lingo)

Syntax: `union (`*rect1*`, `*rect2*`)`

This function returns the smallest rectangle that encloses the two rectangles *rect1* and *rect2*.

Example

Lingo example: union rect

```
put union(rect (0, 0, 10, 10), rect (15, 15, 20, 20))-- rect (0, 0, 20, 20)
```

unLoad (Lingo)

Syntax: `unLoad`
`unLoad` *theFrameNum*
`unLoad` *fromFrameNum, toFrameNum*

This command clears the cast members used in a specified frame from memory. When used without an argument, the `unload` command clears the cast members in all the frames of a movie from memory--except any being used in the current frame.

When used with one argument, *theFrameNum*, the `unLoad` command clears from memory the cast members in that frame. director automatically unloads the least recently used cast members to accommodate `preLoad` commands or normal cast loading.

When used with two arguments, *fromFrameNum* and *toFrameNum*, the `unLoad` command unloads all cast members in the range specified. You can specify a range of frames by frame numbers or frame labels.

Example

Related topics: preLoad, preLoadCast, and unloadCast commands; purgePriority of cast property

Lingo example: unLoad

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement clears the cast members used in frame 10 from memory:

```
unLoad 10
```

This statement clears the cast members used from the frame labeled first to the frame labeled last:

```
unLoad "first","last"
```

unloadCast (Lingo)

Syntax: `unLoadCast`
`unLoadCast` *castName*
`unLoadCast` *fromCastName, toCastName*

This command clears the specified cast members from memory. When used without an argument, `unLoadCast` causes all cast members in a movie to be cleared from memory--except for any being used in the current frame.

When used with one argument, *castName*, the `unLoadCast` command clears from memory the cast member name or number that you specify.

When used with two arguments, *fromCastName* and *toCastName*, the `unLoadCast` command unloads all cast members in the range specified.

Example

Related topics: preLoad and preLoadCast commands; purgePriority of cast property

Lingo example: unloadCast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement clears the cast member Screen 1:

```
unLoadCast "Screen1"
```

This statement clears from memory all cast members from cast member 11 to cast member 18:

```
unloadCast 11, 18
```

updateMovieEnabled (Lingo)

Syntax: `the updateMovieEnabled`

This property specifies whether changes made to the current movie are automatically saved when the movie branches to another movie.

- When the `updateMovieEnabled` property is TRUE, changes to the movie are automatically saved when the movie branches to another movie.

- When the `updateMovieEnabled` property is FALSE, changes to the movie are not automatically saved when the movie branches to another movie.

The default value is FALSE.

Example

Lingo example: updateMovieEnabled

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement has Director save changes to the current movie whenever the movie branches to another movie.

```
set the updateMovieEnabled = TRUE
```

updateStage (Lingo)

Syntax: `updateStage`

This command redraws the stage immediately. Normally the stage is updated only between frames, but the `updateStage` command updates the stage any time the command is executed from a handler or factory method.

The `updateStage` command is useful for creating animation within one frame, which is common when animating puppets.

Do not use `updateStage` with the `perFrameHook` property. Otherwise, unexpected results could occur.

Example

Lingo example: updateStage

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This handler makes the sprite specifed by whichSprite a puppet sprite, changes the sprite's horizontal and vertical locations, and redraws the stage so that the sprite appears in the new location:

```
on moveRight whichSprite, howFar

   puppetSprite whichSprite, TRUE

   set the locH of sprite whichSprite ¬

      to the locH of sprite whichSprite + howFar

   updateStage

end moveRight
```

value (Lingo)

Syntax: `value(`*string* `)`

This function returns the numerical value of a string. This is useful when making use of a numerical string that the user has typed into a text cast member or data from XObjects that return numerical strings.

Example

Related topics: string function

Lingo example: value

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement displays the numerical value of the string "the sqrt of" && "2.0":

```
put value("the sqrt of" && "2.0")
```

The result is 1.4142.

This statement displays the numerical value of the string "penny":

```
put value("penny")
```

The resulting display in the message window is <VOID>, because the word penny has no numerical value.

version (Lingo)

Syntax: `global version`

This system variable contains the version string for Macromedia Director. The same string appears the the Finder's Get Info dialog box.

video of cast (Lingo)

Syntax: `the video of cast` *castName*

This cast property enables or disables playing the video that is associated with the cast member.

Example

Lingo example: video of cast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement turns off the video associated with the Interview cast member:

```
set the video of cast "Interview" to FALSE
```

visible of sprite (Lingo)

Syntax: `the visible of sprite` *whichSprite*

This sprite property determines whether the sprite specified by *whichSprite* is visible.

- When the `visible of sprite` property is TRUE, the sprite is visible.
- When the `visible of sprite` property is FALSE, the sprite is not visible.

It is a good idea to make the sprite a puppet before using Lingo to change its visible of sprite property.

The `visible of sprite` property can be tested and set.

Example

Lingo example: visible of sprite

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement makes sprite 8 visible:

```
set the visible of sprite 8 to TRUE
```

visible of window (Lingo)

Syntax: `the visible of window` *whichWindow*

This window property determines whether the window specified by *whichWindow* is visible.

- When the `visible of window` property is TRUE, the window is visible.
- When the `visible of window` property is FALSE, the window is not visible.

The `visible of window` property can be tested and set.

Example

Lingo example: visible of window

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement makes the window Panel visible:

```
set the visible of window "Panel" to TRUE
```

voidP (Lingo)

Syntax: `voidp(`*variableName*`)`

This function specifies whether the variable specified by *variableName* has been given an initial value.

- When the `voidp` property is TRUE, the variable has not been given an initial value.
- When the `voidp` property is FALSE, the variable has been given an initial value.

Example

Lingo example: voidP

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement checks whether the variable answer has been given an initial value:

```
put voidP(answer)
```

volume of sound (Lingo)

Syntax: `the volume of sound` *whichChannel*

This sound property determines the volume of the sound channel specified by *whichChannel*. Sound channels are numbered 1, 2, 3, ... 1 and 2 are the channels that appear in the score.

The value of the `volume of sound` property ranges from 0 (silent) to 255 (maximum volume).

Example

Related topics: fadeIn and fadeOut commands; soundEnabled and soundLevel properties

Lingo example: volume of sound

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the volume of sound channel number i to 130, which is a medium setting:

```
set the volume of sound i to 130
```

volume of sprite (Lingo)

Syntax: `the volume of sprite` *spriteNum*

This property can be used to control the volume of a digital video movie cast member. You can use a cast name or number. The values for volume range from -256 to 256. Values of zero or less are silent.

Example

Related topic: soundLevel property

Lingo example: volume of sprite

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the volume of the digital video movie playing in sprite channel 7 to 256, which is the maximum sound volume:

```
set the volume of sprite 7 to 256
```

when keyDown (Lingo)

Syntax: `when keyDown then` *statement*

This command, which was used in earlier versions of Director, establishes a Lingo statement to be executed each time a key is pressed (at each `keyDown` event). The `when keyDown then` command can only be used in Director 3.1 movies that are opened in Director 4 with the Allow Outdated Lingo checkbox turned on in the Movie Info dailog box.

The statement to be executed must be only one line long. It can be a single command, a one-line test, or--if you need to execute multiple statements when a `keyDown` event occurs--a handler call.

The `keyDown` action remains in effect until you turn it off with `when keyDown then nothing`.

Example

Lingo example: when keyDown

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement causes the computer to beep when the key is pressed:

```
when keyDown then beep
```

This statement causes the movie to advance to the end when the Return key is pressed:

```
when keyDown then ¬
   if the key = return then go to frame "ending"
```

This statement turns off the keyDown action:

```
when keyDown then nothing
```

Note: The keyDown action is automatically turned off when you load a new movie.

See also dontPassEvent command, keyDownScript property, keyCode, key functions

when mouseDown (Lingo)

Syntax: `when mouseDown then` *statement*

This command, which was used in earlier versions of Director, establishes a Lingo statement to be executed each time the mouse button is pressed (at each `mouseDown` event). The `when mouseDown then` command can only be used in Director 3.1 movies that are opened in Director 4 with the Allow Outdated Lingo checkbox turned on in the Movie Info dailog box.

The statement to be executed must be only one line long. It can be a single command, a one-line test, or--if you need to execute multiple statements when a `mouseDown` event occurs--a handler call.

The `mouseDown` action remains in effect until you turn it off with `when mouseDown then nothing`.

This command performs the same function as the mouseDownScript property.

Lingo example: when mouseDown

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement causes the computer to beep when the mouse is pressed:

```
when mouseDown then beep
```

This statement causes the movie to advance to the end when the mouse and option key are pressed:

```
when mouseDown then ¬
   if the optionDown then go to frame "ending"
```

This statement turns off the mouseDown action:

```
when mouseDown then nothing
```

Note: The mouseDown action is automatically turned off when you load a new movie.

See also dontPassEvent command, the mouseDownScript property

when mouseUp (Lingo)

Syntax: `When mouseUp then` *statement*

This command, which was used in earlier versions of Director, establishes a Lingo statement to be executed each time the mouse button is released (at each `mouseUp` event). The `when mouseUp then` command can only be used in Director 3.1 movies that are opened in Director 4 with the Allow Outdated Lingo checkbox turned on in the Movie Info dailog box.

This command establishes a Lingo statement to be executed each time the mouse button is released (at each `mouseUp` event).

The statement to be executed must be only one line long. It can be a single command, a one-line test, or--if you need to execute multiple statements when a `mouseUp` event occurs--a handler call.

The `mouseUp` action remains in effect until you turn it off with `when mouseUp then nothing`.

Example

Lingo example: when mouseUp

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement causes the movie to beep when the mouse is released:

```
when mouseUp then beep
```

This statement causes the movie to advance to the end when the mouse and option key are released:

```
when mouseUp then ¬
   if the optionDown then go to frame "ending"
```

This statement turns off the mouseUp action:

```
when mouseUp then nothing
```

Note The mouseUp action is automatically turned off when you load a new movie.

when timeOut (Lingo)

Syntax: `when timeOut then` *statement*

This command, which was used in earlier versions of Director, establishes a Lingo statement to be executed each time the user doesn't click the mouse, type a key, or play a movie for a specified amount of time (at each `timeOut`). The `when timeOut then` command can only be used in Director 3.1 movies that are opened in Director 4 with the Allow Outdated Lingo checkbox turned on in the Movie Info dailog box.

The length of the timeout is determined by the `timeoutLength` property:

```
set the timeoutLength to numberOfTicks
```

The system keeps track of how long the user has been inactive in the `timeoutLapsed` property. A timeout occurs when the `timeoutLapsed` property reaches the time specified by the `timeoutLength` property. Whenever the user interacts with the system (for example, by pressing the mouse button), the `timeoutLapsed` property is reset to zero. Therefore, the value of the `timeoutLapsed` property usually never reaches the time specified in the `timeoutLength` property while the user is doing things. You can select which actions (such as `mouseDown`, `keyDown`, or playing a movie) reset the `timeoutLapsed` to zero with the following commands:

```
set the timeoutKeydown to true

set the timeoutMouse to true

set the timeoutPlay to true
```

Setting these properties to true means that clicking the mouse, typing a key, or playing a movie resets the `timeoutLapsed` property to zero. Setting them to false means that these events do not reset the `timeoutLapsed` property to zero. The defaults are as follows:

```
timeoutKeydown - true

timeoutMouse - true

timeoutPlay - false
```

You can also set the `timeoutLapsed` property to zero directly via Lingo with this script:

```
set the timeoutLapsed to 0
```

Example

Lingo example: when timeOut

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This example causes the movie to advance to the help frame when the user has not responded within the sepcified time:

```
when timeOut then go to frame "help"
```

This statement cancels a previous when...then command:

```
when timeOut then nothing
```

Note A timeOut action remains in effect even if you go to another movie, so make sure the action is valid for any movies it may be executed in.

width of cast (Lingo)

Syntax: `the width of cast` *whichCastMember*

This cast property determines the width in pixels of the cast member specified by *whichCastMember*. The `width of cast` applies only to bitmap and shape cast members. It does not affect text or button cast members.

Example

Related topics: the height of cast property

Lingo example: width of cast

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement assigns the width of cast member 50 to the variable height:

```
put the width of cast 50 into height
```

width of sprite (Lingo)

Syntax: `the width of sprite` *whichSprite*

This sprite property determines the horizontal size in pixels of the sprite specified by *whichSprite*. The width applies only to bitmap and shape cast members. It does not affect text or button cast members.

The `width of sprite` property can be tested and set.

Setting this property has no effect on bitmap sprites unless the sprite's `stretch of sprite` property is set to TRUE.

When you set this property within a script while the playback head is not moving, be sure to use the `updateStage` command to redraw the stage. When you are changing several sprite properties--or several sprites--you have to use only one `updateStage` command at the end of all the changes.

Example

Related topics: height of sprite and stretch of sprite sprite properties; spriteBox command

Lingo example: width of sprite

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the width of sprite 10 to 26 pixels:

```
set the width of sprite 10 to 26
```

This statement assigns the width of sprite number i + 1 to the variable howWide:

```
put the width of sprite (i + 1) into howWide
```

window (Lingo)

Syntax: `window` *whichWindow*

This keyword refers to the movie window--a window that contains a Director movie--specified by *whichWindow*.

Windows that play movies are useful for creating floating palettes, separate control panels, and windows of different shapes. By using windows that play movies, you can have several movies open at once and allow them to interact.

Example

Related topics: close window, moveToBack, moveToFront, and open window commands

Lingo example: window

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement opens the window Panel:

```
open window "Panel"
```

This statement moves the window Panel to the front:

```
moveToFront window "Panel"
```

windowList (Lingo)

Syntax: `the windowList`

This property is a list of all the known movie windows.

Example

Lingo example: windowList

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement displays all the known movie windows in the message window:

```
put the windowList
```

This statement clears the windowList:

```
set the windowList = []
```

windowType (Lingo)

Syntax: `the windowType of window` *whichWindow*

This window property specifies the display style of the window specified by *whichWindow*. Director adapts the window's appearance for the platform the movie is playing on. These are possible values, which correspond to values in the Macintosh Standard Tool Box numbers.

0--Standard document window
1--Alert box Style window
2--Plain box
3--Plain box with shadow
4--Document window without size box on the Macintosh or maximize and minimize boxes under Windows
8--Document window with zoom box
12--Document window with zoom box, but without size box on the Macintosh or maximize and minimize boxes under Window
16--Window with curved border

Example

Lingo example: windowType

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement sets the value of the display style of the window Panel to 8:

```
set the windowType of window "Panel" to 8
```

word...of (Lingo)

Syntax: `word` *whichWord* `of` *chunkExpression*
`word` *firstWord* `to` *lastWord* `of chunkExpression`

This chunk expression keyword specifies a word or a range of words in a chunk expression. A word chunk is any sequence of characters delimited by spaces. (Any non-visible character--such as a Tab or Return--is considered a space.)

The expressions *whichWord*, *firstWord*, and *lastWord* must evaluate to integers that specify a word in the chunk.

Chunk expressions refer to any character, word, item, or line in any source of text. Sources of text include fields (text cast members) and variables that hold strings.

Example

Related topics: char...of, line...of , and item...of chunk expression keywords; the number of words in in chunk function

Lingo example: word...of

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

These statements set the variable named animalList to the string "fox dog cat" and then insert the word elk before the second word of the list:

```
put "fox dog cat" into animalList

put "elk " before word 2 of animalList
```

The result is the list "fox elk dog cat".

This statement has the message window display the fifth word of the same string:

```
put word 5 of "fox elk dog cat"
```

Because there is no fifth word in this string, the message window displays two quote marks (""), which indicate an empty string.

xFactoryList (Lingo)

Syntax: `xFactoryList(`*whichLibrary*`)`

This function returns a string list of all the currently available XObject factories in the XLibrary file specified by the string *whichLibrary*. The XLibrary must have been previously opened with the `openXlib` command. If you specify EMPTY for *whichLibrary*, this function returns a list of all XObject factories in all open XLibraries.

The XObject factories appear one per line in the returned string list. Each line ends with a Return character.

Example

Lingo example: xFactoryList

To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement displays the XObjects available in the Xlibrary named AppleCD XObj:

```
put xfactoryList("AppleCD XObj")
```

This statement displays the first line of the list of all available XObjects in all open Xlibraries:

```
put line 1 of xfactoryList(EMPTY)
```

zoomBox (Lingo)

Syntax: `zoomBox` *startSprite*`,` *endSprite* `[,` *delayTicks*`]`

This command creates a zooming effect, like the expanding windows in the Finder. The zoom effect starts at the bounding rectangle of *startSprite* and finishes at the bounding rectangle of *endSprite*. `zoomBox` uses the following logic when executing:

1--Looks for *endSprite* in the current frame, otherwise,
2--Looks for *endSprite* in the next frame.

Note, however, that the `zoomBox` command does not work for an *endSprite* in the same channel as *startSprite*.

delayTicks is the delay in ticks between each movement of the zoom rectangles. If *delayTicks* is not specified, the delay is 1.

Example

Lingo example: zoomBox

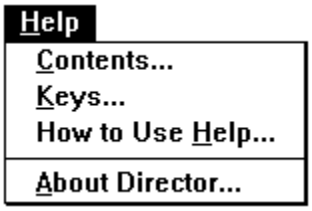
To copy the contents of this topic to the Clipboard, select the Copy command from the Edit menu of the Director Help window.

This statement creates a zoom effect between sprites 7 and 3:

```
zoomBox 7, 3
```

Help menu

Click a menu command for more information:



Contents command

Displays the help window you are viewing now.

Related topic: Using Director Help

Keys command

Displays a summary of the keyboard shortcuts you can use in Director.

Related topics: Keyboard shortcuts

How to Use Help command

Displays information about using Director Help.

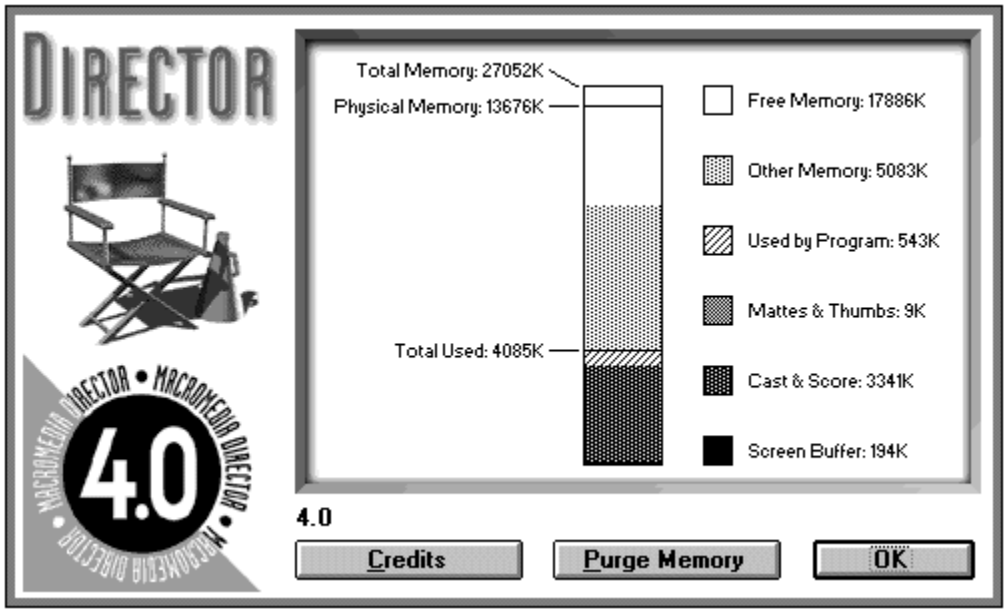
About Director command

Choosing About Director displays a dialog box that shows you how much memory is available to Director for your movie. It also indicates how much memory different parts of the current movie use and the total disk space the movie occupies.

Click and hold here to see an illustration of the dialog box.

Click the name of an element of the About Director box for more information:

- Total Memory
- Total Used
- File Size on Disk
- Other Memory
- Physical Memory
- Memory Limit
- Free Memory
- Used by Program
- Mattes & Thumbs
- Cast & Score
- Screen Buffer
- Credits
- Purge Memory


DIRECTOR
MACROMEDIA DIRECTOR • MACROMEDIA DIRECTOR •
4.0
Total Memory: 27052K
Physical Memory: 13676K
Total Used: 4085K
Free Memory: 17886K
Other Memory: 5083K
Used by Program: 543K
Mattes & Thumbs: 9K
Cast & Score: 3341K
Screen Buffer: 194K
4.0
Credits
Purge Memory
OK

Total Memory (About Director command)

Shows you the total memory available to Director to run your movie. This number depends on the amount of RAM installed on your computer, and any virtual memory that's available.

Total Used (About Director command)

Indicates how much RAM is being used for a movie.

File Size on Disk (About Director command)

Shows how much disk space the current movie occupies. This number is updated every time you save the file. If you have created a new movie and have not saved it yet, this item is not displayed.

Other Memory (About Director command)

Indicates the amount of memory taken up by Windows, by DIRECTOR.EXE, and by other applications.

Physical Memory (About Director command)

Indicates how much physical RAM is currently available to Director. (Physical RAM excludes virtual memory and swap files.)

Memory Limit (About Director command)

Indicates the minimum amount of memory you need to run the current movie as a projector. This information appears only if you've selected the Limit Memory Size checkbox in the Preferences dialog box.

Free Memory (About Director command)

Indicates how much more memory is currently available to Director.

Used by Program (About Director command)

Indicates the amount of memory used by Director (excluding the amount of memory taken up by DIRECTOR.EXE).

Mattes & Thumbs (About Director command)

Shows how much memory is used by cast members that use the Matte ink in the score window and by thumbnail images in the cast window.

Cast & Score (About Director command)

Indicates the amount of memory used by the cast members in the cast window and the notation in the score window. Cast members include all the artwork in the paint window, all the text in the text windows, and any sounds, palettes, buttons, digital video movies, or linked files imported into the cast and currently loaded into memory.

Screen Buffer (About Director command)

Shows how much memory Director reserves for a "working area" while animating on the stage.

Credits button (About Director command)

Click Credits to display the names of the team that brought you Director.

Purge Memory button (About Director command)

Click Purge Memory to remove all purgeable items from RAM, including all thumbnail images in the cast window. All cast members that have a purge priority greater than zero (as specified in the Cast Member Info dialog box) are removed from memory. (Edited cast members don't get purged.) This is useful for gaining as much free memory as possible before importing a large file.

Keyboard shortcuts

Click a category for more information:

File menu shortcuts

Command	Shortcut
New	Control-N
Open	Control-O
Close Window	Control-W
All Windows	Control-Alt-W (or Alt-Close Window)
Save	Control-S
Save and Compact	Control-Alt-S
Save As	Control-Shift-S
Import	Control-J
Export	Control-Shift-E
Movie Info	Control-U
Preferences	Control-Alt-U
Page Setup	No shortcut
Print	Control-Alt-P
Quit	Control-Q

Edit menu shortcuts

Command	Shortcut
Undo	Control-Z
Cut	Control-X
Copy	Control-C
Paste	Control-V
Clear	Backspace
Select All	Control-A
Play	Control-P or E keypad plus sign; keypad Enter toggles between Play and Stop
Stop	Control-period; keypad 2, 5, or decimal point; keypad Enter toggles between Play and Stop
Rewind	Control-R or keypad 0
Step Backward	Control-left arrow or keypad 1 or 4
Step Forward	Control-right arrow or keypad 3 or 6
Disable Sounds	Control-~ or Control-` (grave accent) or keypad 7
Loop	Control-L or keypad 8
Selected Frames Only	Control-\ or Control-| (pipe, not the letter I)

Window menu shortcuts

Command	Shortcut
Stage	Control-1
Control Panel	Control-2
Cast	Control-3 or Control-keypad down arrow
Score	Control-4
Paint	Control-5
Text	Control-6
Tools	Control-7
Color Palettes	Control-8 or Control-keypad up arrow
Digital Video	Control-9
Script	Control-0
Message	Control-M
Tweak	Control-Shift-T
Markers	Control-Shift-M

Cast menu shortcuts

Command	Shortcut
Cast Member Info	Control-I
Open Script	Control-' (apostrophe)
Edit Cast Member	Double-click or Control-down arrow on cast member
Duplicate Cast Member	Control-D
Find Cast Member	Control-; (semicolon)

Score menu shortcuts

Command	Shortcut
Sprite Info	Control-K
Set Blend Value	Control-Alt-B
Delete Sprites	Control-Delete
Paste Relative	Control-Shift-V
Insert Frame	Control-]
Delete Frame	Control-[
In-Between Linear	Control-B
In-Between Special	Control-Shift-B
Switch Cast Members	Control-E

Text menu shortcuts

Command	Shortcut
Find/Change	Control-F
Find Again	Control-G
Find Selection	Control-H
Change Again	Control-T
Find Handler	Control-: (colon)
Comment	Control-> (greater than)
Uncomment	Control-< (less than)
Recompile Script	Control-Shift-R
Recompile All Scripts	Control-Alt-R

Paint menu shortcuts

Command	Shortcut
Show/Hide Tools	Control-Shift-J
Show/Hide Rulers	Control-Shift-K
Zoom In	Control-+ (and Control-=)
Zoom Out	Control-- (and Control-_)

Effects menu shortcuts

Command	Shortcut
Repeat Effect	Control-Y

Help menu shortcuts

Command	Shortcut
Help Contents	Alt-F1

Stage contents shortcuts

When there is a sprite selected on the stage, the arrow keys nudge the sprite. Exceptions (i.e., windows that grab the arrow keys first) are within a text selection in the paint, text, markers, and script windows.

Action	Shortcut
Open cast member editor	Double-click sprite
Inks pop-up	Control-click sprite
Toggle step record light on and off	Alt-click sprite
Real-time record	Control-spacebar drag cast member
Display cast member info	Click sprite with right mouse button
Display sprite script	Alt-click sprite with right mouse button
Change contents of stage to black	Keypad minus sign
Invert everything on stage	Keypad /
Previous marker comment	Keypad: Alt-1 or Alt-4
Next marker comment	Keypad: Alt-3 or Alt-6

Score window shortcuts

Action	Shortcut
Duplicate selection of cells	Alt-drag (if Drag and Drop is selected in Preferences dialog box); Alt-Spacebar-drag (if Drag and Drop is not selected)
Open cast editor for selected sprite	Double-click cast thumbnail
Select entire range of a cast member	Double-click cell with a sprite in it
Select channel	Double-click channel number
Select multiple channels	Double-click channel number and drag up or down
Toggle record light	Alt-click channel number
Move playback head to end of movie	Tab
Move playback head to first frame	Shift-Tab
Playback head to beg/end of selection	Control-Shift-left/right arrow
Open marker window	Double-click any marker
Open appropriate Set dialog box	Double-click cell in tempo, palette, or transition channel
Open Set Sound dialog box	Select one or more cells in a sound channel and then Control-double-click in the selected sound cells
Go to next marker comment (or jump 10 frames)	Control-Alt-right arrow
Previous marker comment (or back 10 frames)	Control-Alt-left arrow
Shuffle Backward	Control-Shift-left arrow
Shuffle Forward	Control-Shift-right arrow
Temporarily turn off/on drag and drop	Hold down Spacebar

Cast window & Cast Editor window shortcuts

Action	Shortcut
Open cast member editor	Double-click a paint, text, palette or script cast member or Control-Shift-down arrow
Display cast member info	Control-click cast thumbnail
Display cast member script	Alt-click cast thumbnail with the right mouse button
Open script in new window	Alt-click Script button, Alt-Open Script command, or Control-Alt-' (apostrophe)
Open new text window	Alt-Add button in text window
Open new script window	Alt-Add button in script window
Open new digital video window	Alt-Add button in digital video window
Place sprite in center of stage	Control-Shift-L (places in center of stage)
Center many cast members over time	Control-Alt-L
Add new cast member	Control-Shift-A
Select or display previous cast member	Control-Shift-left arrow
Select or display next cast member	Control-Shift-right arrow
Scroll up/down one window	Page Up, Page Down
Scroll to top left of cast window	Home
Scroll to show last occupied cast member	End
Scroll to a cast member	Type a cast number (scrolls after a brief pause; press Enter to scroll immediately). Backspace key deletes last digit typed
Scroll to a selected cast member	Double click cast number at top of window

Paint window shortcuts

(**Action**: Shortcut)

Undo: ~

Next/previous cast member: Keypad left/right arrow keys (Num Lock must be off)

Turn selected tool into foreground eyedropper: Hold down right mouse button (release to select color)

Turn selected tool into background eyedropper: Hold down Alt and right mouse button (release to select color)

Turn selected tool into destination eyedropper: Control-Option key, while pressed

Turn selected tool into hand tool: Spacebar, while pressed

Nudge selection rect. or lasso selection: Keypad arrows with selection rectangle or lasso

Change airbrush size: Keypad up/down arrows with air brush selected

Change airbrush flow: Keypad left/right arrows with air brush selected

Change foreground color: Keypad up/down arrows, with any tool other than the air brush selected

Change background color: Shift-keypad up/down arrows, with any tool other than the air brush selected

Change destination color: Alt-keypad up/down arrows, with any tool other than the air brush selected

Draw border w/current pattern: Alt-shape or line tools

Select background color: Shift-eyedropper

Select destination color: Alt-eyedropper

Toggle between custom and grayscale patterns: Alt-click pattern

Polygon lasso: Alt-lasso

Duplicate selection: Alt-drag

Stretch: Control-drag

Draw with background color: Alt-pencil tool

Open cast window & select cast member: Double-click cast number at top of window

Open Gradients dialog box and set ink to gradient: Double-click ellipse, paintbrush, rectangle, paint bucket, or polygon tool

Open Air Brushes dialog box: Double-click air brush

Clear visible part of window: Double-click eraser

Open color palettes window: Double-click foreground, background, or destination color chip

Open Patterns dialog box: Double-click pattern chip

Open Brush Shapes dialog box: Double-click paintbrush

Open Paint Window Options: Double-click line width selector

Open Transform Bitmap dialog box: Double-click color resolution indicator

Toggle Zoom in/Zoom out: Control-click in window or double-click pencil tool

Color palettes window shortcuts

Action	Shortcut
Open Set Color dialog box	Double-click a color

Markers window shortcuts

Action	Shortcut
Move cursor to next / previous comment	Control-Alt-left/right arrow

Message window shortcuts

Action	Shortcut
Go to top/bottom of window	Control-Up/down arrow
Delete contents of window	Control-Shift-delete