MX

macromedia®

DIRECTOR® MX 2004

Using Director

Second Edition: September 2004

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

CONTENTS

CHAPTER 1
Introduction

Welcome to Macromedia Director MX 2004. With Director, you can develop high-performance multimedia content and applications for CDs, DVDs, kiosks, and the Internet. This guide, *Using Director*, includes comprehensive descriptions and detailed step-by-step instructions that explain how to use all of the features in Director.

In this chapter, you can get an overview of Director, learn about the features that are new in this release of Director, and find additional sources of information.

About Director

With Director, a proven multimedia authoring tool for professionals, you can develop high-performance content and deploy it anywhere: on CDs, DVDs, intranets, kiosks, or the Internet. Whether you are creating enhanced CD/DVD-ROM content, educational content, or entertainment applications, Director handles the widest variety of media, letting you unleash your creativity and build rich, interactive experiences that deliver measurable results. Director provides all the tools you need to maximize productivity, including a choice of scripting languages and cross-platform publishing.

With Director, you can now do all of the following:

- Embed most major media formats in your multimedia projects, including DVD-Video, Windows Media, RealMedia, QuickTime, and Macromedia Flash content, in addition to audio, bitmap, and 3D formats.
- Work seamlessly with other Macromedia MX products, including Macromedia Flash MX 2004.
- Extend the authoring environment and playback engine with the Xtra extension plug-in architecture in Director. Use third-party Xtra extensions or write your own to control external devices, such as joysticks and cameras, and perform sophisticated operating system-level tasks.
- Write scripts to add interactivity and automation to your projects, using the Lingo scripting language, JavaScript syntax, or a combination of the two.
- Publish content across platforms and to different formats, including Macintosh and Windows projectors.

Users view your completed Director movies in one of the following ways:

- In a projector, which plays on your user's computer as a stand-alone application.
- In the Macromedia Shockwave Player format, which plays in Shockwave-enabled web browsers. Millions of web users already have the Shockwave Player on their computers, browsers, or system software. Others have downloaded Shockwave Player, which is free, from the Macromedia website at www.macromedia.com/shockwave/download/.

Tip: To see some of the exciting and varied ways in which developers use Director, visit the Director showcase at www.macromedia.com/go/discreet_inspiration. You can also see great examples of Shockwave content at www.shockwave.com.

What's new in Director MX 2004

Director MX 2004 introduces many new features, designed to save time during authoring or to improve integration with other software, media types, and multimedia development processes.

Importing Macromedia Flash MX 2004 content You can now access Macromedia Flash MX 2004 from within Director.

Scripting in JavaScript syntax Director now supports scripting in JavaScript syntax, in addition to Lingo. You can use whichever language is more familiar to you or use both languages in the same movie to develop interactive features and functionality.

Using prebuilt components Prebuilt Macromedia Flash MX 2004 components are now available to reduce your scripting time. You can drop components, such as calendars and user interface elements, into movies to cut the time that it takes to develop interactive features.

Adding DVD-Video You can now embed, control, and play back the popular DVD-Video format inside Director movies. This feature makes DVD possible and affordable for a wide variety of developers—including entertainment studios, distance learning professionals, DVD authors, and corporate presentation specialists.

Publishing to Mac and Windows in one step After you complete a movie, you can now publish across platforms in one step, creating either stand-alone applications or web-based Shockwave content that runs on Mac and Windows. The new projector publishing feature eliminates extraneous dialog boxes, saves projector settings on a per-project basis, and eliminates redundancies.

Naming sprites and channels Sprites and channels can now have custom names, and absolute references to sprites are no longer necessary. This feature is an advantage when you make last-minute changes to your movie. With absolute references, your Lingo or JavaScript syntax scripts might break. But with sprite names, you can move sprites on the Score without worrying about broken scripts.

Customizing your workspace You can arrange your workspace in multiple configurations and save each workspace for later use. As a new part of this feature, you can now create workspace settings that include Movie in a Window (MIAW) windows that are docked alongside your other windows. These docked MIAW windows can extend Director functionality.

Integrating with other Macromedia Studio MX 2004 products If you already use other Macromedia Studio MX 2004 products, such as Macromedia Flash MX 2004 and Macromedia Fireworks MX 2004, then the Director interface is already familiar to you. In addition to a common interface, you can now start and edit other Macromedia Studio MX 2004 files from directly within Director.

Integrating with Macromedia server technologies You can now choose to integrate Director with Macromedia server technologies, such as Macromedia ColdFusion MX 6.1 and Macromedia Flash Communication Server MX. For multi-user games, distance learning content, and other server-controlled content, this link between your Director content and IT infrastructure extends your interactivity options.

Getting help from the reference panel A new reference panel is now available in Director to simplify getting help with using Director, behaviors, Xtra extensions, and application programming interfaces (APIs) for both Lingo and JavaScript syntax.

Guide to instructional media

Director contains a variety of information sources to help you learn the program quickly and become proficient in creating multimedia. This information includes several printable PDF files and online help. The Director workspace contains tooltips and context-sensitive help, and additional help is available on the Director website at www.macromedia.com/go/director_support.

Getting online help

As you use Director, you can get immediate online help by opening Director Help. You can also get help with the specific item in Director that you are currently using. This feature is called context-sensitive help.

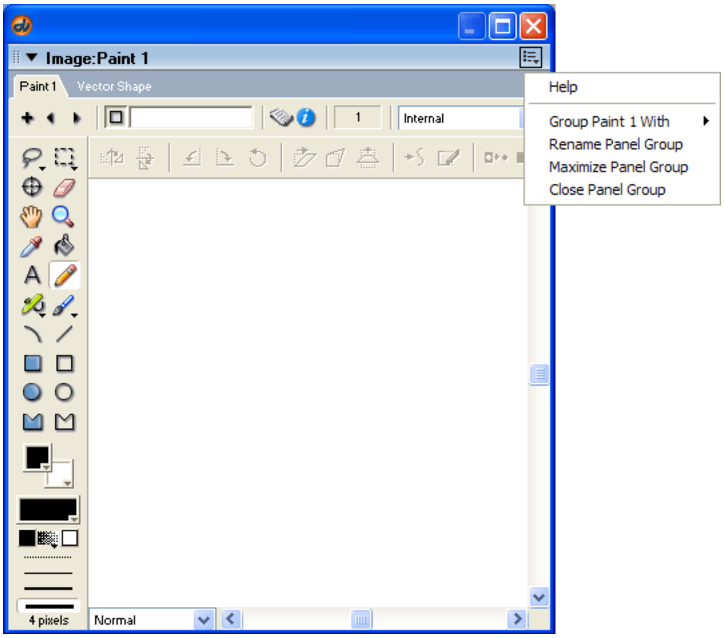
To access Director Help:

1 Select Help > Director Help.

2 Browse for a topic on the Contents tab, or type a keyword on the Search tab.

To access context-sensitive help, do one of the following:

- If you are working in a window, display the Options menu for that window and select Help.



- If you are working in a dialog box, click the Help button.

  Director Help opens and displays a topic that relates to the window or dialog box that you are using.

Sources of information

Getting Started with Director This printed manual contains the essential information that you need to get started, including information about installing the latest version of Director. This manual also guides you through the workspace and offers a tutorial, designed for those who are new to Director.

Director Help Panel This online help system is the comprehensive information source for all Director features. It includes overviews of the features, examples, how-to procedures, descriptions of interface elements, and a reference of all scripting objects in both Lingo and JavaScript syntax. Topics are linked and indexed to make finding information and jumping to related topics quick and easy. To view the Director Help Panel, select Help > Director Help while you are working in Director.

Using Director This manual is available in printable PDF format from the Director Documentation Center at www.macromedia.com/go/director_docs. It explains how to use all of the features and functionality offered in Director. Information in the manual is also available in the online Director Help.

Director Scripting Reference This manual is available in printable PDF format from the Director Documentation Center at www.macromedia.com/go/director_docs. It provides a complete reference of the Director application programming interfaces (APIs), including both Lingo and JavaScript syntax. Information in the manual is also available in the online Director Help.

Creating Your First 3D Movie in Director This tutorial is available in printable PDF format from the 3D tutorial page on the Macromedia website at www.macromedia.com/go/drmx2004_3d_tutorial_en. It takes you step by step through creating a simple 3D movie.

Tooltips When you place your mouse pointer over a Director tool or another item in the Director workspace for a few seconds, a small tooltip appears that explains what you can do with the item.

Director Support Center The Director Support Center website (www.macromedia.com/go/director_support) contains the latest information about Director, plus additional topics, examples, tips, and updates. Check the website often for the latest news and how to get the most out of Director.

Document conventions

Director Help and the Director product manuals all follow a few basic conventions:

- The term *Director* typically refers to the most recent version of Director.
- *Lingo* refers to a scripting language that is shipped with Director. *JavaScript syntax* refers to the Director implementation of JavaScript.
- How-to procedures are identified with bold headings that begin with "To..." and end with a colon. For example, if **To access Director Help:** appears in a heading, then the heading is followed by a set of steps.
- Examples of Lingo and JavaScript syntax are shown in a fixed-width font. For example, `answer = 2 + 2` is a sample Lingo statement.
- Variables used to represent parameters in Lingo and JavaScript syntax appear in italics. For example, *`whichCastMember`* is commonly used to indicate where you should insert the name of a cast member in Lingo or JavaScript syntax.
- Text that you should type in a window or dialog box is shown in a **bold** font.

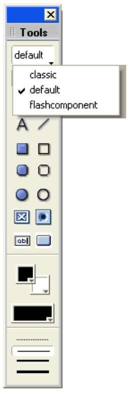
CHAPTER 2
Score, Stage, and Cast

If you are new to Macromedia Director MX 2004, see the Getting Started topics in the Director Help Panel to become familiar with the Director workspace and features. (In Macromedia Director MX 2004, select Help > Director Help to view the Getting Started topics.) These topics include overviews and basic definitions of the primary windows and tools in Director, some of the basic tasks that you need to know how to do, and a tutorial that guides you through the process of creating a simple movie.

The topics presented here go into greater detail about how to use three of the most important windows in Director: the Score, the Stage, and the Cast windows.

Using the Tool palette with the Stage

The Tool palette contains useful tools for creating and manipulating sprites on the Stage. It also contains drawing tools and elements you can place directly on the Stage. There are three views that you can access: Default, Classic, and Flashcomponent. Each view of the Tool palette contains some of the same tools; for example, they each contain the arrow, hand, and magnifying glass tools. The Flashcomponent view contains primarily Flash components, while the Classic view has no components at all. The Default view combines elements from the other two: some Flash components but also some Classic items.



To change Tool palette views:

1 If the Tool palette is not already available, select Window > Tool Palette.

  The default Tool palette appears.

2 Click on the Tool palette view menu and select the view you want: classic, flashcomponent, or default.

The Tool palette changes to display the tools available for each view.

In Windows, you can also dock the Tool palette to the docking channels contained by the application window. (There are no docking channels on the Macintosh.)

- To dock the Tool palette (Windows only), click and drag the palette by the palette gripper over a docking channel. A placement preview line or rectangle appears when the palette can be docked.

How to use each tool in the Tool palette is covered in topics that relate to that specific tool. For example, to find out how to use the Flash component tools, see "Selecting components using the Tool palette" on page 208.

Setting Stage properties

When active, the Stage has three tabs in the Property inspector available at all times: Guides, Movie, and Display Template.

- The Display Template tab lets you set properties for Movies in a Window. For more information, see Chapter 18, "Movies in a Window," on page 409.
- The Movie tab lets you set the properties of the movie on the Stage. These properties include color definitions, size, and location of the Stage while the movie plays and channels in the movie's Score. For more information about setting movie properties, see "Setting movie properties" in the Getting Started topics in the Director Help Panel. (In Director, select Help > Director Help to view the Getting Started topics.)
- The Guides tab lets you control the guides and grid that appear on the Stage to assist with movie authoring. The bottom half of the Guides tab contains grid settings.

Setting guides and grid properties

Guides are horizontal or vertical lines that you can either drag around the Stage or lock in place to assist you with sprite placement. The Guides tab also lets you activate the grid. The grid contains cell rows and columns of a specified height and width that you use to assist in visually placing sprites on the Stage. Moving a sprite with the Snap to Grid or Snap to Guides feature selected lets you snap the sprite's edges and registration point to the nearest grid or guide line. (For more information about sprites, see Chapter 3, "Sprites," on page 51.)

You must create guides before they become available; you do this by using the Guides tab on the Property inspector. The grid is always available. Guides and the grid are visible during authoring only. When you are not using the guides or the grid, you can hide them.

To create and set guide properties:

1 With the Property inspector open, click the Guides tab.

  The top half of the tab contains settings for Guides.

2 Click the Guide Color box to select a different color.

3 Select the options you want to make the guides visible, lock them, or to make the sprites snap to the guides.

4 To add a guide, move the cursor over the new horizontal or vertical guide, and then drag the guide to the Stage. Numbers in the guide tooltip indicate the distance, in pixels, the guide is located from the top or left edge of the Stage.

5 To reposition a guide, move the pointer over the guide. When the sizing handle appears, drag the guide to its new position.

6 To remove a guide, drag it off the Stage.

7 To remove all guides, click Remove All on the Guides tab in the Property inspector.

To set grid properties:

1 With the Property inspector open, click the Guides tab.

  The bottom half of the Guides tab contains grid settings.

2 To change the grid color, click the Grid Color box and select a different color.

3 Select the desired options to make the grid visible and to make the sprites snap to the grid.

4 To change the width and height of the grid, enter values in the W and H text boxes.

5 Select the desired options to display the grid as dots or lines.

Using multiple Score windows

You can view and work in different parts of a movie at the same time by opening additional Score windows. If your sprite spans occupy many frames in the Score, for example, you can open a second Score window to work on another place in the movie without scrolling. You can also drag sprites from one Score window to another.

To open a new Score window:

1 Activate the current Score window.

2 Select Window > New Score Window.

You can scroll in this window to a different location in the Score.

Changing Score settings

To control the appearance of the Score and the information that appears in numbered sprite channels, you set preferences for the Score. By doing so, you can display a script preview and cast member information.

To change Score settings:

1 Select Edit > Preferences > Score.

Note: If you are using a Macintosh OS X operating system, select the Director menu, instead of the Edit menu, to access Preferences.

2 The Extended display option lets you display information about sprites in the Score. For more information, see "Displaying sprite labels in the Score" on page 62. To specify what information appears in the numbered sprite channels when Extended display is on, select from the following options:

Name displays sprites by name, if they have names assigned to them.

Cast Member displays the cast member number, name, or both.

Behaviors displays the behaviors attached to the sprite.

Ink Mode displays the type of ink applied to the sprite.

Blend displays the blend percentage applied to the sprite.

Location shows the sprite's *x* and *y* screen coordinates.

Change in Location shows the change in *x* and *y* coordinates relative to the previous cast member in that channel.

3 To display the first few lines of the selected script in a box at the top of the Score, select Script Preview.

4 To display the cast member's name and number when the pointer is over a sprite for a few seconds, select Show Data Tips.

You can also change when a sprite span starts in the score (different frames or a marker, perhaps) by setting it through Edit > Preferences > Sprite. For more information, see "Sprites" on page 51.

Note: If you are using a Macintosh OS X operating system, select the Director menu, instead of the Edit menu, to access Preferences.

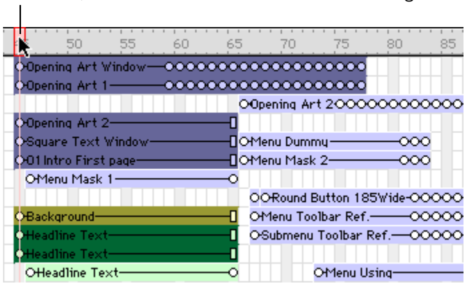
Selecting and editing frames in the Score

You can select a range of frames in the Score and then copy, delete, or paste all the contents of the selected frames.

To move, copy, or delete all the contents of a range of frames:

1 Double-click in the frame channel to select frames.

Double-click here to select all sprites in a frame, including markers, special effects, and sounds. Double-click and drag to select a range of frames.



2 If you want to move or copy frames, select Edit > Cut Frames or Edit > Copy Frames.

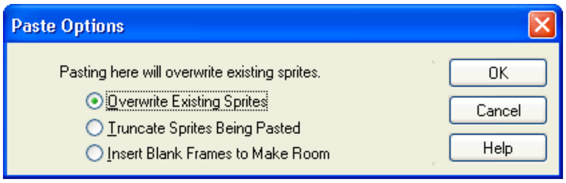
3 If you want to delete frames, select Edit > Clear Frames, or press the Delete key on your keyboard.

If you cut, clear, or delete the selected frames, Director removes the frames and closes up the empty space.

Note: To delete a single frame, you can also select Insert › Remove Frame.

4 To paste frames that you have cut or copied, select any frame or sprite, and select Edit > Paste Sprites.

If there aren't enough empty frames available for the entire sprite to be pasted, the Paste Options dialog box appears, so you can decide how you want the frames to be pasted.


Paste Options

Pasting here will overwrite existing sprites.

[✓ Overwrite Existing Sprites]
Truncate Sprites Being Pasted
Insert Blank Frames to Make Room

Overwrite Existing Sprites copies the entire sprite over the frames of any existing sprites

Truncate Sprites Being Pasted pastes the sprite into the number of available empty frames by shortening its frame span.

Insert Blank Frames to Make Room inserts frames into the Score so the entire sprite can fit without being truncated or overwriting other sprites.

To add new frames:

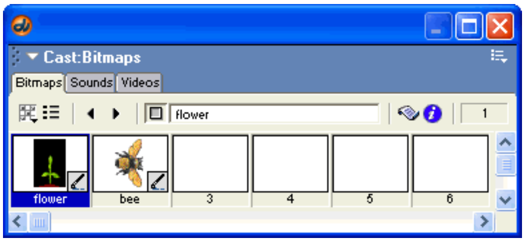
1 Select a frame in the Score.
2 Select Insert > Frames.
3 Enter the number of frames to insert.

The new frames appear to the right of the selected frame. Sprites in the frames you select are extended or tweened. For more information about tweening, see Chapter 4, “Animation,” on page 83.

About Cast Members

Cast members are the media and other assets in your movie. They can be bitmaps, vector shapes, text, scripts, sounds, Macromedia Flash content or components, DVD content, QuickTime movies, Windows Media video or audio, Macromedia Shockwave 3D content, Rich Text, sounds of various formats, and more. When you place a cast member on the Stage or in the Score, you create a sprite. For more information about sprites, see Chapter 3, “Sprites,” on page 51.

You use windows called casts to group and organize your cast members. To populate casts, you import and create cast members. You can create and use multiple casts in a movie. You can also group multiple Cast windows together in a tabbed panel layout:

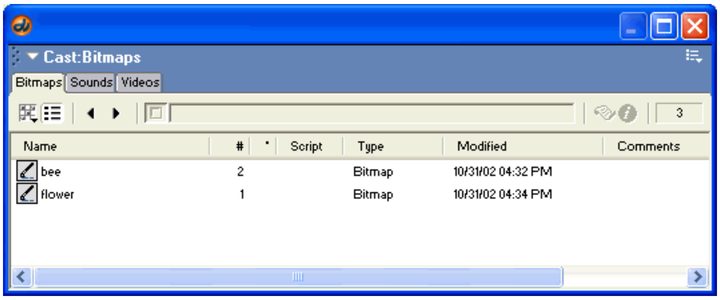


Cast window with tabs in Thumbnail view

You can create and edit cast members in Director by using basic tools and media editors such as the Paint and Text windows, or you can edit cast members by using external editors. In addition, you can import cast members from nearly every popular media format into a movie file. For some media types, you can link cast members as external files located on a disk or the Internet. Linked cast members can be updated dynamically.

The Property inspector contains asset management fields for cast members on the Member tab. These fields let you name cast members, add comments about them, and view information such as creation and modification dates and file size.

Casts can be internal—stored inside the movie file and exclusive to that movie—or external—stored outside the movie file and available for sharing with other movies. When you create a new movie, an empty internal cast is created automatically, and when you open the Cast window it is in the default List view. For more information about Cast window views, see "Switching from one Cast window view to another" on page 26.



Name	#	*	Script	Type	Modified	Comments
bee	2			Bitmap	10/31/02 04:32 PM	
flower	1			Bitmap	10/31/02 04:34 PM	

Cast window in List view

External casts are also useful for creating groups of commonly used cast members. You can use external casts as a way of switching large groups of cast members in a single step. For example, you could switch the text cast members in your movie from English to French by simply switching the cast that the movie uses, rather than each individual cast member.

Using external casts can keep the movie size small for downloading because an external cast can download separately from the movie file if or when it is needed.

Creating new casts

Before assembling a large number of cast members, it's good practice to create the casts that are necessary to keep them organized. You can sort casts by type, edit cast properties, and use external casts for storing and sharing common media elements.

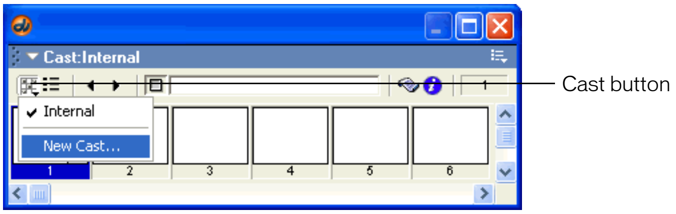
You can create as many casts as necessary; the number of casts does not affect the size of a movie for downloading.

You can include as many as 32,000 cast members in a single cast, but it's usually best to group media such as text, buttons, and images logically in a few different casts for each movie.

To create a new cast:

1 Do one of the following:

- Select File > New > Cast.
- Select Modify > Movie > Casts to open the Movie Casts dialog box, and click the New button.
- In the Cast window, click the Cast button and select New Cast from the pop-up menu. (See "Using Cast window controls" on page 29.)



2 In the New Cast dialog box, type a name for the new cast.

3 Specify how to store the cast:

Internal stores the cast within the movie file. This option makes the cast available only to the current movie.

External stores the cast in a separate file outside the movie file. This option makes the cast available for sharing with other movies. For information about internal and external casts, see "Managing external casts" on page 47.

4 If you chose External and you don't want to use the cast in the current movie, deselect the Use in Current Movie option.

Note: You can link the external cast to your movie later. See "Managing external casts" on page 47.

5 Click Create.

The cast is created and appears as a tabbed panel in the Cast window.

Note: Creating a new cast by selecting Modify > Movie > Casts does not automatically display a tabbed panel. To display the tabbed panel, click the Cast button and select the cast you created from the pop-up menu.

6 If you created an external cast, select File > Save while its Cast window is active, and then save the cast in the desired directory.

Creating cast members

You can create several types of cast members in Director. Director includes editors to create and edit common media such as video, text, shapes, and bitmaps. You can also define external editors to launch from within Director when you double-click a cast member, and you can edit almost any type of supported media. For more information, see "Launching external editors" on page 46.

You can also import cast members. For more information, see "Importing cast members" on page 42.

To create a new cast member from the Insert menu:

1 Open the Cast window for the cast member you are creating.

  To place a cast member in a specific position in the Cast window, select the position in Thumbnail view. For more information, see "Using Cast Thumbnail view" on page 33. Otherwise, Director places the new cast member in the first empty position or after the current selection in the Cast window.

2 Select Insert > Media Element, and then select the type of cast member to create.

  For more information about each choice, see the following sections:

  - "Using the Paint window" on page 102
  - "Using the Color Palettes window" on page 151
  - "Streaming linked Shockwave Audio and MP3 audio files" on page 238
  - "Creating text cast members" on page 165
  - "Embedding fonts in movies" on page 164
  - "Creating an animated color cursor cast member" on page 299
  - "Drawing vector shapes" on page 135
  - "Using Flash Content" on page 181
  - "Using Windows Media files in Director" on page 252
  - "Using DVD media content in Director" on page 254
  - "About digital video formats" on page 244
  - "Using animated GIFs" on page 101

3 To create a control or button, do one of the following:

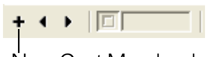
  - Select Insert > Control > Push Button, Radio Button, or Check Box to create a button cast member and a sprite on the Stage. For more information, see "Using shapes" on page 143.
  - Select Insert > Control > Field to create a field cast member. Creating a field cast member also creates a sprite on the Stage. For more information, see "Working with fields" on page 172.
  - (Windows only) Select Insert > Control >ActiveX to create an ActiveX cast member. For more information, see "Using ActiveX controls" on page 204.

Note: You can also use Flash components to create controls and buttons. For more information about using Flash components, see "Selecting Flash components" on page 207.

To create a cast member in a media editing window:

1 Open a media editing window by selecting Window and then selecting the type of cast member you want to create (Paint, Vector Shape, Text, Windows Media, DVD, and so on).

2 Click the New Cast Member button to create a cast member of the corresponding type. The cast member is added to the most recently active Cast window.



To create a cast member using the Script window:

1 Open the Script window by selecting Window > Script.

2 Click the New Cast Member button to create a script cast member.

To create a cast member on the Stage:

1 Open the Tool palette, if it is not already open, by selecting Window > Tool Palette.

2 Using the tools in the Tool palette, create content directly on the Stage. Each object you create automatically becomes a cast member.

Note: Cast members created on the Stage are automatically placed in the Score.

Using the Cast window

In the Cast window, you can view the cast in either the default List view or the Thumbnail view. (You can change the default so that the Cast window opens in Thumbnail view. See "Setting Cast window preferences" on page 36.)

The Cast window lets you do the following actions:

- Organize and display all media in a movie.
- Move groups of cast members.
- Start editors for cast members.
- Launch the Property inspector to view, add, and change comments about your cast members and to view and modify cast member properties.
- Group multiple casts in a tabbed view using panel groups (see "Working with Cast panel groups" on page 27).

To view the Cast window:

- Select Window > Cast or press Control+3 (Windows) or Command+3 (Macintosh).

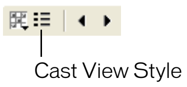
  If there is more than one cast in the movie, you can select which Cast window to open by selecting Window > Cast and then selecting a cast name from the Cast submenu.

Switching from one Cast window view to another

You can easily toggle between List and Thumbnail views of the Cast window.

To switch from one Cast window view to another, do one of the following:

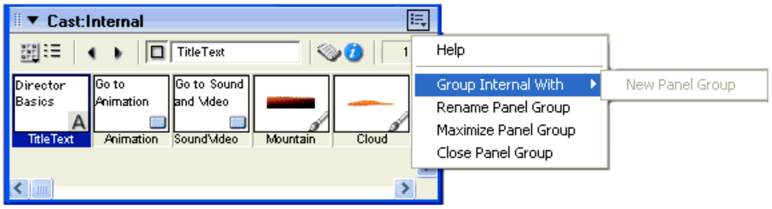
- Click the Cast View Style button on the Cast window to toggle between the two views.



- With the Cast window active, select View > Cast, and select either List or Thumbnail, as desired.
- Right-click (Windows) or Control-click (Macintosh) the Cast window, and select either List or Thumbnail, as desired, from the context menu.

Working with Cast panel groups

Each Cast panel and panel group has an Options menu located in its upper right corner. The Options menu contains items for grouping, closing, and renaming panels.



To use a Cast panel Options menu:

- Click the Options menu control in the upper right of the panel, and select the desired menu item.

  Help launches the page in the help system that is relevant to the current panel.

  Group [Panel name] With lets you group the currently selected tab in a panel group with another Cast panel or panel group.

  Rename Panel Group opens the Rename Panel Group dialog box where you can rename the Cast panel group.

  Maximize Panel Group maximizes the panel group to occupy the entire height of the docking channel.

  Close Panel Group closes the panel group.

To group a Cast panel with another Cast panel or Cast panel group:

1 Select a Cast panel or a tab within a Cast panel group.

2 From the panel's Options menu, select Group [Panel Name] With, then select a panel or panel group name from the submenu that appears.

To remove a panel (tab) from a Cast panel group:

1 Select a tab within a Cast panel group.

2 From the panel group's Options menu, select Group [Panel Name] With, then select New Panel Group from the submenu that appears.

The selected panel opens in its own floating panel. The new panel assumes the name previously assigned to it.

Note: The New Panel Group submenu option is dimmed if the panel group only contains a single panel.

To rename a Cast panel group:

1 Select Rename Panel Group from the panel's Options menu.

2 In the Rename Panel Group dialog box, type a new name for the panel group in the Panel Group Name text box, and click OK.

To rearrange the order of tabs within a Cast panel group:

1 Select a tab within the Cast panel group.

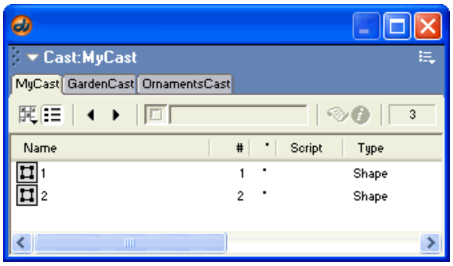
2 Select Group [Panel Name] With from the panel group's Options menu, then choose the name of the Cast panel group that contains the selected panel.

The tab is moved to the last (rightmost) position in the panel group.

Note: Save your panel layout if you want to restore your Cast panel configuration the next time you open your file. To save your panel layout, select Window › Panel Sets › Save Panel Layout.

Managing Casts

When casts are grouped with other casts, they appear as tabs in the Cast panel group.



Name	#	*	Script	Type
1	1	*		Shape
2	2	*		Shape

To save the configuration of your Cast panel tabs, you must save the panel layout before closing your file. When you open the file again, restore the tab configuration by opening the panel layout you created. For more information, select Help > Director Help and see "Saving panel sets" in the Getting Started topics.

Managing casts in older Director movies

When you open a multicast movie created in a previous version of Director, only the first cast appears in the Cast window. You can display the remaining casts as tabs in a panel group, or in a new Cast window.

To open a cast as a tabbed panel:

- In the Cast window, click the Cast button and select the cast from the pop-up menu.

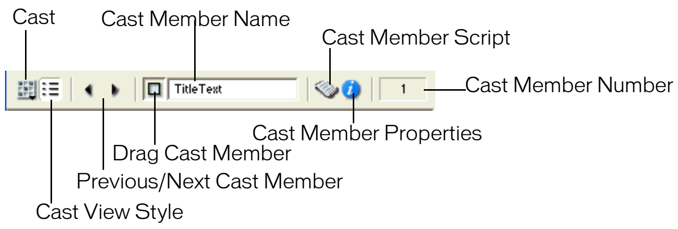
To open a cast in a new window:

- Alt+click (Windows) or Option+click (Macintosh) the Cast button and select a cast from the pop-up menu.

A dialog box reminds you to save your panel layout if you want to restore the configuration of your Cast windows the next time you open your file.

Using Cast window controls

The controls along the top of the Cast window are the same in both the List and Thumbnail views. You use the controls to change the cast that appears in the Cast window, the cast member selection, or the name of a cast member. You can also use them to move cast members and to open a cast member's Script window or the Property inspector.



To change the cast displayed in the current Cast window, do one of the following:

- Click the Cast button and select a cast from the pop-up menu.

  The cast is displayed as a tabbed panel in the current panel group.
- Click a tabbed panel to make it active.
- Press Control+Alt (Windows) or Command+Option (Macintosh) followed by the Right Arrow key or Left Arrow key to move from tab to tab.

To open a cast in a new Cast window:

- Alt+click (Windows) or Option+click (Macintosh) the Cast button and select a cast from the context menu.

  A dialog box reminds you to save your panel layout if you want to restore your Cast panel configuration the next time you open your file.

To select the previous or next cast member:

- Click the Previous Cast Member or Next Cast Member button.

To move a selected cast member to a new position in the Cast window (Thumbnail view) or to the Stage:

- Drag the Drag Cast Member button to the desired position in the Cast window or to the Stage.

  This procedure is useful when the selected cast member has scrolled out of view.

To enter a cast member name:

- Select a cast member, and enter the name in the Cast Member Name text box.

To edit a cast member script:

- Select a cast member and click the Cast Member Script button.

To view cast member properties:

1 Select a cast member.

2 Do one of the following:

- Click the Cast Member Properties button.
- Right-click (Windows) or Control-click (Macintosh), and select Cast Member Properties from the context menu.
- Select Window > Property Inspector. The Property inspector displays only those properties associated with the selected cast member.

For more information, see "Viewing and setting cast member properties" on page 39.

To view the cast member number:

- Refer to the Cast Member Number field in the upper right corner of the Cast window.

Selecting cast members in the Cast window

Before changing, sorting, or moving cast members, you must select them in the Cast window.

To select a single cast member, do one of the following:

- In List view, click the name or icon (Windows) or click any part of the text or icon (Macintosh).
- In Thumbnail view, click the thumbnail image.

To select multiple adjacent cast members, do one of the following:

- In List view, Shift-click or marquee-select the cast members.
- In Thumbnail view, click the first cast member in the range and then Shift-click the last cast member in the range.

To select multiple nonadjacent cast members:

- In either List or Thumbnail view, Control-click (Windows) or Command-click (Macintosh) each cast member that you want to select.

Copying cast members

You can easily create multiple versions of a cast member in a single cast. For example, you might want several cast members to be identical except for color or size. You can also copy cast members from one Cast window to another.

To copy a cast member:

1 In either List or Thumbnail view, select the cast member (or multiple cast members) that you want to copy.

2 Alt+click (Windows) or Option+click (Macintosh), and drag the cast member to a new location in Thumbnail view or to the bottom of the list in List view.

   You can drag the cast member to a location in the same Cast window or to a different Cast window. Director creates a cast member with a new number, but with all of its other information identical to the original.

3 If you copied the cast member into the same Cast window, change the name of the copied cast member so that you (and scripts) can distinguish it from the original. For more information, see "Naming cast members" on page 31.

Naming cast members

To avoid problems in Lingo or JavaScript syntax when referring to cast members, you should name them and then refer to them by name. Naming cast members doesn't affect Director performance. The name stays the same even if the cast member number changes.

Avoid duplicating cast member names. If more than one cast member has the same name, Lingo or JavaScript syntax uses the cast member with the lowest number in the cast.

To name a cast member:

1 Select the cast member in either the List or the Thumbnail view of the Cast window.

2 Do one of the following:

- Enter a name in the Cast Member Name text box at the top of the Cast window or in any of the editing windows.
- Enter a name in the Name text box on the Cast or Member tab in the Property inspector.

To name a cast member using Lingo or JavaScript syntax:

- Set the `name` cast member property. For more information about this property, see the Scripting Reference topics in the Director Help Panel.

Using Cast List view

Cast List view, the default view in which the Cast window opens, provides seven columns of information by default. They are shown in the following table:

Column Title	Column Information
Name	The name of the cast member and an icon that describes the cast member type. For information about what the icons represent, see "Using Cast Thumbnail view" on page 33.
#	The number that is assigned to the cast member. This number represents the order in which this cast member appears in Thumbnail view.
*	An asterisk (*) in this column indicates the cast member has changed, but you have not yet saved those changes.
Script	The word Member in this column means the cast member contains a script. The word Movie in this column means the cast member is a movie script. The word Behavior in this column means the cast member is a Behavior. You can use the Script icon to view the script or behavior.
Type	The cast member type
Modified	The date and time the cast member was changed
Comments	Displays text entered on the Property inspector Member tab, in the Comments text box.

Four additional columns are available in the Cast Window Preferences dialog box. For more information, see "Setting Cast window preferences" on page 36. The additional columns that you can display are described in the following table:

Column Title	Column Information
Size	The size in bytes, kilobytes, or megabytes
Created	The date and time the cast member was created
Modified By	Who modified the cast member. This value comes from the user login name (Windows) or the Sharing setup name (Macintosh).
Filename	The full path to the cast member if it is a linked asset

Resizing columns in Cast List view

You can resize columns in Cast List view.

To resize a column:

1 Hold the pointer over the column boundary to activate the Resizing tool.

2 Drag the column to the desired size.

Sorting Cast List view columns

You can sort the Cast List view columns in ascending and descending order by clicking the column title. When you sort the Cast List window by clicking the column title, you're changing the way in which the information appears but are not changing any cast member attributes.

About cast member order in Cast List view

Unlike the way in which cast members appear in Thumbnail view, in List view the cast member order does not always correspond to the member's physical location in the cast.

When you work in List view, also keep in mind the following:

- In List view, Director places new cast members at the end of the list, and the cast member number becomes the first available number after the current selection.
- You can use Thumbnail view to reorder (and renumber) cast members by dragging them to different locations in the window; you cannot reorder cast members by dragging in List view.

Using Cast Thumbnail view

As the name suggests, the Cast Thumbnail view shows a very small (thumbnail) version of the cast member, along with an icon that represents the cast member media type, as shown in the following table:

Icon	Cast member type	Icon	Cast member type
	Animated GIF		Behavior
	Bitmap		Button
	Check box		Custom cursor
	Digital video		DVD
	Field		Film loop
	Flash component		Flash movie
	Font		Linked bitmap (all linked cast member icons are changed in the same way)
	Movie script		OLE
	Palette		Parent script
	PICT		QuickTime video
	Radio button		RealMedia
	Shape		Shockwave 3D
	Shockwave Audio		Sound
	Text		Transition
	Vector shape		Windows Media
	Xtra		

To turn off or on the display of cast member icons in Thumbnail view and change the Cast window display:

- Select Edit > Preferences > Cast. For more information, see "Setting Cast window preferences" on page 36.

  Note: If you are using a Macintosh OS X operating system, select the Director menu, instead of the Edit menu, to access Preferences.

Creating a custom cast member thumbnail

Note: For most cast members, Director displays a scaled version as the thumbnail unless you define a custom thumbnail. Creating a custom thumbnail is useful for behaviors that you want to identify in the Library palette, because behaviors have no identifying image.

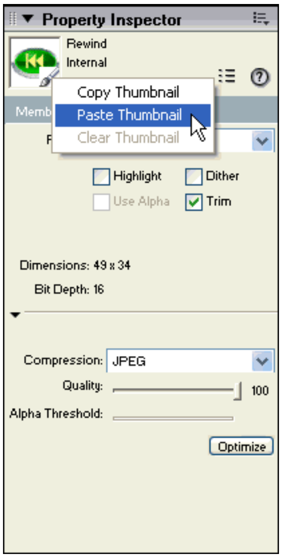
To create a custom cast member thumbnail:

1 Select the bitmap image to use as the new thumbnail, and copy it to your system's Clipboard.

  You can copy the image from any bitmap editor, including the Paint window. The image can be any size, but smaller images look better because they require less scaling.

2 Select any cast member in the Cast window and open the Property inspector.

3 Right-click or Control-click (Macintosh) the Thumbnail window.



Dimensions: 49 x 34
Bit Depth: 16
Compression: JPEG
Quality: 100
Alpha Threshold:

4 From the context menu, select Paste Thumbnail.

  The image from the Clipboard replaces the current cast member thumbnail, creating a custom thumbnail.

  You can also use text as a thumbnail. Create and copy text from any bitmap editor instead of an image and follow the same steps.

Moving cast members within the Cast window

To move a cast member to a new position within the Cast window, you can use Thumbnail view to see the representation of the cast member's position.

Note: When you move a cast member to a new position, Director assigns it a new number and updates all references to the cast member in the Score, but it doesn't automatically update references to cast member numbers in Lingo or JavaScript syntax. The best practice is to always name cast members and refer to them by name in scripts that refer to them.

To move a cast member to a new position or a different cast:

- Using Thumbnail view, drag the cast member to a new position in any open Cast window.

  In Thumbnail view, a highlight bar indicates where the cast member will be placed. If you drag the cast member over a position that already contains a cast member, Director places your selected cast member in that position and moves the existing cast member one position to the right.

  In List view, the cast member is added to the bottom of the list.

To cut, copy, and paste cast members to a new position or a different cast:

1 Select one or more cast members, then select Cut or Copy from the Edit menu.

2 Do one of the following:

- In Thumbnail view, select an empty position in any open Cast window, and then select Edit > Paste.
- In List view, deselect all cast members by clicking anywhere in the window except on a cast member name. Then select Edit > Paste.

Note: In either Thumbnail or List view, if you paste cast members while other cast members are selected, you will overwrite the selected cast members.

To move a cast member to a position that is not currently visible in Thumbnail view:

1 Select the cast member you want to move.

2 Scroll the Cast window to display the destination position.

3 Drag the Drag Cast Member button to the destination position.

Organizing cast members within the Cast window

The Sort command in the Modify menu helps clean up and organize the Cast window. Use Sort to sort cast members by their media type, name, size, or usage in the Score. You can also use Sort to remove empty positions in a Cast window.

When you use the Sort command to organize a Cast window, Director can move cast members to new positions, with new cast member numbers.

Note: If you've written scripts that refer to cast members by number, these scripts can't find moved cast members. To avoid this problem, always name your cast members and refer to them by name in your scripts.

If you want to view the cast members in a different sort order without changing cast member numbers, click a column title in Cast List view. See "Sorting Cast List view columns" on page 32.

To sort the cast using the Modify menu:

1 With the Cast window active, select the cast members to sort or select Edit > Select All.

2 Select Modify > Sort.

3 In the Sort Cast Members dialog box, select one of the following sorting methods:

 Usage in Score places selected cast members that are used in the Score at the beginning of the selection.

 Media Type groups all cast members according to their media type.

 Name groups the selection alphabetically by cast member name.

 Size arranges the selection with the largest files appearing first.

 Empty at End places all empty cast positions in the selection at the end.

4 Click Sort.

 Director reorders the cast members according to the sorting method you selected. The Score automatically adjusts to the new cast member numbers.

Setting Cast window preferences

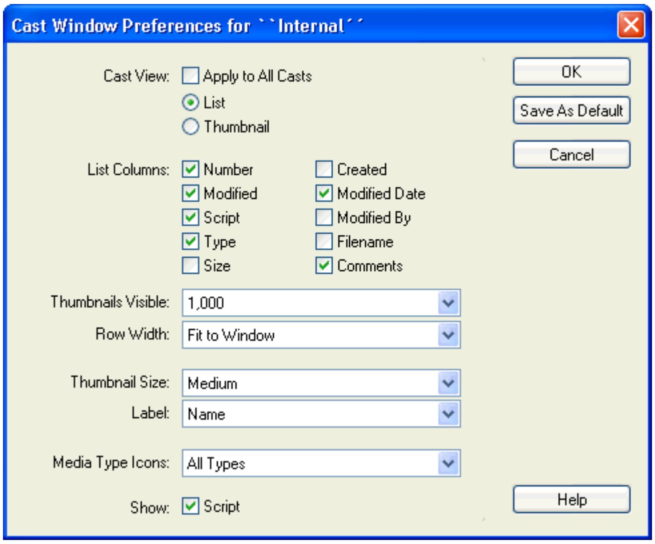
You use the Cast window preferences settings to control the appearance of the current Cast window or, if desired, all Cast windows. You can set different preferences for each Cast window. The title bar of the dialog box displays the name of the Cast window preferences you are changing.

To set Cast window preferences:

1 Select a Cast window to change, or a click a tab within a Cast panel group.

2 Select Edit > Preferences > Cast.

 Note: If you are using a Macintosh OS X operating system, select the Director menu, instead of the Edit menu, to access Preferences.



Cast Window Preferences for ``Internal´´

Cast View: ☐ Apply to All Casts
● List
○ Thumbnail

List Columns:
☑ Number ☐ Created
☑ Modified ☑ Modified Date
☑ Script ☐ Modified By
☑ Type ☐ Filename
☐ Size ☑ Comments

Thumbnails Visible: 1,000
Row Width: Fit to Window
Thumbnail Size: Medium
Label: Name
Media Type Icons: All Types
Show: ☑ Script

3 To set the Cast window to display in either List or Thumbnail view, select the appropriate Cast View option.

4 If you want your preferences to apply to all Cast windows, select Apply to All Casts.

5 To select the columns that appear in Cast List view, select the desired List Columns options. See "Using Cast List view" on page 31.

6 To specify the maximum number of cast members to appear in the Cast window, select a value from the Thumbnails Visible pop-up menu.

This option does not limit the 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 positions to make better use of the vertical scroll bar. The default is 1000.

7 To specify the number of thumbnails in each row of the Cast window, select an option from the Row Width pop-up menu.

The options for 8 Thumbnails, 10 Thumbnails, and 20 Thumbnails 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 scroll bar 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 scroll bar is disabled because the entire width of the cast is always in view. The default is Fit to Window.

8 To set the size of each cast thumbnail image that appears in the Cast window, select one of the following options from the Thumbnail Size pop-up menu:

Small 44 x 33 pixels

Medium 56 x 42 pixels (default)

Large 80 x 60 pixels

Thumbnails always maintain the standard 4:3 aspect ratio.

If the thumbnails appear fuzzy, they are probably displaying larger than their original size. To correct this, change the Cast window preferences thumbnail setting to a smaller size. Click OK when the alert message asks whether thumbnails should be regenerated.

9 To select the display format of the cast member ID that appears below each cast thumbnail image in the Cast window, select one of the following options from the Label pop-up menu:

Number displays the cast number.

Name displays the cast name, if one exists; otherwise, this option displays the cast number in decimal format.

Number:Name displays the cast number in decimal format and the cast name, separated by a colon (:) (for example, 340:Dancing Potato). If no name exists, this setting displays the cast number in decimal format.

The selected format is also used in other windows, including the Score, whenever a cast ID appears.

10 To specify whether Director displays an icon in the lower right corner of each cast member that indicates the cast member's type, select one of the following from the Media Type Icons pop-up menu: All Types, All but Text and Bitmap, or None.

11 To display a script indicator icon in the lower left corner of each cast member that has a script attached, select Script.

12 To make your preference settings the default settings, click Save as Default.

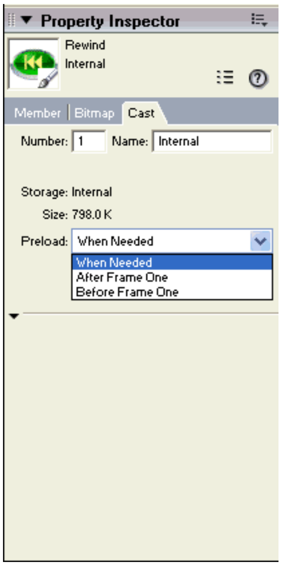
13 When you finish selecting your preferences, click OK.

Changing Cast properties

You use the Property inspector to change the name of a Cast and to define how its cast members are loaded into memory.

To change Cast window properties:

1 With the Cast window as the active window, open the Property inspector and click the Cast tab.



2 To change the name of the current cast, enter the new name in the Name text box.

3 Select one of the following Preload options to define how cast members are loaded into memory when the movie runs:

When Needed loads each cast member into memory when it is required by the movie. This setting can slow down the movie while it plays, but it makes the movie begin playing sooner. This setting is the best choice when controlling cast members loading with Lingo or JavaScript syntax.

After Frame One loads all cast members (except those required for frame 1) when the movie exits frame 1. This setting can ensure that the first frame appears as quickly as possible, and it might be the best choice if the first frame of the movie is designed to remain onscreen for a number of seconds.

Before Frame One loads all cast members before the movie plays frame 1. This setting makes the movie take longer to start playing, but it provides the best playback performance if there is enough memory to hold all cast members.

Viewing and setting cast member properties

You can display and set properties for individual cast members, or for multiple cast members at once, even if the cast members are different types. In both cases, you use the Property inspector.

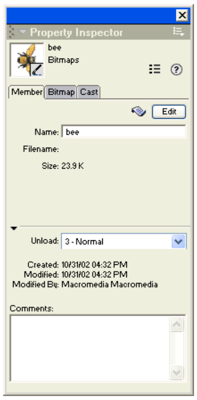
You can also set cast member properties by using Lingo or JavaScript syntax (see "Setting cast member properties using Lingo or JavaScript syntax" on page 49).

To view and set cast member properties:

1 Select one or more cast members.

2 Do one of the following:

- If the Property inspector is open, click the Member tab.
- If the Property inspector is not open, select Window > Property Inspector, and click the Member tab.



bee
Bitmaps

Name: bee
Filename:
Size: 23.9 K
Unload: 3 - Normal
Created: 10/31/02 04:32 PM
Modified: 10/31/02 04:32 PM
Modified By: Macromedia Macromedia
Comments:

As with all fields in the Property inspector, if you've selected multiple cast members, the information that is common to all the selected cast members appears. Any changes you make apply to all the selected cast members.

3 Display the Graphical view on the Member tab.

The Member tab displays the following items:

- Editable fields to view or change the cast member's name (the Name text box), a Comments text box to enter text that appears in the Comments column of the Cast List window, and an Unload pop-up menu that lets you select how to remove a cast member from memory. For more information about using the Unload pop-up menu, see "Controlling cast member unloading" on page 47.
- View-only fields which indicate the cast member's size, when the cast member was created and modified, and the name of the person who modified the cast member.

For an Xtra cast member, the information displayed in the Property inspector is determined by the developer of the Xtra. Some Xtra extensions have options in addition to those listed here. For non-Macromedia Xtra extensions, refer to documentation supplied by the developer.

For information about specific cast member properties, see the following topics:

- "Using animated GIFs" on page 101"
- "Embedding fonts in movies" on page 164
- "Using Flash Content" on page 181
- "Setting bitmap cast member properties" on page 128
- "Setting vector shape properties" on page 142
- "Synchronizing media" on page 240
- "Setting film loop properties" on page 95
- "Setting palette cast member properties" on page 155
- "Setting PICT cast member properties" on page 128
- "To specify a shape's fill in Lingo or JavaScript syntax:" on page 144
- "Setting sound cast member properties" on page 232
- "Setting text or field cast member properties" on page 174
- "Setting transition cast member properties" on page 162
- "Setting Xtra cast member properties" on page 49
- "Creating an animated color cursor cast member" on page 299
- "Streaming linked Shockwave Audio and MP3 audio files" on page 238
- "Setting Flash component parameters" on page 209
- "Setting Windows Media properties" on page 253
- "Setting DVD Cast member properties" on page 257

Launching cast member editors

You can open any cast member in the appropriate editor directly from the Cast window. You can use the Director internal media editors, such as the Text, Paint, or Vector Shape window, or you can specify external editors for certain types of cast members. For more information, see "Launching external editors" on page 46.

To launch an editor for a cast member, do one of the following:

- Double-click a cast member in the Cast window.
- Double-click a sprite that contains the cast member in the Score or on the Stage. See "Sprites" on page 51.

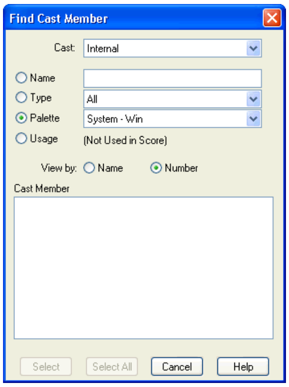
Finding cast members

You can search for cast members by name, type, and color palette. You can search for selected cast members used in the Score, such as when you are preparing a movie for distribution. You can also search for cast members not used in the Score—for example, to clean up a movie and reduce the space and memory required to save and run the movie.

Before releasing a movie, it's a good idea to remove unused cast members to make the movie as small as possible for downloading.

To find cast members:

1 Select Edit > Find > Cast Member.


Find Cast Member

Cast: Internal

Name [ ]
Type: All
[✓ Palette]: System - Win
Usage (Not Used in Score)

View by: Name [✓ Number]

Cast Member

2 In the Find Cast Member dialog box, select a Cast window to search from the Cast pop-up menu.

To search every cast in the movie, select All Casts.

3 Select a search option:

- Select Name, and enter search text in the text box. For example, to search for a group of related cast members that share a common element in their names, you might enter the word `Bird` to search for cast members named Bird 1, Bird 2, and Bird 3.
- Select Type, and select an option from the pop-up menu to search for cast members by media type.
- Select Palette, and select an option from the pop-up menu. You can use this option to search for and resolve palette conflicts.
- Select Usage to locate all cast members that aren't used in the Score. Cast members that you find with this option might be used in the movie by a script.

Director displays the specified cast member.

4 Do one of the following:

- Select a cast member on the list, and click Select to close the dialog box and select the cast member in the Cast window.
- Click Select All to close the dialog box and select all listed cast members in the Cast window.

To find a cast member in the Score:

1 Select a cast member for which to search in the cast or the Score. If you select a sprite that includes multiple cast members, Director searches for the first cast member in the sprite; to select a cast member other than the first, open the sprite to select the cast member. (For information about selecting sprites, see "Selecting sprites" on page 53.)

2 Select Edit > Find > Selection, or press Control+Shift+F (Windows) or Command+Shift+F (Macintosh).

Director searches the Score and highlights the first Score cell it finds.

3 Select Edit > Find Again to find the next occurrence of the cast member in the Score.

Importing cast members

Importing lets you create cast members from external media. You can either import data into a Director movie file or create a link to the external file and re-import the file each time the movie opens. Linked files let you display dynamic media from the Internet, such as sports scores, sounds, and weather pictures, which makes downloading movies faster. For more information about linked files, see "About linking to files" on page 44.

Director can import cast members from almost every popular media file format. See "About import file formats" on page 44.

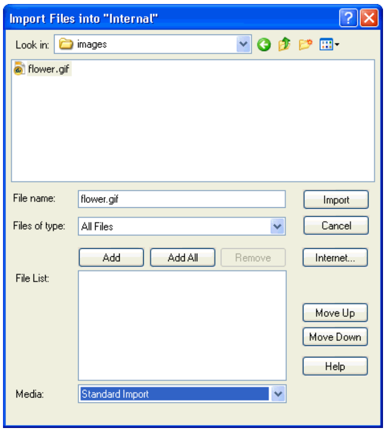
You can import files by using the Import dialog box, dragging files from the desktop to a Cast window, or by using Lingo or JavaScript syntax.

To import cast members and specify import options:

1 In Thumbnail view, select an empty position in a cast.

If no cast position is selected, Director places the new cast member in the first available position in the current cast in Thumbnail view. In List view, Director places the new cast member at the end of the list.

2 Select File > Import.



3 To import a file from the Internet, click Internet and enter a URL in the Find URL text box.

4 To import local files, select the type of media to import from the Files of Type (Windows) or the Show (Macintosh) pop-up menu.

All the files in the current directory appear unless you make a selection.

5 To select a file or files to import, do one of the following:

- Double-click a file.
- Select one or more files, and click Add.
- Click Add All.

You can switch folders and import files from different folders at the same time.

6 From the Media pop-up menu at the bottom of the dialog box, select an option to specify how to treat imported media:

Standard Import imports all selected files, storing them inside the movie file but not updating them when changes are made to the source material. If you selected the option to import from the Internet in step 3, Director retrieves the file immediately from the Internet if a connection is available.

Note: All digital video files like DVD, Windows Media, QuickTime, RealMedia, and so on, are linked automatically to the original external file (see the next option, Link to External File), even if you select Standard Import.

Link to External File creates a link to the selected files and imports the data each time the movie runs. If you choose to import from a URL via the Internet, the media is dynamically updated. For more information, see "About linking to files" on page 44.

Note: Text and RTF files are always imported and stored inside the movie file (see the previous option, Standard Import), even if you select Link to External File.

Include Original Data for Editing preserves the original data in the movie file for use with an external editor.

When this option is selected, Director keeps a copy of the original cast member data and sends the original to the external editor when you edit the cast member. This option preserves all the editor's capabilities. For example, if you specify Photoshop to edit PICT images, Director maintains all the Photoshop object data. For more information, see "Launching external editors" on page 46.

- Import PICT File as PICT prevents PICT files from being converted to bitmaps.

7 When you finish selecting the files, click Import.

If you've imported a bitmap with a color depth or color palette that differs from the current movie, the Image Options dialog box appears, so you can enter additional information. See "Choosing import image options" on page 45.

For information about importing specific media, see the following sections:

- "About importing bitmaps" on page 100
- "Importing internal and linked sounds" on page 231
- "Using Director movies within Director movies" on page 202
- "Importing internal and linked sounds" on page 231
- "Importing text" on page 165
- "Using animated GIFs" on page 101
- "Using Flash Content" on page 181
- "Importing Windows Media" on page 252
- "Using DVD media content in Director" on page 254
- "Using Flash components" on page 207

To import files by dragging:

1 In the Explorer (Windows) or on the system desktop (Macintosh), select a file or files to import.

2 Drag the files from the desktop to the desired position in the Cast window Thumbnail view or to the Cast window List view.

If you drag the files to List view, the imported files are added at the bottom of the list.

To import files with Lingo or JavaScript syntax:

- Use the `importFileInto` method to import a file. Set the `fileName` cast member property to assign a new file to a linked cast member. For more information about this property, see the Scripting Reference topics in the Director Help Panel.

About import file formats

Director can import files in all the formats listed in the following table. For information about additional file formats Director might support, see the Director Support Center website at www.macromedia.com/support/director.

Type of file	Supported formats
Animation and multimedia	Flash movies, animated GIF, PowerPoint presentations, Director movies, Director external cast files
Image	BMP, GIF, JPEG, LRG (xRes), Photoshop 3.0 (or later), MacPaint, PNG, TIFF, PICT, Targa
Multiple-image file	Windows only: FLC, FLI Macintosh only: PICS, Scrapbook
Sound	AIFF, WAV, MP3 audio, Shockwave Audio, Sun AU, uncompressed and IMA compressed
Digital Video	DVD; Windows Media (WMV); QuickTime; AVI; RealMedia
Text	RTF, HTML, ASCII (often called Text Only), Lingo, or JavaScript syntax
Palette	PAL, Photoshop CLUT

About linking to files

Director reimports media every time a movie runs when Link to External File is selected in the Import dialog box (select File > Import). Linking makes it easy to use bulky media such as long sounds and is especially useful for showing media from the Internet that changes frequently. Linking also makes downloading movies faster; users can choose whether to view linked files, so the files do not download unless they're needed.

When you link to an external file, Director creates a cast member that stores the name and location of the file. Saving a movie saves only the link to the linked cast member. Keep linked files in a folder that's close to the original movie file. Paths are restricted to 4096 characters by the system. URLs can be as many as 260 characters. If you store a file too many folders away from the movie or using a very long URL, it might not link correctly.

When distributing movies with linked media, use the following guidelines:

- If you distribute a movie, you also must include all linked cast member files, and they must be in their expected locations. In addition, the Xtra extensions that are used to import the media must be present when the movie runs (either on the user's computer or included in your movie). For more information, see "Setting Xtra cast member properties" on page 49.
- When you link to media on the Internet, the media must be present at the specified URL when the movie runs. Provide for link failure because you can't guarantee that an Internet transaction will be successful.

- To retrieve media from the Internet during playback, Director requires that the projector include certain Xtra extensions. To include these Xtra extensions automatically, click Add Network in the Movie Xtras dialog box. Movies playing in web browsers do not require these Xtra extensions.

  Note: Select Edit > Preferences > Network to define standard network settings for the Director authoring environment. If you are using a Macintosh OS X operating system, select the Director menu, instead of the Edit menu, to access Preferences.

Choosing import image options

If you import a bitmap cast member with a color depth or color palette that is different from that of the Stage (the current movie), Director lets you select the image's color depth and color palette. You can choose to import the bitmap at its original color depth or at the Stage color depth. (The Stage color depth is the same as the system color depth.) You can also choose to import the image's color palette or remap the image's colors to a palette in the movie.

In many cases, it's easiest to change the image's color depth to the depth of the movie and remap the image to the color palette that is used in the rest of the movie. For more information about controlling color in Director, see Chapter 7, "Color, Tempo, and Transitions," on page 145.

If you change 16-, 24-, or 32-bit cast members to 8 or fewer bits, you must remap the cast members to an existing color palette.

To select bitmap image options for importing:

1 Import a bitmap image by selecting File > Import. (For more information about this procedure, see "Importing cast members" on page 42.)

2 If the Image Options dialog box appears while you are importing a bitmap image using File > Import, select one of the following Color Depth options:

  Image specifies the color depth and palette of the image.

  Stage specifies the color depth of the current Stage.

3 Select a Palette option to change palette settings for 2-, 4- or 8-bit images:

  Import imports the image with its color palette. The palette appears as a new cast member immediately following the bitmap cast member.

  Remap To replaces the image's colors with the most similar solid colors in the palette you select from the pop-up menu.

4 Select Image options:

  Trim White Space removes any white pixels from the edges of the image. Deselect this option to preserve the white canvas around an image.

  Dither blends the colors in the new palette in the Palette section to approximate the original colors in the graphic.

5 To apply the current settings to all the remaining files that you selected for importing, select Same Settings for Remaining Images.

Launching external editors

You can specify external applications to edit many types of media. All the types of media for which you can define an external editor are listed in the Editors Preferences dialog box. After you set up an external editor for a particular media type, Director starts the application when you edit a cast member of that type. When you finish editing a cast member in an external editor and then save and close the file, Director re-imports the cast member media.

You can easily edit Flash cast members using the launch-end-edit feature in Director MX. For more information, see "Editing a Flash cast member" on page 184.

If you want to use an external editor for an imported cast member, select Include Original Data for External Editing during import. For more information, see "Importing cast members" on page 42.

You cannot define an external editor for any cast member created by an Xtra, such as text, vector shapes, and custom pointers.

To define an external editor:

1 Select Edit > Preferences > Editors.

   Note: If you are using a Macintosh OS X operating system, select the Director menu, instead of the Edit menu, to access Preferences.

2 Select a type of media for which you want to define an external editor.

3 Click Edit.

4 Click Browse or Scan to locate the application.

   You can specify any application capable of editing the selected type of media.

5 To determine which editor appears when you double-click a cast member, do one of the following:

   - If you prefer to make changes inside of Director and only occasionally want to use the external editor, select Use Internal Editor.
   - If you prefer to use the external editor to make changes to the cast member, select Use External Editor.

To launch an external editor:

1 Select a cast member of a media type for which you have defined an external editor, and do one of the following:

   - If you specified Use External Editor when you defined the external editor for this media type, double-click the cast member.
   - Select Edit > Launch External Editor.
   - While the cast member is selected and the Cast window is active, right-click (Windows) or Control-click (Macintosh) and select Launch External Editor from the context menu.

   Director launches or switches to the application that created the cast member, sending the original data to the external editor.

   Note: If you've specified an external editor and you want to edit a cast member with the Director internal editors, select the cast member and select Edit > Edit Cast Member.

2 Edit the cast member.

   If you change an image in the Paint window and then edit the image with an external editor, changes made in the Paint window, with the exception of registration points, are lost. Director warns you of this possibility.

3 Save and close the file. Director re-imports the cast member.

Controlling cast member unloading

When Director runs low on memory, it removes cast members from memory. You use the Property inspector to specify the priority with which a cast member is removed from memory. When a cast member is available in memory, it appears almost instantly. When it needs to be loaded from disk, the loading can cause a delay. Set your cast members so that frequently used cast members remain in memory as long as possible.

These settings are the same for all types of cast members.

To specify the Unload setting:

1 Select the cast members in the Cast window.

2 On the Property inspector Member tab, display the Graphical view and then select an option from the Unload pop-up menu:

   3–Normal sets the selected cast members to be removed from memory after any priority 2 cast members are removed.

   2–Next sets the selected cast members to be among the first removed from memory.

   1–Last sets the selected cast members to be the last removed from memory.

   0–Never sets the selected cast members to be retained in memory; these cast members are never unloaded.

Managing external casts

An external cast is a separate file that must be explicitly linked to a movie for the movie to use its cast members.

If you link an external cast to a movie, Director opens the cast every time it opens the movie. If you don't link an external cast to a movie, you must open and save the file separately. You can use unlinked external casts as libraries to store commonly used elements for authoring, such as scripts, buttons, and so on. For more information, see "Creating libraries" on page 48.

When you distribute a movie that uses an external cast, you must include the external cast file. For disk-based movies, the cast must be in the same relative path in your files as it was when the movie was created. For Shockwave movies on the web, the cast must be at the specified URL.

To create an external cast:

1 Select File > New > Cast.

2 Type a name for the new cast.

3 Specify that the cast be stored as an external cast.

   If you don't want to use the cast in the current movie, deselect the Use in Current Movie option.

4 Click Create.

The cast is created, and a Cast window for the cast appears in List view. For more information, see "Using the Cast window" on page 26.

5 Select File > Save while the Cast window is active, and save the cast in the desired directory.

To link an external cast to a movie:

1 Select Modify > Movie > Casts.

2 In the Movie Casts dialog box, click Link.

3 Locate and select the external cast you want, and click Open.

You can link to casts on your local disk or to casts that are stored at any URL. Click Internet and enter a URL (in the Find URL text box) for a linked external cast. Click OK.

To unlink a cast from a movie:

1 Select Modify > Movie > Casts.

2 In the Movie Casts dialog box, select the external cast.

3 Click Remove.

To save a movie and all open casts, linked or unlinked:

- Select File > Save All.

Note: To use a cast member from an external cast without creating a link to the external cast, first copy the cast member to an internal cast or to a (different) linked external cast.

Creating libraries

A library is a special type of unlinked external cast that appears in the Library palette. When you drag a cast member from an external cast library to the Stage or Score, Director automatically copies the cast member to one of the movie's internal casts. Libraries are useful for storing any type of commonly used cast members, especially behaviors. A library cannot be linked to a movie. For more information, see "Attaching behaviors" on page 275.

When you create a library, as explained in the following procedure, it appears in the Library List pop-up menu in the Library palette.

To create a library:

1 Create an external cast file, following the procedure under "Creating new casts" on page 23. Do not select Use in Current Movie.

2 With the Cast window for the external cast active, select File > Save, and place the external cast in the Libs folder in the Director application folder.

3 Restart Director to see the cast that you just created appear in the Library palette.

Setting cast member properties using Lingo or JavaScript syntax

Lingo or JavaScript syntax lets you control and edit cast members by setting their properties. Some properties are available for every type of cast member, and other properties are available only for specific cast member types. For more information about these properties, see the Scripting Reference topics in the Director Help Panel.

To specify the cast member's content:

- Set the `media` cast member property.

To specify the cast member's name:

- Set the `name` cast member property.

To set the contents of the cast member's comments field:

- Set the `comments` cast member property. You can store any text information in this field that you find useful, and access it at runtime by getting the `comments` property.

To specify the cast member's purge priority:

- Set the `purgePriority` cast member property.

To specify the content of the script that is attached to the cast member:

- Set the `scriptText` cast member property.

To specify the file that is assigned to a linked cast member:

- Set the `fileName` cast member property.

For additional cast member properties that you can test and set using Lingo or JavaScript syntax, see the properties in the Scripting Reference topics in the Director Help Panel.

Setting Xtra cast member properties

Xtra cast members have the same Name and Unload properties as other cast members, but they can also contain an extra panel of options that is accessible from the Property inspector. To set cast member properties, use the Member tab and the custom tab for the type of cast member you are working with. The Member tab contains an Edit button and might contain a More Options button, depending on the type of Xtra. Use the Edit button to edit the cast member with its default editor. Use the More Options button to display the Cast Member Properties dialog box for the current cast member.

The custom tab for the type of cast member you are working with might also contain a More Options button. This button displays the Cast Member Properties dialog box for the current cast member.

The content of the Properties dialog box is determined by the developer of the Xtra. For non-Macromedia Xtra extensions, refer to any documentation that the developer supplies.

To view or change Xtra cast member properties:

1 Select an Xtra cast member.

2 Open the Property inspector, and click the Member tab.

  The Member tab displays the following information about the member:

  - The cast member name
  - The name of the cast that contains the cast member
  - The size in kilobytes
  - The creation date
  - The date the cast member was last modified
  - The name of the user who last modified the cast member

3 Use the Name field to view or edit the cast member name.

4 To specify how Director removes cast members from memory if memory is low, select options from the Unload pop-up menu. For information about these options, see "Controlling cast member unloading" on page 47.

5 To set special options for the current Xtra cast member, click the custom tab for the cast member you are working with. Some types of Xtra cast members also have a More Options button on this tab. You can use this button to set any properties of the cast member that are not displayed on the tab.

CHAPTER 3
Sprites

A sprite is an object that controls when, where, and how cast members appear in a Macromedia Director MX 2004 movie. Multiple sprites can use the same cast member. You can also switch cast members assigned to a sprite as the movie plays. You use the Stage to control where a sprite appears, and you use the Score to control when it appears in your movie.

Sprites appear on the Stage layered according to the channel in which they are assigned in the Score. Sprites in higher-numbered channels appear in front of sprites in lower-numbered channels. A movie can include as many as 1000 sprite channels. Use the Movie tab of the Property inspector to control the number of channels.

Sprite properties include the sprite's size and location, the cast member assigned to the sprite, the sprite's name, and the frames in which the sprite occurs. Different properties can alter the appearance of a sprite. You can rotate, skew, flip, and change the color of sprites without affecting cast members. You can change sprite properties with the Property inspector or Lingo or JavaScript syntax.

You can also give each sprite a unique name. You can assign a name by using the Property inspector, and then view the sprite by name in the Score and on the Stage. Assigning a name lets you refer to the sprite by that name in Lingo or JavaScript syntax and not just by the channel number that it occupies. You can move a sprite to a different channel and not worry about changing scripts. Editing scores and code scripts is much easier when you refer to a sprite by its name.

In Lingo or JavaScript syntax, some properties are available only for certain types of sprites. Such properties typically are characteristics that are related to the specific sprite type. For example, Lingo or JavaScript syntax has several digital video properties that determine the contents of tracks in digital video sprites.

Creating sprites

You create a sprite by dragging a cast member to either the Stage or the Score: the sprite appears in both places. New sprites, by default, span 30 frames.

To create a new sprite:

1 Click to select the frame in the Score where you want the sprite to begin.

2 From the Cast window, in either List or Thumbnail view, do one of the following:

- Drag a cast member to the position on the Stage where you want to place the sprite.
- Drag a cast member to the Score. Director places the new sprite in the center of the Stage.
- To create a sprite one frame long, press Alt (Windows) or Option (Macintosh) and drag a cast member to the Stage or Score.

Setting sprite general preferences

You use the Sprite Preferences dialog box to control the way sprites behave and appear in the Score window and on the Stage.

To change preferences for sprites:

1 Select Edit > Preferences > Sprite.

Note: If you are using a Macintosh OS X operating system, select the Director menu, instead of the Edit menu, to access Preferences.

2 To determine if selecting a sprite on the Stage selects the entire span of the sprite or only the current frame in the sprite, select one of the following Stage Selection options:

Entire Sprite selects the sprite in all frames that it occupies.

Current Frame Only selects only the current frame of the sprite.

3 To determine the appearance and behavior of sprites yet to be created, select the following Span Defaults options. These options don't change settings for existing sprites.

Display Sprite Frames turns on Edit Sprite Frames for all new sprites. See "Editing sprite frames" on page 89.

Tweening turns on tweening for all tweenable properties. This option is on by default. With this option off, sprites must be manually tweened when new frames or keyframes are added to the sprite. For additional information about tweening, see Chapter 4, "Animation," on page 83.

4 To determine the length of sprites measured in frames, select the following Span Duration options:

Frames defines the default number of frames for sprites.

Width of Score Window sets the sprite span to the visible width of the Score window.

Terminate at Markers makes new sprites end at the first marker.

5 To specify the frame used as the beginning of a sprite span when creating or editing new sprites on the Stage, select from the following Span Starts options:

Previous Marker sets the sprite span to begin at the sprite's previous marker.

Current Frame sets the sprite span to begin at the current frame.

First Frame in Score Window sets the sprite span to begin at the first frame in the current score window.

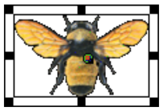
Selecting sprites

To edit or move a sprite, you must select it. You can select sprites, frames within sprites, and groups of sprites in several ways.

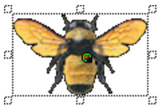
You use the Arrow tool on the Tool palette to select sprites before most operations. You can also select sprites with the Rotate and Skew tool to enable rotation and skewing. See "Rotating and skewing sprites" on page 72.

When selecting sprites, you often want to select a certain frame or range of frames within the sprite instead of the entire sprite. When you make certain changes to a frame within a sprite, it becomes a selectable object called a keyframe. See "Editing sprite frames" on page 89.

A selected sprite appears on the Stage with a double border. When you select a single frame within a sprite, the sprite appears on the Stage with a single border.



Entire sprite selected



Single frame within sprite selected

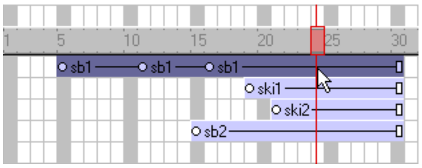
To select sprites, do one of the following:

Note: The following techniques select an entire sprite only if Edit Sprite Frames is not enabled for the sprite(s) you select.

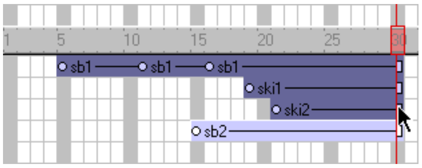
- On the Stage, click a sprite to select the entire sprite span.

  You can change sprite preferences so that selecting a sprite on the Stage selects only the current frame instead of the entire sprite. See "Setting sprite general preferences" on page 52.

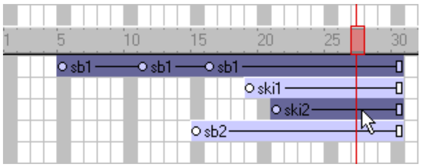
- In the Score, click the horizontal line within a sprite bar; don't click the keyframes, the start frame, or the end frame.



- To select a contiguous range of sprites either on the Stage or in the Score, select a sprite at one end of the range and then Shift-click a sprite at the other end of the range. You can also drag to select all the sprites in an area.

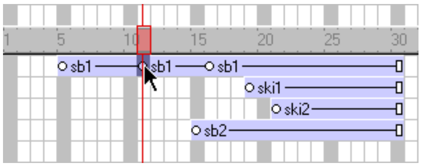


- To select discontiguous sprites, Control-click (Windows) or Command-click (Macintosh) the discontiguous sprites.

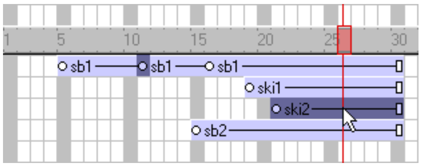


To select a keyframe, do one of the following:

- To select only a keyframe, click the keyframe indicator.

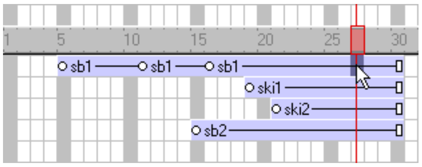


- To select a keyframe and sprites at the same time, Control-click (Windows) or Command-click (Macintosh) the keyframe and the desired sprites.



To select a frame within a sprite that is not a keyframe, do one of the following:

- In the Score, Alt-click (Windows) or Option-click (Macintosh) the frame within the sprite.



- On the Stage, Alt-click (Windows) or Option-click (Macintosh) to select only the current frame of the sprite. The sprite appears on the Stage with a single border.

To select all the sprites in a channel:

- Click the channel number at the left side of the Score.

Naming sprites

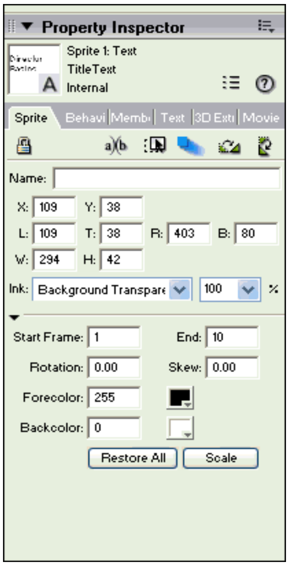
You can assign a name to a sprite by using the Property inspector and then view the sprite by name in the Score and on the Stage. Assigning a name lets you refer to the sprite by that name in Lingo or JavaScript syntax and not just by the channel number that it occupies. You can move a sprite to a different channel and not worry about changing scripts that once referred to the sprite by its channel number. The sprite name is different from a cast member name because a sprite is an instance of that cast member. If you want the sprite name to be displayed in the Score and Stage, select Edit > Preferences > Score > Name. (If you are using a Macintosh OS X operating system, select the Director menu, instead of the Edit menu, to access Preferences.) For more information, see, "Changing Score settings" on page 20.

To name a sprite using the Property inspector:

1 Select a sprite in the Score or on the Stage.

2 Select Window > Property Inspector, and select the Sprite tab.

3 Enter a name for the selected sprite in the Name text box.

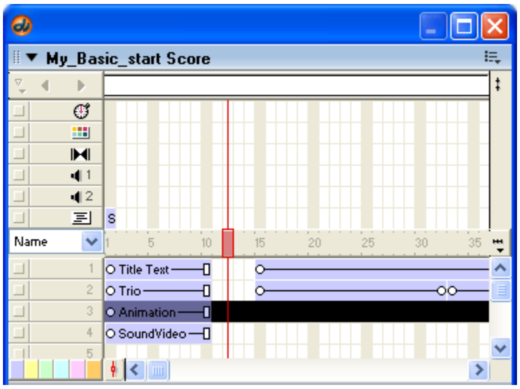


You can name the sprite anything you like, but you may want something that is easily recognizable and that will be easy to script.

To view a sprite in the Score by its name:

1 In the Score, select the Sprite labels pop-up menu.

2 Select Name. All the sprites in the score appear with their sprite name listed.



To view a sprite on the Stage by its name:

- Select a sprite on the Stage. The sprite name appears on the second line of the sprite overlay.

To edit the name of a sprite:

1 Select the sprite and open the Property inspector.

2 Enter a name in the Name text box.

Setting the Sprite name property in Lingo or JavaScript syntax

The new sprite name property is accessible as a standard sprite property. The `name` property can be read at all times, but names can only be assigned via script when in score recording mode.

The syntax is as follows:

```
put sprite(1).name -- this displays the name in the message window.
```

You can also refer to the sprite by using its given name when evoking script commands on the sprite. For example:

```
sendSprite ("pete", #handlername) -- call the "handlername" method in
  sprite("pete")
put sprite("somename").rect -- display the sprite's rect
```

To find out what sprite is targeted for a particular name, you can use the following expression:

```
sprite("myName").spriteNum
```

To create a name for a sprite, the assignment must be made in score recording mode. Note that you don't have to use the updateFrame command, but the record frame must be a frame where the sprite exists in the score.

```
beginrecording
  sprite(2).name = "tubular"
endrecording
```

For more information about this property, see the Scripting Reference topics in the Director Help Panel.

Finding sprites

You can search for and find sprites by name. When you have many sprites in a movie, searching for a sprite by name is easier and more efficient than searching by channel name or number.

To find a sprite by name:

1 Select Edit > Find > Find Sprite. The Find Sprite dialog box opens with a list of the named sprites in the current cast.

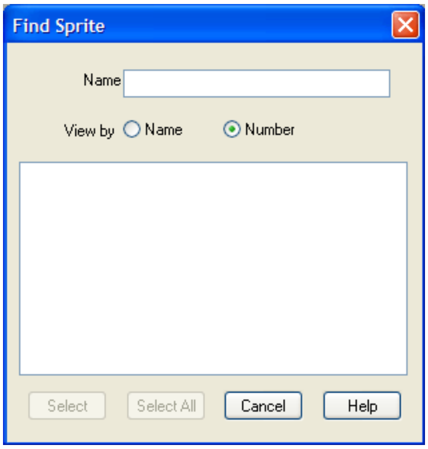
Note: Only the sprites that you have already named appear in the list.

2 In the Name text box, type the name of the sprite that you want to find.

You can type just the first letters to limit the list to sprites that start with those letters.

3 Select Name if you want the sprites sorted by name; select Number if you want the sprites sorted by Channel number.

4 Select the sprite that you want to find and click Select. The score opens at the location of the selected sprite.



View by: Name [✓ Number]

Creating sprite channel names

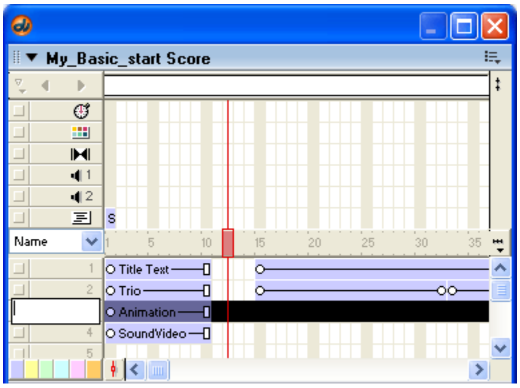
Rather than just referring to channels by number, you can also name sprite channels. When authoring sprites in Lingo or JavaScript syntax, you often must work in a particular channel or manage several different channels. Naming a sprite channel can expedite your work when authoring and managing many composite layers.

To name a sprite channel:

1 Double-click a channel in the Score sprite channel column.

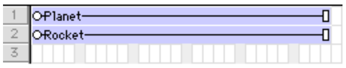
A text box appears.

2 Type a name for the channel and press Enter.



Layering sprites

A sprite appears in front of other sprites on the Stage according to its channel. Sprites in higher-numbered channels appear in front of sprites in lower-numbered channels.



The rocket in channel 2 appears in front of the planet in channel 1.

To change a sprite's layer:

1 In the Score, select the sprite. To select the contents of an entire channel, click the channel number at the left side of the Score.

2 Do one of the following:

- Select Modify > Arrange, and select a command from the submenu to change the order of sprites.
- Drag the sprite in the Score from one channel to another.
- If you selected a channel, drag its contents to another channel.

Note: If you give each sprite a unique name, you don't have to update any scripts you have written when you move a sprite to a new channel. For more information, see "Naming sprites" on page 55.

Displaying and editing sprite properties

As you work with sprites in your movie, you'll want to monitor and possibly modify sprite properties. Director offers several methods of accomplishing this by using one or more of the following:

- The Property inspector
- The Sprite toolbar, which includes a subset of Sprite text boxes found in the Property inspector
- The Sprite Overlay, which displays, directly on the Stage, the most commonly used properties for selected sprites
- Sprite labels, which appear within the sprite bars in the Score and let you view important sprite properties
- Script in Lingo or JavaScript syntax

Displaying and editing sprite properties in the Property inspector

Depending on your preference, you can use either the Sprite toolbar or the Property inspector to perform many of the same procedures.

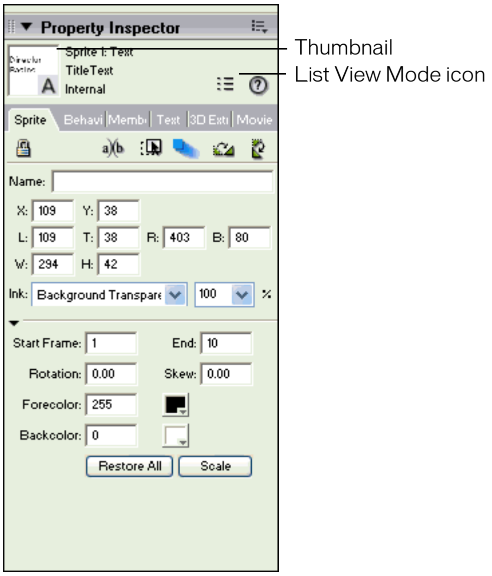
To display and edit sprite properties in the Property inspector:

1 Select one or more sprites on either the Stage or the Score.

2 If the Property inspector is not open, select Window > Property Inspector.

The Property inspector opens with focus on the Sprite tab. The Graphical view is the default view. You can toggle to the List view by clicking the List View Mode icon.

The Property inspector displays settings for the current sprite. If you select more than one sprite, the Property inspector displays only their common settings.



Sprite 1: Text
TitleText
Internal

Name:

X: 109 Y: 38
L: 109 T: 38 R: 403 B: 80
W: 294 H: 42
Ink: Background Transpare 100 %

Start Frame: 1 End: 10
Rotation: 0.00 Skew: 0.00
Forecolor: 255
Backcolor: 0

A thumbnail image of the sprite's cast member appears in the upper left corner of the Property inspector.

Note: To open a window in which you can edit the sprite's cast member, you can double-click the thumbnail image.

3 Edit any of the following sprite settings in the Property inspector:

Lock changes the sprite to a locked sprite so you or other users can't change it. For additional information about locked sprites, see "Locking and unlocking sprites" on page 63.

Editable applies only to text sprites and lets you edit the selected text sprite on the Stage during playback. For more information, see "Selecting and editing text on the Stage" on page 166.

Moveable lets you position the selected sprite on the Stage during playback. For more information, see "Visually positioning sprites on the Stage" on page 65.

Trails makes the selected sprite remain on the Stage, leaving a trail of images along its path as the movie plays. If Trails is not selected, the selected sprite is erased from previous frames as the movie plays.

Flip Horizontal and **Flip Vertical** reverse the sprite horizontally or vertically to form an inverted image. See "Flipping sprites" on page 75.

The Name text box lets you enter a name for the sprite. For more information, see "Naming sprites" on page 55.

Reg Point Horizontal (X) and **Vertical (Y)** display the location of the registration point in pixels from the upper left corner of the Stage. For more information, see "Editing sprite properties with Lingo or JavaScript syntax" on page 63.

Left (L), Top (T), Right (R), and **Bottom (B)** show the location of the edges of the sprite's bounding rectangle.

Width (W) and **Height (H)** show the size of the sprite's bounding rectangle in pixels.

The Ink pop-up menu displays the ink of the current sprite and lets you select a new ink color. For more information, see "Using sprite inks" on page 77.

Blend determines the blend percentage of the selected sprites. For more information, see "Setting blends" on page 76.

Start Frame and **End Frame** display the start and end frame numbers of the sprite. Enter new values to adjust how long the sprite plays. For more information, see "Changing the duration of a sprite on the Stage" on page 70.

Rotation rotates the sprite by the number of degrees you enter. For more information, see "Rotating and skewing sprites" on page 72.

Skew slants the sprite by the number of degrees you enter. For more information, see "Rotating and skewing sprites" on page 72.

Forecolor and **Backcolor** color boxes determine the colors of the selected sprite. For more information, see "Changing the color of a sprite" on page 75.

Restore All reverts the height and width to that of the cast member.

Scale opens the Scale Sprite dialog box, where you can resize the selected sprite. For more information, see "Resizing and scaling sprites" on page 71.

Displaying sprite properties in the Sprite toolbar

The Sprite toolbar displays a subset of the same information and text boxes found on the Sprite tab in the Property inspector. You can use either the Sprite toolbar or the Property inspector, depending on your preference, to perform many of the same procedures.

To show or hide the Sprite toolbar in the Score:

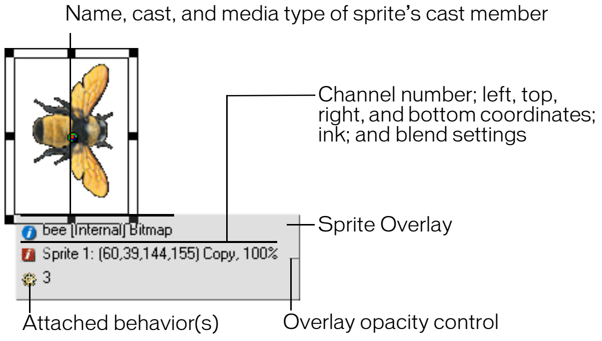
- While the Score is active, select View > Sprite Toolbar. The Sprite toolbar is displayed across the top of the Score.

Using the Sprite Overlay

The Sprite Overlay displays important sprite properties directly on the Stage. You can open editors, inspectors, and dialog boxes to change sprite properties by clicking the corresponding icons in the Sprite Overlay.

To display the Sprite Overlay when a sprite is selected:

- Select View > Sprite Overlay > Show Info.


Name, cast, and media type of sprite's cast member
Channel number; left, top, right, and bottom coordinates; ink; and blend settings
Sprite Overlay
Attached behavior(s)
Overlay opacity control

bee [Internal] Bitmap
Sprite 1: (60,39,144,155) Copy, 100%
3

To use Sprite Overlay options to change how the overlay appears:

1 Click the Sprite on the Stage to select it.

2 In the Sprite Overlay, click the icon that represents the data you want to edit:

- To edit the Sprite's cast member, click this icon to open the tab in the Property inspector that applies to this type of sprite. For example, clicking this icon displays the Vector tab for a vector sprite, the Text tab for a text sprite, and so on.
- To open the Sprite tab in the Property inspector, click this icon.
- To open the Behavior tab in the Property inspector, click this icon. See Chapter 12, "Behaviors," on page 275.

To change the Sprite Overlay's appearance to suit your preferences:

1 Select View > Sprite Overlay > Settings.

2 Select a Display option to determine when sprite properties are visible and active:

Roll Over displays sprite properties only when the pointer is over a single sprite.

Selection displays sprite properties when you select a sprite.

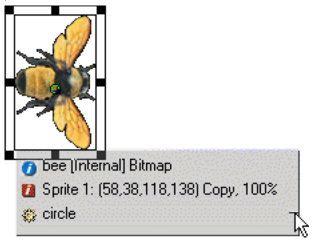
All Sprites displays sprite properties for all sprites on the Stage.

3 Use the Text Color box to select the color for text that appears in the Sprite Overlay.

Tip: If the Stage has a dark background, changing the color of the text to a light color lets you read the sprite information in the overlay.

To change the opacity of the Sprite Overlay:

- Drag up or down the small thin line that appears on the right edge of the Sprite Overlay.

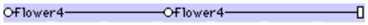

bee [Internal] Bitmap
Sprite 1: (58,38,118,138) Copy, 100%
circle

Displaying sprite labels in the Score

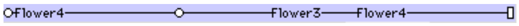
Sprite labels appear in the Score's sprite bars and display key information about the sprite in relation to the movie. For example, if you detect a strange blip caused by an ink effect, you can select Ink from the Sprite label pop-up menu and quickly locate the problem in those sprites that have Ink properties by sorting by the Ink label. You can select the way information displays in channels by selecting from the different sprite labels available; for example, you can use the Extended display option to display the precise location of a sprite in every frame. i

To display sprite labels:

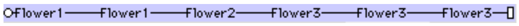
1 With the Score as the active window, do one of the following:
   - Select View > Sprite Labels.
   - Right-click (Windows) or Control-click (Macintosh) on any Score channel, and select Sprite Labels.
2 Select from the following options:
   - Keyframes



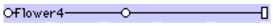
   - Changes Only (shown at 800%)



   - Every frame (shown at 800%)



   - First frame



   - None

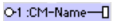
   Many options are useful only when the Score is zoomed to 400% or 800%.

To change sprite label options:

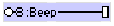
- Select a display option from the Display pop-up menu in the Score or from the View > Display menu.

  Name displays the name of the sprite.

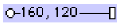
  Cast Member displays the name and number of the sprite's cast member.



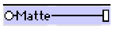
Behavior displays the behavior that is assigned to the sprite.



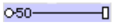
Location displays the *x* and *y* coordinates of the sprite's registration point.



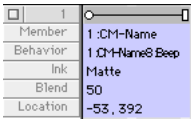
Ink displays the ink effect that is applied to each sprite.



Blend displays the blend percentage.



Extended displays any combination of display options; select options by selecting Edit > Preferences > Score. (If you are using a Macintosh OS X operating system, select the Director menu, instead of the Edit menu, to access Preferences.)



	1
Member	1 :CM-Name
Behavior	1 :CM-Name8:Beep
Ink	Matte
Blend	50
Location	-53, 392

Editing sprite properties with Lingo or JavaScript syntax

You can use Lingo or JavaScript syntax to check and edit sprite properties with scripts as the movie plays.

Note: Sprite properties changed with Lingo are not saved in the score unless you're using score recording.

To check a property value:

- Use the `put` method or check in the Watcher window. For more information, see the Scripting Reference topics in the Director Help Panel.

To edit a property:

- Use the equals (=) operator or the `set` command to assign a new value to the property. For more information, see the Scripting Reference topics in the Director Help Panel.

Locking and unlocking sprites

During authoring, you can lock sprites to avoid inadvertent changes to the sprite, either by you or by someone else working on the same project. When you lock a sprite, you can no longer change its settings, although you still see it represented on the Stage and in the Score. While preserving the settings of your locked sprites, you can continue to create and edit unlocked sprites.

Locking sprites is not supported during playback.

Note: If you try to perform an operation on a group of locked and unlocked sprites, a message appears that indicates the operation will affect only the unlocked sprites.

To lock a sprite:

In the Stage or the Score, select one or more sprites to lock, and do one of the following:

- Select Modify > Lock Sprite.
- On the Sprite tab of the Property inspector, click the padlock icon.
- Right-click (Windows) or Option-click (Macintosh), and select Lock Sprite from the context menu.

In the Score, a locked sprite appears with a padlock in front of its name. On the Stage, a locked sprite appears with a padlock in its upper right corner.

To select a locked sprite on the Stage:

- Hold down the L key while selecting the sprite.

To unlock a sprite:

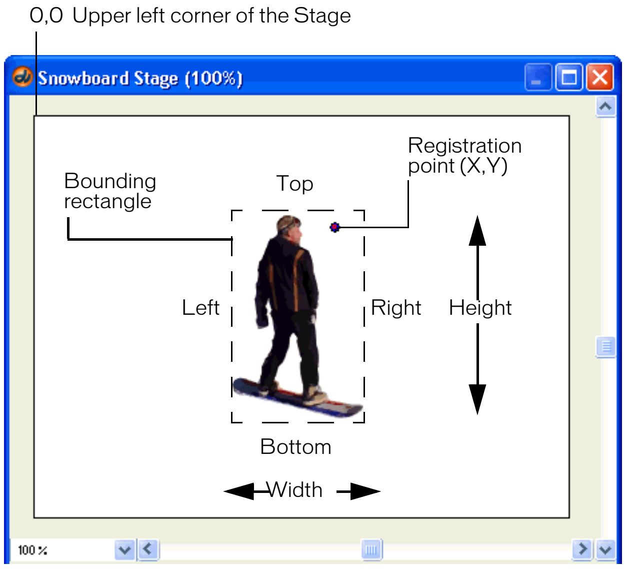
1. In the Score or on the Stage, select one or more sprites to unlock.
2. Do one of the following:
   - Select Modify > Unlock Sprite.
   - On the Sprite tab in the Property inspector, click the padlock icon.
   - Right-click (Windows) or Option-click (Macintosh), and select Unlock Sprite from the context menu.

Positioning sprites

The easiest way to position a sprite is to simply drag the sprite into place on the Stage. To position a sprite more precisely, you can do any of the following:

- Set a sprite's position on the Stage by entering coordinates in the Property inspector.
- Use the Tweak window.
- Use guides or the grid.
- Use the Align window.
- Use the arrow keys to manually move a selected sprite.
- Set the sprite's coordinates in Lingo or JavaScript syntax.

The following diagram shows all the sprite coordinates you can specify.



Director places the image of a cast member on the Stage by specifying the location of its registration point. For many cast members, such as bitmap or vector shapes, the registration point is in the center of the bounding rectangle by default. For other types of cast members, the registration point is at the upper left corner. (For instructions on changing the location of the registration point of bitmap cast members, see "Changing registration points" on page 112. For instructions on changing a vector shape cast member's registration point, see "Editing vector shapes" on page 139.)

Visually positioning sprites on the Stage

You can position sprites on the Stage by dragging them or by using the arrow keys.

To visually position a sprite on the Stage:

1 Select Window > Stage to display the Stage.

2 Do one of the following on the Stage:

- Drag a sprite to a new position. Hold down Shift to limit the movement to horizontal or vertical straight lines.
- Select a sprite and use the arrow keys to move the selected sprite 1 pixel at a time. Hold down Shift as you press an arrow key to move the selection 10 pixels at a time.

To visually position a sprite on the Stage during playback:

1 Select a sprite that you want to position during playback.

2 On the Sprite tab in the Property inspector, click Moveable. See "Displaying and editing sprite properties in the Property inspector" on page 59.

3 Begin playing back the movie.

4 On the Stage, drag the sprite to the new position.

Positioning sprites with the Property inspector

You can use the Property inspector to specify the exact coordinates of a sprite.

To set sprite coordinates in the Property inspector:

1 With the Property inspector open and in Graphical view, select a sprite to reposition.

2 On the Sprite tab in the Property inspector, specify the sprite coordinates in pixels, with 0,0 at the upper left corner of the Stage, as follows:

- Specify attributes in the X and Y text boxes to change the horizontal and vertical coordinates of the registration point.
- Specify coordinates in the W and H text boxes to change the width and height of the sprite.
- Specify values in the L, T, R, and B text boxes to change the left, top, right, and bottom edges of the sprite's bounding rectangle.

To move the sprite without resizing it, adjust only the *x* and *y* coordinates.

Positioning sprites with the Tweak window

You can use the Tweak window when you want to move sprites by a certain number of pixels.

To position sprites with the Tweak window:

1 Select Modify > Tweak.

2 Select the sprite or sprites you want to move, as described in "Selecting sprites" on page 53.

3 In the Tweak window, drag the point on the left side of the window or enter the number of pixels in the text boxes for horizontal and vertical change, and then click Tweak.

4 If you want to repeat the move, click Tweak again.

Positioning sprites using guides, the grid, or the Align window

On the Stage, you can align sprites by using guides, the grid, or the Align window.

The grid consists of cell rows and columns of a specified height and width that you use to assist you in visually placing sprites on the Stage. The grid is always available.

Guides are horizontal or vertical lines you can either drag around the Stage or lock in place to assist you with sprite placement. You must create guides before they become available.

Moving a sprite with the Snap to Grid or Snap to Guides feature selected lets you snap the sprite's edges and registration point to the nearest grid or guide line. When you are not using the guides or the grid, you can hide them.

Guides and the grid are visible only during authoring.

You can create and modify the guides and the grid from the Property inspector or by using menu commands.

To add and configure guides:

1 With the Property inspector open, click the Guides tab.

The top half of the tab contains settings for Guides.

2 To change the guide color, click the Guide Color box and select a different color.

3 Select the desired options to make the guides visible, to lock them, and to make the sprites snap to the guides.

4 To add a guide, move the cursor over the new horizontal or vertical guide, and then drag the guide to the Stage. Numbers in the guide tooltip indicate the distance, in pixels, the guide is located from the top or left edge of the Stage.

5 To reposition a guide, move the pointer over the guide. When the sizing handle appears, drag the guide to its new position.

6 To remove a guide, drag it off the Stage.

7 To remove all guides, click Remove All on the Guides tab in the Property inspector.

To display guides and align sprites:

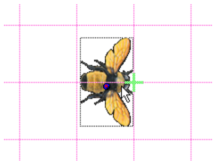
1 If guides don't appear on the Stage, select View > Guides and Grid > Show Guides.

2 If Snap to Guides is not selected, select View > Guides and Grid > Snap to Guides.

3 Move a sprite on the Stage near a guide line to make the sprite snap to that exact location.

To display a grid and align sprites:

1 If grid lines don't appear on the Stage, select View > Guides and Grid > Show Grid.



2 If Snap to Grid is not selected, select View > Guides and Grid > Snap to Grid.

3 Move a sprite on the Stage near a grid line to make the sprite snap to that exact location.

Note: Press the G key while moving or resizing a sprite to temporarily turn Snap to Grid off or on.

To configure the grid:

1 With the Property inspector open, click the Guides tab.

The bottom half of the Guides tab contains Grid settings.

2 To change the grid color, click the Grid Color box and select a different color.

3 Select the desired options to make the grid visible and to make the sprites snap to the grid.

4 To change the width and height of the grid, enter values in the W and H text boxes.

5 Select the desired options to display the grid as dots or lines.

To align sprites using the Align window:

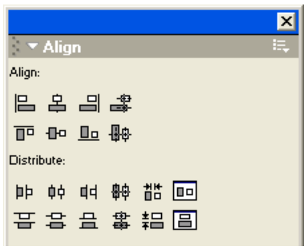
1 On the Stage or in the Score, select the sprites to align.

Select entire sprites, keyframes, or frames within sprites in as many different frames or channels as you need. All of the elements align to the last sprite or frame selected.

2 Select Window > Align to open the Align panel.

3 Select alignment buttons to modify the selected objects:

- For Align, select Align Left Edge, Align Horizontal Center, Align Right Edge, Align Horizontal Registration Point, Align Top Edge, Align Vertical Center, Align Bottom Edge, or Align Vertical Registration Point.
- For Distribute, select Distribute Left Edge, Distribute Horizontal Center, Distribute Right Edge, Distribute Horizontal Registration Point, Distribute Width, Distribute Horizontally Across Stage, Distribute Top Edge, Distribute Vertical Center, Distribute Bottom Edge, Distribute Vertical Registration Point, Distribute Height, or Distribute Vertically Across Stage.



Positioning sprites with Lingo or JavaScript syntax

Script lets you control a sprite's position by setting the sprite's coordinates on the Stage. You can also test a sprite's coordinates to determine a sprite's current position and whether two sprites overlap.

To check the location of a sprite's registration point or bounding rectangle on the Stage:

- Test the `bottom`, `left`, `loc`, `locH`, `locV`, `right`, or `top` sprite property.

  The `bottom`, `left`, `right`, and `top` sprite properties determine the location of the sprite's individual edges. For more information about these properties, see the Scripting Reference topics in the Director Help Panel.

To place a sprite at a specific location:

- Set one of the following properties (for more information about these properties, see the Scripting Reference topics in the Director Help Panel):

  The loc sprite property sets the horizontal and vertical distance from the upper left corner of the Stage to the sprite's registration point. The value is given as a point.

  The locV sprite property sets the number of pixels from the top of the Stage to a sprite's registration point.

  The locH sprite property sets the number of pixels from the left of the Stage to a sprite's registration point.

  The rect sprite property sets the location of the sprite's bounding rectangle on the Stage.

  The quad sprite property sets the location of the sprite's bounding rectangle on the Stage. You can specify any four points; the points don't have to form a rectangle. The `quad` sprite property can set the sprite's coordinates as precise floating-point numbers.

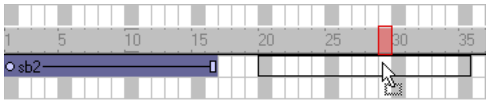
To determine whether two sprites overlap:

- Use the `sprite...intersects` operator to determine whether a sprite's bounding rectangle touches the bounding rectangle of a second sprite. Use the `sprite...within` operator to determine whether a sprite is entirely within a second sprite. For more information about these operators, see the Scripting Reference topics in the Director Help Panel.

Changing when a sprite appears on the Stage

A sprite controls where and when media appear on the Stage. You change when a sprite appears on the Stage by moving the sprite to different frames in the Score and by changing the number of frames the sprite spans. You can either drag sprites to new frames or copy and paste them. Copying and pasting is easier when moving sprites more than one screen width in the Score. You can also copy and paste to move sprites from one movie to another.

Note: When you copy a sprite from one movie to another, save the source movie first.



Moving a sprite in the Score

To change when a sprite appears on the Stage:

1 Select Window > Score to display the Score.

2 Select a sprite or sprites, as described in "Selecting sprites" on page 53.

3 Drag the sprite to a different frame.

To move a sprite without spreading it over additional frames, hold down the Spacebar and drag. This technique is also useful for moving any sprite that consists mostly (or entirely) of keyframes.

To copy or move sprites between frames:

1 Select a sprite or sprites, as described in "Selecting sprites" on page 53.

2 Select Edit > Cut Sprites or Edit > Copy Sprites.

3 Position the pointer where you want to paste the sprite, and select Edit > Paste Sprites.

If the pasting will overwrite existing sprites, select one of the following Paste options in the Paste Options dialog box:

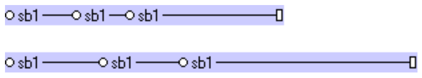
Overwrite Existing Sprites replaces the sprites with the content of the Clipboard.

Truncate Sprites Being Pasted pastes the Clipboard contents in the space available without replacing existing sprites.

Insert Blank Frames to Make Room adds new frames for the contents of the Clipboard.

Changing the duration of a sprite on the Stage

By default, Director assigns each new sprite a duration of 30 frames. You can change the duration of a sprite—that is, the amount of time the sprite appears in a movie—by adjusting its length, changing the number of frames in which it appears, or by using the Extend command.



Director maintains the spacing proportions of keyframes when a sprite is lengthened. For a description of keyframes, see Chapter 4, “Animation,” on page 83.

To extend or shorten a sprite:

1 Select Window > Score to display the Score.

2 Do one of the following:

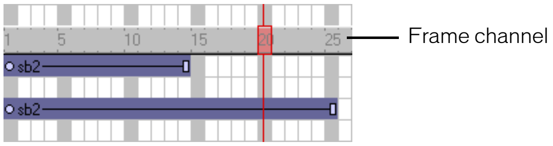
- Drag the start or end frames.To extend a one-frame sprite, Alt-drag (Windows) or Option-drag (Macintosh).
- To extend a sprite and leave the last keyframe in place, Alt-drag (Windows) or Option-drag (Macintosh) a keyframe at the end of the sprite.
- To extend a sprite and leave all keyframes in place, Control-drag (Windows) or Command-drag (Macintosh) the end frame.
- Enter new values in the Start and End text boxes on the Sprite tab in the Property inspector to change the start and end frames.

To extend a sprite to the current location of the playhead:

1 Select the sprite or sprites to extend.

2 Click the frame channel to move the playhead:

- To extend the sprite, move the playhead past the right edge of the sprite.



- To shorten the sprite, move the playhead to the left of the sprite’s right edge, inside the sprite.
- To move the sprite’s start frame, place the playhead to the left of the sprite.

3 Select Modify > Extend Sprite.

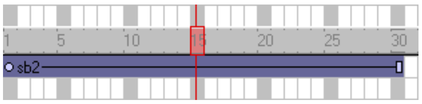
Splitting and joining sprites

You might need to split an existing sprite into two separate sprites or join separate sprites. If, for example, you created a complex animation as separate sprites and now want to move the entire sequence in the Score, you would join the sprites. Splitting and joining also lets you update movies created with older versions of Director that might have several fragmented sprites.

To split an existing sprite:

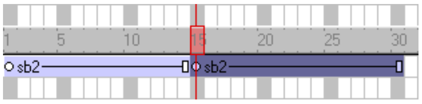
1 In the Score, click the frame within a sprite where you want the split to occur.

The playhead moves to the selected frame.



2 Select Modify > Split Sprite.

Director splits the sprite into two new ones.



To join separate sprites into a single sprite:

1 Select the sprites you want to join, as described in "Selecting sprites" on page 53.

Director fills the gaps between the selected sprites. You can also select sprites in several channels. Director joins selected sprites in each individual channel.

2 Select Modify > Join Sprites.

Changing the appearance of sprites

You can change the appearance of sprites on the Stage without affecting the cast member assigned to the sprite. You can resize, rotate, skew, flip, and apply new foreground and background colors to sprites. Applying these changes allows you to reuse the same cast member to create several different versions of an image. For example, you can create a flipped and rotated sprite with a new color. Since each cast member adds to downloading time, reusing cast members in this way reduces the number of cast members in your movie and makes it download faster. Reusing the same cast member for multiple sprites also reduces the amount of memory required.

Resizing and scaling sprites

You can resize sprites directly on the Stage by dragging their handles. To resize the sprite precisely, you can enter coordinates or scale sprites by a specified percentage on the Sprite tab in the Property inspector. You can also set the sprite's size with Lingo or JavaScript syntax.

Changing a sprite's size on the Stage does not change the size of the cast member that is assigned to the sprite, nor is the size of the sprite affected if you resize its cast member.

In some cases, resizing bitmap sprites can cause noticeable delays. If a bitmap sprite must be a particular size, make the cast members that appear in the sprite the proper size. You can do this with Modify > Transform Bitmap or in any image-editing program. Scaling and resizing sprites works best with vector shapes.

Note: The procedure for resizing a rotated or skewed sprite is different from the following procedures. For more information, see "Rotating and skewing sprites" on page 72.

To resize a sprite by dragging its handles:

1 Select the sprite.

2 On the Stage, drag any of the sprite's resize handles. Hold down Shift while dragging to maintain the sprite's proportions.

To scale a sprite by pixels or by an exact percentage:

1 Select the sprite you want to scale and click the Sprite tab of the Property inspector (Graphical view).

2 Click the Scale button.

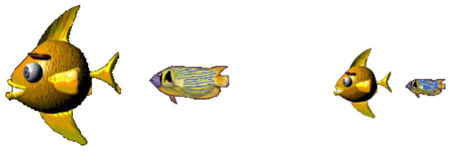
The Scale Sprite dialog box appears.

3 Enter new values to scale the sprite by doing one of the following:

- Specify a pixel size in the Width or Height text boxes. If Maintain Proportions is selected, all the updatable text boxes adjust to reflect the new scaled size. If Maintain Proportions is not selected, you can specify new proportions in the Width and Height text boxes.
- Enter a percentage in the Scale text box.

4 Click OK.

The sprite is scaled relative to its current size not to the size of its parent cast member.



To restore a sprite to its original dimensions, do one of the following:

- On the Sprite tab in the Property inspector (Graphical view), click Restore All.
- Select Modify > Transform > Reset Width and Height or Reset All.

To resize a sprite's bounding rectangle with script:

- Set the sprite's `quad` or `rect` sprite property. For more information about these properties, see the Scripting Reference topics in the Director Help Panel.

  The `rect` sprite property determines the coordinates of a sprite's bounding rectangle. The coordinates are given as a `rect` value, which is a list of the left, top, right, and bottom coordinates.

To change a sprite's height or width with script:

- Set the `height` or `width` sprite property. For more information about these properties, see the Scripting Reference topics in the Director Help Panel.

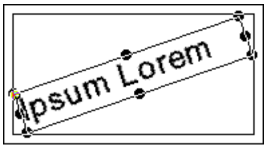
Rotating and skewing sprites

You can rotate and skew sprites to turn and distort images and to create dramatic animated effects. You rotate and skew sprites on the Stage by dragging. To rotate and skew sprites more precisely, use Lingo or JavaScript syntax or the Property inspector to enter degrees of rotation or skew. The Property inspector is also useful for rotating and skewing several sprites at once by the same angle.

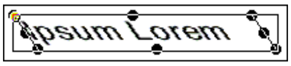
Director can rotate and skew bitmaps, text, vector shapes, Macromedia Flash content, QuickTime videos, and animated GIFs.

Director rotates a sprite around its registration point, which is a marker that appears on a sprite when you select it with your mouse. By default, Director assigns a registration point in the center of all bitmaps. You can change the location of the registration point by using the Paint window. For more information, see "Changing registration points" on page 112.

Rotation changes the angle of the sprite. Skewing changes the corner angles of the sprite's rectangle.



Rotated sprite



Skewed sprite

After a sprite is rotated or skewed, you can still resize it.

Director can automatically change rotation and skew from frame to frame to create animation. See "Tweening other sprite properties" on page 86.

To rotate or skew a sprite on the Stage:

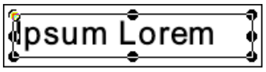
1 Select a sprite on the Stage.

2 Select Window > Tool Palette to display the Tool palette.

3 Click the Rotate and Skew tool in the Tool palette.

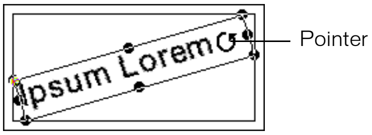
You can also press Tab while the Stage window is active to select the Rotate tool.

The handles around the sprite change to indicate the new mode.

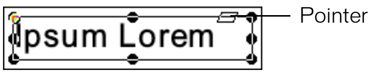


4 Do either of the following:

- To rotate the sprite, move the pointer inside the sprite and drag in the direction you want to rotate.



- To skew the sprite, move the pointer to the edge of the sprite until it changes to the skew pointer and then drag in the direction you want to skew.

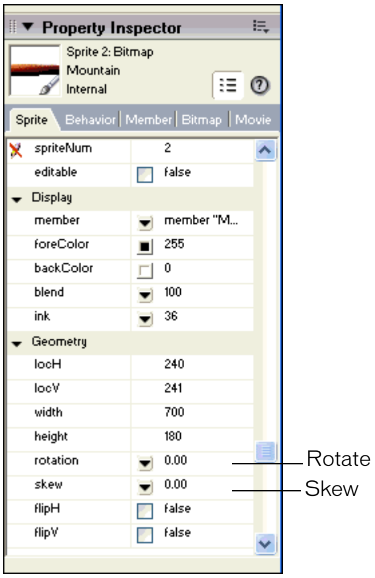


To rotate or skew a sprite with the Property inspector:

1 Select the sprite you want to rotate or skew and click the Sprite tab in the Property inspector (List view).

2 To rotate the selected sprite, display the Rotation pop-up menu and enter the number of degrees in the Rotation text box.

3 To skew the selected sprite, display the Skew pop-up menu and enter the number of degrees in the Skew text box.



spriteNum	2
editable	false
member	member "M...
foreColor	255
backColor	0
blend	100
ink	36
locH	240
locV	241
width	700
height	180
rotation	0.00
skew	0.00
flipH	false
flipV	false

To resize a rotated or skewed sprite, do one of the following:

- Click the Rotate and Skew tool and drag any of the sprite's handles. Use Alt-drag (Windows) or Option-drag (Macintosh) to maintain the sprite's proportions as you resize.
- Enter new values on the Sprite tab in the Property inspector.

  Director resizes the sprite at the current skew or rotation angle.

To restore a skewed or rotated sprite to its original orientation:

- Select Modify > Transform > Reset Rotation and Skew or Reset All.

To skew a sprite with script:

- Set the `skew` sprite property. For more information about this property, see the Scripting Reference topics in the Director Help Panel.

Flipping sprites

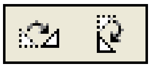
Flipping a sprite creates a horizontally or vertically inverted image of the original sprite.

To flip a sprite:

1 Select a sprite.

2 Do any of the following:

- Click the Flip Vertical or Flip Horizontal button on the Sprite tab in the Property inspector to flip the sprite without moving the registration point or changing the current skew or rotation angles.



- Select Modify > Transform > Flip Horizontal in Place or Flip Vertical in Place to flip the sprite so that its bounding rectangle stays in place and the registration point is moved, if necessary.
- Select Modify > Transform > Mirror Horizontal or Mirror Vertical to flip the sprite without moving the registration point but inverting the skew and rotation angles.

Changing the color of a sprite

You can tint or color sprites by selecting new foreground and background colors from the Property inspector or with Lingo or JavaScript syntax. Selecting a new foreground color changes black pixels within the sprite to the selected color and blends dark colors with the new color. Selecting a new background color changes white pixels within the sprite to the selected color and blends light colors with the new color.

Director can animate foreground and background color changes in sprites, shifting gradually between the colors you specify in the start and end frames of a sprite. See "Tweening other sprite properties" on page 86.

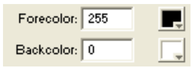
To reverse the colors of an image, change the foreground color to white and the background color to black.

To change the color of a sprite:

1 Select a sprite.

2 Do one of the following:

- Select colors from the Forecolor and Backcolor boxes on the Sprite tab in the Property inspector.



Forecolor: 255
Backcolor: 0

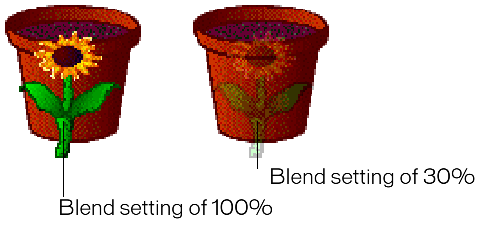
- Enter RGB values (hexadecimal) or palette index values (0-255) for the foreground and background colors on the Sprite tab in the Property inspector.

To change the color of a sprite with Lingo or JavaScript syntax, set the appropriate sprite property:

- The `color` sprite property sets the sprite's foreground color. The value is an RGB value. For more information about this property, see the Scripting Reference topics in the Director Help Panel.
- The `bgColor` sprite property sets the sprite's background color. The value is an RGB value. For more information about this property, see the Scripting Reference topics in the Director Help Panel.

Setting blends

You can use blending to make sprites transparent. To change a sprite's blend setting, use the Sprite tab in the Property inspector.



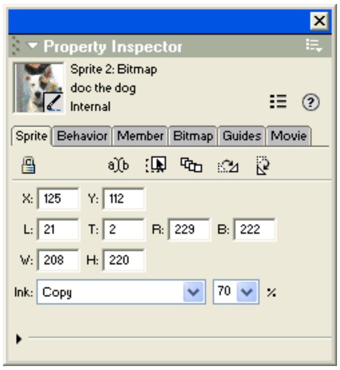
Director can gradually change blend settings to make sprites fade in or out. See "Tweening other sprite properties" on page 86.

The Blend percentage value affects only Copy, Background Transparent, Matte, Mask, and Blend inks.

To set blending for a sprite:

1 Select the sprite.

2 Select a percentage from the Blend pop-up menu in the Property inspector, or enter a blend percentage between 0 and 100.



X: 125 Y: 112
L: 21 T: 2 R: 229 B: 222
W: 208 H: 220
Ink: Copy 70 %

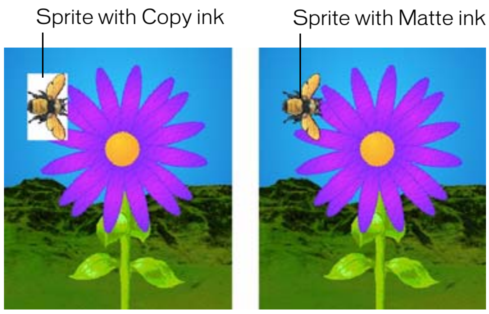
To set blending with Lingo or JavaScript syntax:

- Set the `blend` sprite property. For more information about this property, see the Scripting Reference topics in the Director Help Panel.

Using sprite inks

You can change a sprite's appearance on the Stage by applying inks. Sprite inks change the display of a sprite's colors. Inks are most useful to hide white bounding rectangles around images, but they can also create many compelling and useful color effects. Inks can reverse and alter colors, make sprites change colors depending on the background, and create masks that obscure or reveal portions of a background.

You change the ink for a sprite in the Property inspector or with Lingo or JavaScript syntax.



To achieve the fastest animation rendering on the screen, use Copy ink; other ink types might have a slight effect on performance.

To change a sprite's ink with the Property inspector:

1 Select the sprite.

2 Select the desired type of ink from the Ink pop-up menu on the Sprite tab in the Property inspector.

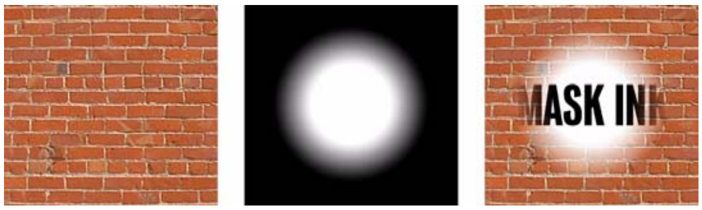
To change a sprite's ink with script:

- Set the sprite's `ink` sprite property. For more information about this property, see the Scripting Reference topics in the Director Help Panel.

  Note: If Background Transparent and Matte inks don't seem to work, the background of the image might not be true white. Also, if the edges of the image have been blended or are fuzzy, applying these inks might create a halo effect. Use the Paint window or an image-editing program to change the background to true white and harden the edges. You can also re-create the image with an alpha channel (transparency) and reimport the image.

Using Mask ink to create transparency effects

To reveal or tint certain parts of a sprite, you use Mask ink. Mask ink lets you define a mask cast member, which controls the degree of transparency for parts of a sprite.



The original cast member, its mask, and the sprite with Mask ink applied.

Black areas of a mask cast member make the sprite completely opaque in those areas, and white areas make it completely transparent (invisible). Colors between black and white are more or less transparent; darker colors are more opaque.

When creating a bitmap mask for a sprite, use a grayscale palette if the mask cast member is an 8-bit (or less) image. An 8-bit mask affects only the transparency of the sprite and does not affect the color. Director ignores the palette of mask cast members that are less than 32-bit images; using a grayscale palette lets you view the mask in a meaningful way. If your mask cast member is a 32-bit image, the colors of the mask tint the sprite's colors.

If you don't need variable levels of opacity, use a 1-bit mask cast member to conserve memory and disk space.

There are many ways to use Mask ink, but the following procedure explains the most basic method.

To use Mask ink:

1 Decide which cast member you want to mask.

The cast member can be a bitmap of any depth.

2 In the next position in the same cast, create a duplicate of the cast member to serve as the mask.

The mask cast member can actually be any image, but a duplicate of the original is usually the most useful.

3 Edit the mask cast member in the Paint window or any image editor.

Black areas of the mask make the sprite completely opaque in those areas, and white areas make it completely transparent (invisible).

4 Drag the original cast member to the Stage or Score to create a sprite.

5 Make sure the new sprite is selected, and select Mask ink from the Ink pop-up menu on the Sprite tab in the Property inspector.

Only the parts of the sprite that are revealed by the mask are visible on the Stage.

About Darken and Lighten inks

Darken and Lighten inks provide a great control over a sprite's RGB properties. You use them to create color effects in sprites varying from subtle to surreal.

Darken and Lighten each change how Director applies the foreground and background color properties of a sprite. Darken makes the background color equivalent to a color filter through which the sprite is viewed on the Stage. Lighten tints the colors in a sprite lighter as the background color gets darker. For both inks, the foreground color is added to the image to the degree allowed by the other color control. Neither ink has any effect on a sprite until you change the foreground or background color from the default settings of black and white.

Darken and Lighten are especially useful for animating unusual color effects. Because the Foreground and Background color properties of the sprite control the effects, you can animate color shifts to create dazzling effects without having to manually edit colors in a cast member. See "Tweening other sprite properties" on page 86.

Ink definitions

The following definitions describe all available ink types.

Copy displays all the original colors in a sprite. All colors, including white, are opaque unless the image contains alpha channel effects (transparency). Copy is the default ink and is useful for backgrounds or for sprites that don't appear in front of other artwork. If the cast member is not rectangular, a white box appears around the sprite when it passes in front of another sprite or appears on a nonwhite background. Sprites with the Copy ink animate faster than sprites with any other ink.

Matte removes the white bounding rectangle around a sprite. Artwork within the boundaries is opaque. Matte functions much like the Lasso tool in the Paint window in that the artwork is outlined rather than enclosed in a rectangle. Matte, like Mask, uses more RAM than the other inks, and sprites with this ink animate more slowly than other sprites.

Background Transparent makes all the pixels in the background color of the selected sprite appear transparent and permits the background to be seen.

Transparent makes all light colors transparent so you can see lighter objects beneath the sprite.

Reverse reverses overlapping colors. When applied to the foreground sprite, where colors overlap, the upper color changes to the chromatic opposite (based on the color palette currently in use) of the color beneath it. Pixels that were originally white become transparent and let the background show through unchanged. Reverse is good for creating custom masks.

Ghost, like Reverse, reverses overlapping colors, except nonoverlapping colors are transparent. The sprite is not visible unless it is overlapping another sprite.

Not Copy reverses all the colors in an image to create a chromatic negative of the original.

Not Transparent, Not Reverse, and **Not Ghost** are all variations of other effects. The foreground image is first reversed, then the Copy, Transparent, Reverse, or Ghost ink is applied. These inks are good for creating odd effects.

Mask determines the exact transparent or opaque parts of a sprite. For Mask ink to work, you must place a mask cast member in the Cast window position immediately following the cast member to be masked. The black areas of the mask make the sprite opaque, and white areas are transparent. Colors between black and white are more or less transparent; darker colors are more opaque. See "Using Mask ink to create transparency effects" on page 77.

Blend ensures that the sprite uses the color blend percentage that is specified on the Sprite tab in the Property inspector. See "Setting blends" on page 76.

Darkest compares RGB pixel colors in the foreground and background and uses the darkest pixel color.

Lightest compares RGB pixel colors in the foreground and background and uses the lightest pixel color.

Add creates a new color that is the result of adding the RGB color value of the foreground sprite to the color value of the background sprite. If the value of the two colors exceeds the maximum RGB color value (255), Director subtracts 256 from the remaining value so the result is between 0 and 255.

Add Pin is similar to Add. The foreground sprite's RGB color value is added to the background sprite's RGB color value, but the color value is not allowed to exceed 255. If the value of the new color exceeds 255, the value is reduced to 255.

Subtract subtracts the RGB color value of the foreground sprite's color from the RGB value of the background sprite's color to determine the new color. If the color value of the new color is less than 0, Director adds 256 so the remaining value is between 0 and 255.

Subtract Pin subtracts the RGB color value of pixels in the foreground sprite from the value of the background sprite. The value of the new color is not allowed to be less than 0. If the value of the new color is negative, the value is set to 0.

Darken changes the effect of the Foreground and Background color properties of a sprite to create dramatic color effects that generally darken and tint a sprite. Darken ink makes the background color equivalent to a color filter through which the sprite is viewed on the Stage. White provides no filtering; black darkens all color to pure black. The foreground color is then added to the filtered image, which creates an effect that is similar to shining light of that color onto the image. Selecting Darken ink has no effect on a sprite until you select nondefault foreground and background colors. See "About Darken and Lighten inks" on page 78.

Lighten changes the effect of the Foreground and Background color properties of a sprite so that it is easy to create dramatic color effects that generally lighten an image. Lighten ink makes the colors in a sprite lighter as the background color gets darker. The foreground color tints the image to the degree allowed by the lightening. See "About Darken and Lighten inks" on page 78.

Note: Mask and Matte use more memory than other inks because Director must duplicate the mask of the artwork.

Assigning a cast member to a sprite with Lingo or JavaScript syntax

Several script properties specify the cast member that is assigned to a sprite. You can use these properties to determine a sprite's cast member and switch the sprite's cast members as the movie plays.

To specify the cast member, including its cast:

- Set the `member` sprite property. For more information about this property, see the Scripting Reference topics in the Director Help Panel.

  Setting this property is the most reliable way to specify a sprite's cast member. You can also set the `memberNum` sprite property, but this is reliable only when the new cast member is in the same cast as the current cast member.

To determine which cast contains the cast member assigned to a sprite:

- Test the `castLibNum` sprite property. For more information about this property, see the Scripting Reference topics in the Director Help Panel.

  This procedure is useful for updating movies that serve as templates.

Exchanging cast members

The Exchange Cast Members command changes the cast member associated with a particular sprite to the currently selected cast member; that is, it replaces the member reference for the selected sprite with the member reference of the selected cast member.

This command modifies the selected sprites in the score; it doesn't modify cast members. The command is enabled only when both a sprite and a cast member are selected.

If multiple cast members are selected, the command associates only the first selected cast member with the sprite.

You can execute this command while the movie is playing.

To exchange cast members:

1 Select a sprite in the Score.

2 Select a cast member in the Cast window, a different cast member than the one already associated with the sprite.

3 Select Edit > Exchange Cast Members.

The selected cast member is associated with the sprite.

CHAPTER 4
Animation

Animation is the appearance of an image changing over time. The most common types of animation in Macromedia Director MX 2004 involve moving a sprite on the Stage (tweening animation) and using a series of cast members in the same sprite (frame-by-frame animation).

- *Tweening* is a traditional animation term that describes the process in which a lead animator draws the animation frames where major changes take place, called keyframes. Assistants draw the frames in between. Tweening in Director lets you define properties for a sprite in frames called keyframes, and Director changes the properties in the frames in between. Tweening is very efficient for adding animation to movies for websites, since no additional data needs to download when a single cast member changes.
- Frame-by-frame animation involves manually creating every frame in an animation, whether that involves switching cast members for a sprite or manually changing settings for sprites on the Stage.

Other forms of animation include making a sprite change size, rotate, change colors, or fade in and out.

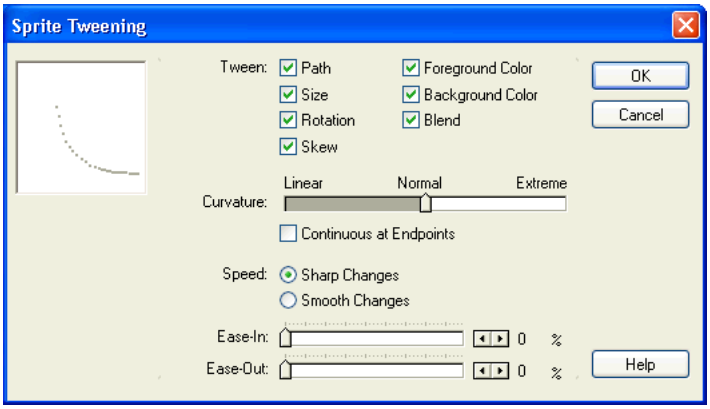
About tweening in Director

To use tweening in Director, you define properties for a sprite in frames called keyframes and let Director change the properties in the frames in between. Tweening is very efficient for adding animation to movies for websites, since no additional data needs to download when a single cast member changes.

To specify tweening properties for a sprite, you use the Sprite Tweening dialog box.

To open the Sprite Tweening dialog box:

- Select a sprite, then select Modify > Sprite > Tweening.


Sprite Tweening

Tween: [✓] Path [✓] Size [✓] Rotation [✓] Skew [✓] Foreground Color [✓] Background Color [✓] Blend

Curvature: Linear — Normal — Extreme (set to Normal)

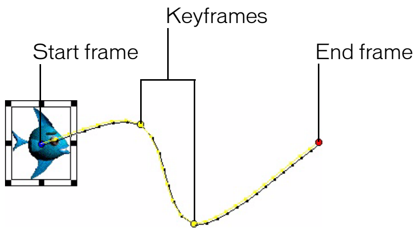
[ ] Continuous at Endpoints

Speed: [✓ Sharp Changes] Smooth Changes

Ease-In: 0 %

Ease-Out: 0 %

A keyframe usually indicates a change in sprite properties. Properties that can be tweened are position, size, rotation, skew, blend, and foreground and background color. Each keyframe defines a value for all of these properties, even if you only explicitly define one.



Tweening the path of a sprite

Sprite paths are the lines Director displays on the Stage to show the movement of a sprite. Sprite paths are controlled by the Sprite Overlay Settings dialog box. You can change settings to make the paths appear for all sprites, for selected sprites, or only when the pointer rolls over a sprite. For more information, see "Using the Sprite Overlay" on page 61.

You can tween a sprite directly on the Stage by editing the sprite's path. Director displays the path of the selected sprite directly on the Stage. You can adjust the path by dragging keyframe indicators.

To tween the path of a sprite:

1 Place a sprite on the Stage where you want the path to start. If the sprite is already on the Stage, select it.

  This places the start frame of the sprite in the proper location. The start frame is also the first keyframe of the sprite.

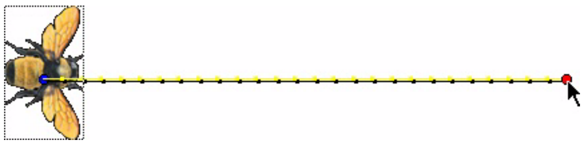
2 If necessary, select View > Sprite Overlay > Show Paths.

  The Show Paths option is on by default. With this option turned on, Director displays the paths of moving sprites on the Stage. Keyframes appear as hollow circles. Small tick marks show the sprite's position in tweened frames.

3 Insert keyframes in any additional frames where you want the sprite's animation path to change.

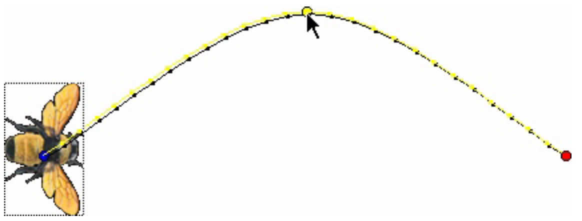
4 Drag the red handle within the sprite to the place on the Stage where you want the sprite's path to end.

The red handle represents the sprite's location in the end frame. For bitmaps, the red handle is usually in the center of the image. For vector shapes and other media types, the handle is often in the upper left corner.



5 Director displays the path the sprite follows. The tick marks along the path show the sprite location in each frame in between.

6 To make the sprite's path curve between more points, hold down the Alt key (Windows) or Option key (Macintosh) and move the pointer on the Stage over a tick mark. When the pointer changes color, drag the tick mark to a new location.



This creates a new keyframe and records the new location. Repeat this step to create additional keyframes.

7 To make the property changes defined by a keyframe occur at a different time, drag the keyframe in the Score to a new frame within the sprite.

8 To change the degree of curvature between keyframes, select Modify > Sprite > Tweening and adjust the Curvature slider. To make the sprite move in the same direction at the beginning and end, select Continuous at Endpoints in the Sprite Tweening dialog box. This creates a circular motion. For more information, see "Changing tweening settings" on page 88.

Accelerating and decelerating sprites

To create more natural motion in tweened sprites, use the following settings in the Sprite Tweening dialog box:

- Ease-In and Ease-Out control how a sprite moves from its start frame to its end frame, no matter how many keyframes are in between. Ease-In makes a sprite move more slowly in the beginning frames; Ease-Out makes the sprite slow down in the ending frames. This setting makes the sprite move more like an object in the real world.

- The Speed settings control how Director moves a sprite between each keyframe. The Sharp Changes option is the default setting. Using this option, Director calculates how to move the sprite between each pair of keyframes separately. If a sprite's keyframes are separated by unequal numbers of frames in the Score, or by different amounts of space on the Stage, abrupt changes in speed may occur as the sprite moves between keyframe locations. Smooth out these speed changes by selecting the Smooth Changes option.

Sprite with modified ease-in and ease-out settings

To change the acceleration or deceleration of a sprite:

1 Use one of the tweening methods to create a moving sprite.

2 Select View > Sprite Overlay > Show Paths to see how far the sprite moves between each frame.

3 Select the sprite and select Modify > Sprite > Tweening.

4 Use the Ease-In and Ease-Out sliders to specify the percentage of the sprite's path through which the sprite should accelerate or decelerate.

5 Select one of the following speed settings:

Sharp Changes moves the sprite between keyframe locations without adjusting the speed.

Smooth Changes adjusts the sprite's speed gradually as it moves between keyframes.

Tweening other sprite properties

In addition to tweening a sprite's path, Director can tween the size, rotation, skew, blend, and foreground and background color of a sprite. Tweening size works best for vector-based cast members created in the Vector Shape window or in Macromedia Flash MX 2004 (bitmaps can become distorted when resized). Director can tween all of these properties at once.

To make a sprite fade in or out, you can tween blend settings. To make sprites spin or tilt, use rotation. To create gradual shifts in color, you can tween color settings.

Note: To prevent Director from tweening a certain sprite property, select Modify > Sprite > Tweening and turn off any of the tweening options.

To tween sprite properties:

1 If the Score isn't open, select Window > Score.

2 Position a sprite on the Stage and make sure it spans all the frames in which you want the sprite to change.

3 Select the start frame of the sprite in the Score.

4 To tween size, scale the sprite or resize the sprite on the Stage. For more information, see "Resizing and scaling sprites" on page 71.

5 To define the beginning property settings, click the Sprite tab of the Property inspector and do any of the following:
   - To make the sprite fade in or out, enter a blend setting in the Property inspector (in List view). Enter 0 to make the sprite fade in or 100 to make it fade out. For more information, see “Setting blends” on page 76.
   - To tween rotation or skew, manually rotate or skew the sprite to the beginning position on the Stage or enter an angle in the Property inspector. For more information, see “Rotating and skewing sprites” on page 72.
   - To tween color, use the color boxes in the Property inspector to open the color palette for foreground and background color, or enter the RGB values for a new color in the boxes at the right (List view) or left (Graphical view).

6 In the Score, select the end frame of the sprite and select Insert > Keyframe.

   The end frame is not a keyframe unless you insert one there.

7 Make sure only the keyframe is selected (not the entire sprite), and then enter the ending values of the sprite properties you are tweening.

   For example, if you entered a blend setting of 0 in the first frame, you could enter a blend setting of 100 in this frame.

8 If necessary, create additional keyframes in the sprite and enter new values for the tweened properties.

9 To make the property changes defined by a keyframe occur at a different time, drag a keyframe in the Score to a new frame within the sprite.

10 To view the tweening, rewind and play the movie.

   Director gradually changes the value of the tweened property in the frames between the keyframes.

Suggestions and shortcuts for tweening

Follow the suggestions listed here to improve results and productivity while tweening sprites.

- For smoother movements, tween across more frames, increasing the tempo if necessary.
- To achieve some types of motion, you may need to split the sprite and tween the sprites separately. For more information, see “Accelerating and decelerating sprites” on page 85.
- To quickly make duplicates, Alt-drag (Windows) or Option-drag (Macintosh) keyframes. This technique is useful when you want the start and end frames to have the same settings. This shortcut also provides a quick way to create a complex path. Insert a single keyframe, drag several duplicates to the proper frames, and then select the various keyframes and set positions on the Stage.
- To extend the sprite and leave the last keyframe in place, Alt-drag (Windows) or Option-drag (Macintosh) a keyframe at the end of a sprite.
- To move many keyframe positions at once, Control-click (Windows) or Command-click (Macintosh) multiple keyframes to select them, and then move the sprite on the Stage.
- To make the animation look smoother, use an image editor to blur the edges of bitmaps.
- When tweening sprites that have a series of cast members, consider using a film loop instead. For more information, see “Using film loops” on page 94.
- To make a sprite jump instantly between settings in different keyframes, turn off all tweening options.

Changing tweening settings

To change tweening properties for sprites, you use the Sprite Tweening dialog box. You can turn tweening on and off for certain properties and control the curve of a tweening path and the way the speed changes as a sprite moves. For information about creating tweened animation, see "Tweening the path of a sprite" on page 84.

To change tweening settings:

1 Select a tweened sprite on the Stage or in the Score.

2 Select Modify > Sprite > Tweening to open the Sprite Tweening dialog box:

The diagram in the upper left corner of the Sprite Tweening dialog box shows the sprite's path as specified by the Curvature, Speed, Ease-In, and Ease-Out settings. This doesn't show the actual path of the sprite, just the type of curve it follows.

If the start and end points of the sprite are the same, the diagram is circular, indicating that the sprite travels in a circle when tweened. If the start and end points are not the same, the diagram describes a curved path, indicating that the sprite ends at a point different from the starting point.

3 To change which properties of the sprite are tweened, change the values for Tween.

A check mark indicates that the property will be tweened. The available properties are Path, Size, Rotation, Skew, Foreground Color, Background Color, and Blend.

4 To change how the sprite curves between positions defined by keyframes, adjust the Curvature slider.

Linear makes the sprite move in a straight line between the keyframe positions.

Normal makes the sprite follow a curved path inside the keyframe positions.

Extreme makes the sprite follow a curved path outside the keyframe positions.

5 To make the sprite move smoothly through start and end frames when it moves in a closed path, select Continuous at Endpoints.

6 To define how the tweened sprite positions change between keyframes, select an option for Speed. For more information, see "Accelerating and decelerating sprites" on page 85.

Sharp Changes makes the changes in position occur abruptly.

Smooth Changes makes the changes in position occur gradually.

7 To define how tweened sprite positions change over the whole length of the sprite, use the sliders to change the values for Ease-In and Ease-Out.

Ease-In defines the percentage of the sprite span through which the sprite accelerates.

Ease-Out defines the percentage of the sprite span through which the sprite decelerates.

Switching a sprite's cast members

To show different content while maintaining all other sprite properties, you exchange the cast member assigned to a sprite. This technique is useful when you have tweened a sprite and you decide to use a different cast member. When you exchange the cast member, the tweening path stays the same.

To exchange cast members in the Score:

1 To change a cast member in every frame, select an entire sprite. To change a cast member only in certain frames, select part of a sprite.

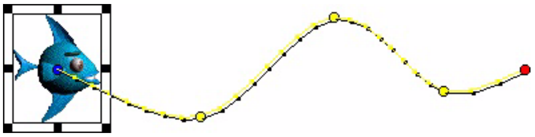
To select part of a sprite, press Alt and click the first frame that you want to select. Then press Control+Alt (Windows) or Option+Alt (Macintosh) and click each additional frame that you want to select.

2 Open the Cast window and select the cast member you want to use next in the animation.

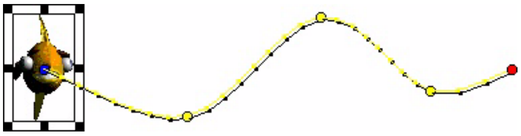
3 Do one of the following:

- Select Edit > Exchange Cast Members.
- Click the Exchange Cast Members button on the Director toolbar (Window > Toolbar).

If you selected an entire sprite, Director replaces the cast member for the entire sprite.



Before cast members are exchanged, the sprite moves like this.



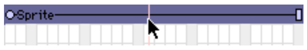
After cast members are exchanged, the sprite still moves in the same way, but it displays a different cast member.

You can also use Lingo or JavaScript syntax to switch the cast member assigned to a sprite. For more information, see "Assigning a cast member to a sprite with Lingo or JavaScript syntax" on page 80.

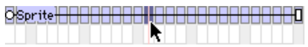
Editing sprite frames

To change how a sprite is selected and how keyframes are created, you use the Edit Sprite Frames option. Use this option with sprites that have animation that you need to adjust frequently; it is especially useful for cell animation in which each frame contains a different cast member in a different position.

Ordinarily, clicking a sprite on the Stage or in the Score selects the entire sprite.



When Edit Sprite Frames is turned on for a certain sprite, clicking the sprite selects a single frame. Any change you make to a tweenable property, such as moving a sprite on the Stage, defines a new keyframe.



To use Edit Sprite Frames, do one of the following:

- Select a sprite or sprites and select Edit > Edit Sprite Frame.
- Alt-double-click (Windows) or Option-double-click (Macintosh) a frame within the sprite.

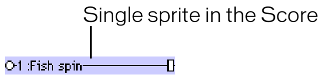
To return sprites to their normal state, do one of the following:

- Select sprites and select Edit > Edit Entire Sprite.
- Alt-double-click (Windows) or Option-double-click (Macintosh) a frame within the sprite.

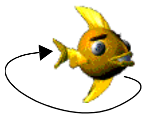
Frame-by-frame animation

To create animation that is more complex than is possible with simple tweening, you can use a series of cast members in frame-by-frame animation. Sprites usually refer to only one cast member, but they can refer to different cast members at different times during the life of the sprite.

For example, an animation of a man walking may display several cast members showing the man in different positions. By placing all the images in a sequence within a single sprite, you can work with the animation as if it were a single object.



A single sprite can display several cast members.



Sprite animating

Use this approach sparingly for movies that are downloaded from the Internet, because all cast members must be downloaded before the animation can run. As an alternative to this type of animation, consider using vector shapes, rotation and skewing on bitmap cast members, or Flash content. For more information, see Chapter 9, “Using Flash, Flash Components, and Other Interactive Media Types,” on page 181).

You can create multiple-cast-member animations in a variety of ways in Director. The following procedure explains a basic approach. The Cast to Time command provides an effective shortcut. For more information, see “Shortcuts for animating with multiple cast members” on page 92.

Note: The best way to prepare cast members for use in multiple-cast-member animation is with onion skinning in the Paint window. For more information, see “Using onion skinning” on page 125.

To animate a sprite with multiple cast members:

1 Create a sprite by placing the first cast member in the animation on the Stage in the appropriate frame.

2 Change the length of the sprite as needed.

Drag the start or end frame in the Score, or enter a new start or end frame number in the Sprite Inspector.

3 Select View > Display > Cast Member.

This setting displays the name of the cast member on each sprite. For more information, see "Displaying sprite labels in the Score" on page 62.

4 Select View > Sprite Labels > Changes Only.

This setting changes the view of the Score to show the name of each sprite's cast member when it changes. This makes it easy to identify frames where the cast member changes. For more information, see "Displaying sprite labels in the Score" on page 62. You may also want to zoom the score to 800% so the frames are wide enough to display the cast member information.

5 Select Edit > Edit Sprite Frames.

Edit Sprite Frames makes it easier to select frames within a sprite. For more information, see "Editing sprite frames" on page 89.

6 Select the frames in the sprite where you want a different cast member to appear.

7 Open the Cast window and select the cast member you want to use next in the animation.

8 Select Edit > Exchange Cast Members.

Director replaces the cast member in the selected frame with the cast member selected in the Cast window.

9 Repeat steps 6-8 to complete the animation. Select Edit > Edit Entire Sprite when you are done.

Sometimes a series of cast members placed in the Score jumps unexpectedly when you play the movie. This occurs because the cast members' registration points are not aligned properly. When you exchange cast members, Director places the new cast member's registration point precisely where the previous cast member's registration point was. By default, Director places registration points in the center of a bitmap cast member's bounding rectangle.

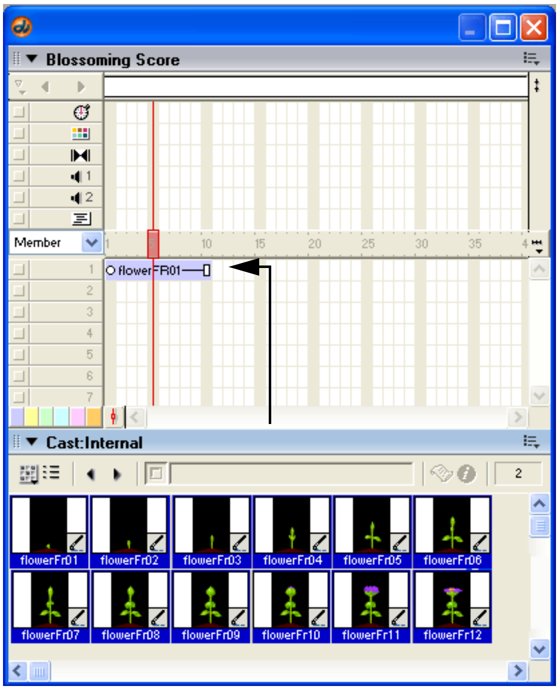
For information about aligning registration points, see "Changing registration points" on page 112. You can also align sprites relative to their bounding rectangles. For more information, see "Positioning sprites using guides, the grid, or the Align window" on page 66.

Shortcuts for animating with multiple cast members

The Cast to Time and Space to Time commands are both useful shortcuts for animating with multiple cast members.

Using the Cast to Time command

To move a series of cast members to the Score as a single sprite, you use the Cast to Time command (Modify > Cast to Time), which is one of the most useful methods for creating animation with multiple cast members. Typically, you create a series of images and then use Cast to Time to quickly place them in the Score as a single sprite. The Director onion skinning feature is also useful for creating and aligning a series of images for use in animation. For more information, see "Using onion skinning" on page 125.



Cast to Time places selected cast members in the Score as a single sprite.

To create a sprite from a sequence of cast members:

1 Select the frame in the Score where you want to place the new sprite.

2 Make the Cast window active.

3 Select the series of cast members to be placed in the new sprite.

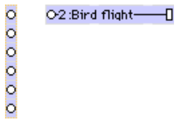
4 Select Modify > Cast to Time, or hold down Alt (Windows) or Option (Macintosh) and drag the cast members to the Stage.

The selected series of cast members becomes a single sprite.

Using the Space to Time command

To move sprites from adjacent channels to a single sprite, you use the Space to Time command (Modify > Space to Time). This method is convenient when you want to arrange several images on the Stage in one frame and then convert them to a single sprite.

Arrange sprites on the Stage in a single frame.



Space to Time converts sprites from adjacent channels to a single sprite.

Onion skinning provides a benefit in the Paint window similar to that provided by Space to Time on the Stage. For more information, see "Using onion skinning" on page 125.

To use the Space to Time command:

1 Select Edit > Preferences > Sprite and set Span Duration to 1 frame.

Set the span duration to any setting you like, but Space to Time works best with shorter sprites.

Note: If you are using a Macintosh OS X operating system, select the Director menu, instead of the Edit menu, to access Preferences.

2 Select an empty frame in the Score.

This is usually at the end of the Score.

3 Drag cast members onto the Stage to create sprites where you want them to appear in the animation.

As you position the sprites on the Stage, Director places each sprite in a separate channel. Make sure all the sprites are in consecutive channels.

4 Select all the sprites that are part of the sequence in the Score or on the Stage.

5 Select Modify > Space to Time.

The Space to Time dialog box appears.

6 Enter the number of frames you want between each cast member in the Separation text box

Director rearranges the sprites so that instead of being arranged from top to bottom in a single frame, they are arranged in sequence from left to right in a single sprite.

Note: Space to Time is a fast way to set up keyframes for a sprite to move along a curve. Arrange the cast members in one frame, select Modify > Space to Time, and add 10 to 20 cells between each cast member to produce a smooth curve.

Using film loops

A film loop is an animated sequence that you can use like a single cast member. For example, to create an animation of a bird flying across the Stage, you can create a film loop of the sequence of cast members that shows the bird flapping its wings. Instead of using the frame-by-frame technique, you create a sprite containing only the film loop and then animate it across as many frames as you need. When you run the animation, the bird flaps its wings and at the same time moves across the Stage.

You can also use film loops to consolidate Score data. Film loops are especially helpful when you want to reduce the number of sprite channels you are using. You can combine several Score channels into a film loop in a single channel.

To determine if a film loop is cropped or scaled within a sprite's bounding rectangle and to make the film loop repeat or mute its sounds, you use the Film Loop Cast Member properties. For more information, see "Setting film loop properties" on page 95.

Film loops are useful for animating repetitive motions and combining sprites to use fewer channels.

To create a film loop:

1 In the Score, select the sprites you want to turn into a film loop.

  Use sprites in as many channels as you need in film loops—even in the sound channel. Select sequences in all the channels you want to be part of the film loop. You can select sprite fragments if you first select a sprite and select Edit > Edit Sprite Frames. Control-click (Windows) or Command-click (Macintosh) to select sequences that are not in adjacent channels.

2 Select Insert > Film Loop.

  A dialog box appears asking you to name the film loop.

3 Enter a name for the film loop.

  Director stores all the Score data and cast member references as a new film loop cast member.

Note: You can drag a selection from the Score to the Cast window to quickly create a film loop cast member in that position.

A film loop behaves just like any other cast member, with a few exceptions:

- When you step through an animation that contains a film loop (either by using Step Forward or Step Backward or by dragging the playhead in the Score), the film loop doesn't animate. Animation occurs only when the movie is running.
- You can't apply ink effects to a film loop. If you want to use ink effects with a film loop, you need to apply them to the sprites that make up the animation before you turn the animation into a film loop.
- Lengthening or shortening a sprite that contains a film loop doesn't affect how fast the film loop plays. It changes the number of times the film loop cycles.

Director provides three other ways of incorporating a completed animation into a movie as a discrete element: you can export it as a digital video (QuickTime or AVI) or a DIB file (BMP), save and import it as a linked Director movie, or play it in a window in another Director movie.

Note: If you need to edit a film loop and you have deleted the original Score data that it was based on, it is possible to restore the Score data for editing. Copy the film loop cast member to the Clipboard, select a cell in the Score, and then paste. Director pastes the original Score data instead of the film loop.

Setting film loop properties

To determine if a film loop is cropped or scaled within a sprite's bounding rectangle and to make the film loop repeat or mute its sounds, you set properties for the film loop cast member.

To set film loop properties:

1. Select a film loop cast member.
2. To display the Property inspector, select Modify > Cast Member > Properties, or Window > Property Inspector.
3. If necessary, click the Member tab and display the Graphical view.

   The following noneditable settings are displayed:

   - The cast member size in kilobytes
   - The cast member creation and edit dates
   - The name of the last person who modified the cast member
4. To view or edit the cast member name, use the Name text box.
5. To add comments about the cast member, use the Comments text box.
6. To specify how Director removes the cast member from memory if memory is low, select one of the following options from the Unload pop-up menu:

   3–Normal sets the selected cast members to be removed from memory after all priority 2 cast members have been removed.

   2–Next sets the selected cast members to be among the first removed from memory.

   1–Last sets the selected cast members to be the last removed from memory.

   0–Never sets the selected cast members to be retained in memory; these cast members are never unloaded.
7. Click the Film Loop tab and display the Graphical view.
8. To determine how the film loop appears within the sprite bounding rectangle, select Framing options:

   Crop makes the movie image appear at its default size. Any portions that extend beyond the sprite's rectangle are not visible.

   Center is available only if Crop is selected. It determines whether transformations occur with the cast member centered within the sprite or with the cast member's upper left corner aligned with the sprite's upper left corner.

   Scale fits the movie inside the bounding rectangle.
9. To determine how the film loop plays back, use the following settings:

   Audio plays the sound portion of the film loop. Turn this option off to mute sounds.

   Loop replays the film loop continuously from the beginning to the end and back to the beginning.

Step-recording animation

Step recording is a process of animating one frame at time. You record the position of a sprite in a frame, step forward to the next frame, move the sprite to its new position, step forward to the next frame, and so on until you have completed the animation. This method is useful for creating sprites that follow irregular paths.

To step-record animation:

1 Place sprites on the Stage where you want the animation to begin.

2 Select all the sprites you want to animate.

3 In the Score, click the frame where you want animation to begin.

4 Select Control > Step Recording.

The step-recording indicator appears next to the channel numbers for the sprites being recorded, and the selection border widens.

5 Press 3 on the numeric keypad (make sure Number Lock is off) or click the Step Forward button in the Control panel.

Note: You can display the Control panel at the bottom of the Stage or in a floating panel.

The movie advances to the next frame. If you reach the last frame of a sprite, Director extends the sprites being recorded into the new frame.

Note: As soon as you move the animation in any way other than stepping—such as using Rewind, Play, or Back—recording stops.

6 Drag the sprite to reposition it.

You can also stretch the sprite, exchange cast members, or change any property.

7 Repeat steps 5 and 6 until you have completed the sequence you want to record.

8 Select Control > Step Recording again to stop recording.

You can also rewind the movie to stop recording.

Real-time recording animation

You can create animation by recording the movement of a sprite as you drag it across the Stage. The real-time recording technique is especially useful for simulating the movement of a pointer or for quickly creating a complex motion for later refinement.

For better control when you are recording in real time, use the Tempo control in the Control panel to record at a speed that is slower than normal.

Note: The Control panel attached to the bottom of the Stage doesn't include tempo settings. Tempo settings are available only from the floating panel version of the Control panel.

To use real-time recording:

1 Select one or more sprites on the Stage or in the Score.

   Recording begins at the playhead. It is best to select a sprite in a channel that contains no other sprites later in the movie.

   To record in a specific range of frames, select the frames, and then click the Selected Frames Only button in the Control panel.

2 Select Control > Real-Time Recording.

   The real-time recording indicator appears next to the channel numbers for the sprite being recorded, and a red and white selection frame appears around the sprite. Recording begins as soon as you drag the sprite on the Stage, so be prepared to move the mouse.

3 Drag the sprite on the Stage to record a path for the sprite.

   Director records the path.

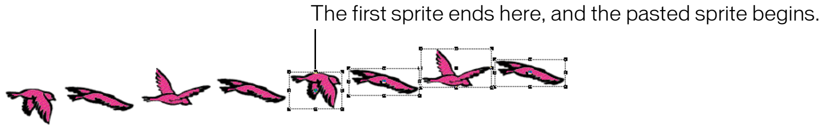
4 Release the mouse button to stop recording.

   The movie continues to play until you stop it.

Note: If you select Trails for the sprite, you can also use real-time recording to simulate handwriting.

Linking a sequence with Paste Relative

Paste Relative automatically aligns the start frame of one sprite with the end frame of the preceding sprite. It is useful for extending animations across the Stage.



To paste one sequence relative to another:

1 Select a sprite in the Score.

2 Select Edit > Copy Sprites.

3 Select the cell immediately after the last cell in the sprite.

4 Select Edit > Paste Special > Relative.

   Director positions the beginning of the pasted sprite where the previous sprite ends.

Repeat the process as many times as you need to create one continuous animation across the Stage.

Animating sprites with Lingo or JavaScript syntax

Lingo or JavaScript syntax can create animation regardless of the settings in the Score. This allows you to create or modify animation depending on movie conditions.

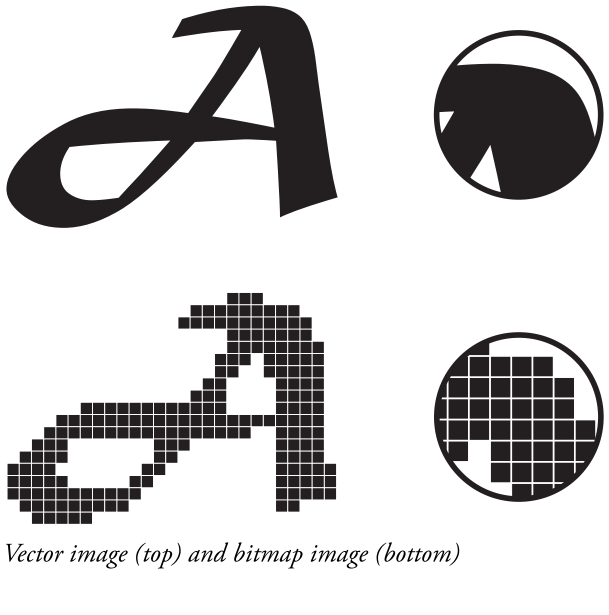
To move a sprite on the Stage, you use script that controls the sprite's location. For more information, see the Scripting Reference topics in the Director Help Panel.

To animate a sprite by switching the sprite's cast members, change the sprite's `member` property. For more information about this property, see the Scripting Reference topics in the Director Help Panel.

CHAPTER 5
Bitmaps

Bitmaps and vector shapes are the two main types of graphics used with Macromedia Director MX 2004. A bitmap defines an image as a grid of colored pixels, and it stores the color for each pixel in the image. A vector shape is a mathematical description of a geometric form that includes the thickness of the line, the fill color, and additional features of the line that can be expressed mathematically.

Bitmaps are suited for continuous tone images such as photographs. You can easily make minute changes to a bitmap by editing single pixels, but resizing the image can cause distortion as pixels are redistributed. Anti-aliasing is a Director feature that blends the bitmap's colors with background colors around the edges to make the edge appear smooth instead of jagged. A vector shape is most appropriate for a simple, smooth, clean-looking image. It typically includes less detail than a bitmap, but you can resize it without distortion.


Vector image (top) and bitmap image (bottom)

A bitmap typically requires more RAM and disk space than a comparable vector shape. If not compressed, bitmaps take longer than vector shapes to download from the Internet. Fortunately, Director offers compression control to reduce the size of bitmaps in movies that you package to play on the web. For more information about bitmap compression, see "Compressing bitmaps" on page 129. For more information about vector shapes, see Chapter 6, "Vector Shapes," on page 135.

You can create bitmaps in the Paint window or import them from any of the popular image editors in most of the popular formats, including GIF and JPEG. Director can also import bitmaps with alpha channel (transparency) data and animated GIFs. The Paint window includes a variety of tools for editing and applying effects to bitmaps.

About importing bitmaps

Importing bitmaps is similar to importing other types of media. If you import a bitmap with a color palette or depth different from that of the current movie, the Image Options dialog box appears. You must choose to import the bitmap at its original color depth or at the current system color depth. If you are importing an 8-bit image, you have the choice of importing the image's color palette or remapping the image to a palette that is already in Director. For more information, see"Choosing import image options" on page 45.

Director can import images with alpha channel (transparency) effects, which are 32 bits. If you reduce the image to a lower color depth, Director removes all the alpha channel data.

When importing bitmaps, you should always consider that they display on the screen at your monitor's resolution (generally 72 to 96 dots per inch). Higher-resolution images that you place on the Stage in Director might appear much larger than you expect. Other applications, particularly those focused on creating images for print, let you work on the screen with high-resolution images at reduced sizes. Within Director, you can scale high-resolution images to the right size, but this might reduce the quality of the image. Also, high-resolution images use extra memory and storage space, even after they've been scaled.

If you are working with a high-resolution image, convert it to between 72 and 96 dots per inch with your image-editing program before you import it into Director.

Director supports JPEG compression at runtime for internal cast members that are imported through the Standard or Include Original Data for Editing import options. A JPEG file that's imported with either of these options contains both the original compressed bits and decompressed bits. After it's imported, the JPEG file decompresses in the authoring environment. The cast member size displays the member's size in RAM after it's decompressed. The amount of RAM required to display a JPEG file is larger than its size on disk, so you can expect that the JPEG cast member size is larger in the Cast Properties window.

Director takes advantage of compressed JPEG data at runtime. The original compressed data bits are saved in Macromedia Shockwave content or a projector (if the Shockwave compression option is on). If you edit the member within Director in the Paint window, the compressed data is lost. An alert appears before the data is overwritten.

If the Shockwave compression option is on, Director also compresses bitmaps into the JPEG format. For more information about bitmap compression, see "Compressing bitmaps" on page 129.

Using animated GIFs

You can import an animated GIF into Director with File > Import, similar to the way in which you import any other bitmap cast member. The only difference is that when the Select Format dialog box appears, you select Animated GIF.

Director supports both the GIF89a and GIF87 formats. GIFs must have a global color table to be imported. You can import an animated GIF within a movie file or link to an external file. You also have the choice of importing the first frame of an animated GIF as a still image. As with an ordinary bitmap, you place an animated GIF in the Score in a sprite channel and extend it through all the frames in which you want it to appear. An animated GIF can play at the same frame rate as the Director movie, at a different rate that you specify, or at its original rate.

Director does not support the following inks for animated GIFs: Background Transparent, Reverse, Not Reverse, Darkest, Lightest, Add, Add Pin, Subtract, and Subtract Pin.

You can make an animated GIF play direct-to-Stage, meaning that it immediately appears on the Stage instead of being first composed in an offscreen buffer with other sprites. A direct-to-Stage GIF takes less time to load, but you can't place other sprites in front of it or use any ink effect.

To set properties for an animated GIF:

1 To specify how Director removes the cast member from memory if memory is low, select an option from the Unload pop-up menu on the Member tab of the Property inspector (Graphical view). For more information, see "Controlling cast member unloading" on page 47.

2 To achieve the fastest playback rate, click the Animated GIF tab and select DTS (Direct to Stage).

   When Direct to Stage is on, you can use only Copy ink and you can't place any sprites on top of the animated GIF sprite.

3 Select an option, described in the following list, from the Rate pop-up menu:

   Normal plays at the GIF's original rate, which is independent of the Director movie. The GIF can't exceed the Director frame rate.

   Fixed plays at the frame rate you enter in the FPS text box.

   Lock-Step plays at the same rate as the Director movie.

4 To set additional animated GIF settings, click More.

   - To change the file of a linked external cast member, enter a new pathname in the Import text box or click Browse to select a new file.
   - To import a file from the Internet, click Internet and enter a new URL in the File URL text box.

Using the Paint window

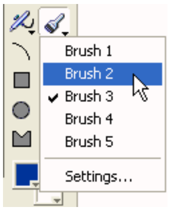
The Paint window has a complete set of paint tools and inks for creating and changing bitmap cast members for movies. Anything you draw in the Paint window becomes a cast member. When you make a change to a cast member in the Paint window, the image in the Cast window is instantly updated—as is the cast member wherever it appears on the Stage.

To open the Paint window, do any of the following:

- Select Window > Paint.
- Click the Paint window icon on the toolbar.
- Press Control-5 (Windows) or Command-5 (Macintosh).
- Double-click a bitmap sprite on the Stage or in the Score, or double-click the sprite's cast member in the Cast window.

Using Paint window tools and controls

If you see an arrow in the lower right corner of a tool, you can click it and hold down the mouse button to display a pop-up menu of options for that tool.



To select an irregular area, do one of the following:

- Click the Lasso tool in the Paint window, and drag to enclose the pixels you want to select.

  The Lasso selects only those pixels of a color that are different from the color the Lasso was on when you first started dragging it.
- Press Alt (Windows) or Option (Macintosh) while dragging to create a polygon selection. Every time you click, you create a new angle in the selection polygon.
- Click the Lasso tool, and hold down the mouse button to select new settings from the pop-up menu. For more information, see "Using the Lasso tool" on page 106.

To select a rectangular area, do one of the following:

- Click and drag the Marquee tool in the Paint window.
- Double-click the Marquee tool to select the entire bitmap.
- Click the Marquee tool, and hold down the mouse button to select new settings from the pop-up menu. For more information, see "Using the Marquee tool" on page 106.

To change the location of the registration point, do one of the following:

- Click the Registration Point tool, and click the spot where you want to set the registration point.
- Double-click the Registration Point tool to set the registration point in the center of the image. For more information, see "Changing registration points" on page 112.

To erase, do one of the following:

- Click and drag the Eraser tool to erase pixels.
- Double-click the Eraser tool to erase the cast member.

To move the view of the Paint window, do one of the following:

- Click and drag the Hand tool to move the visible portion of the image within the Paint window.
- Shift-drag to move straight horizontally or vertically.

  Press the Spacebar to temporarily activate this tool while using other paint tools.

To zoom in or out on an area:

- Click the Magnifying Glass tool, and click in the Paint window to zoom in. Shift-click to zoom out. For more information, see "Zooming in and out in the Paint window" on page 108.

To select a color in a cast member:

1 Click the Eyedropper tool.

2 Do one of the following:
   - Click a color to select it as the foreground color.
   - Shift-click a color to select it as the background color.
   - Alt-click (Windows) or Option-click (Macintosh) to select the destination color for a gradient.

Press the D key to temporarily activate the Eyedropper tool while using other paint tools.

To fill all adjacent pixels of the same color with the foreground color:

- Click the Bucket tool, and click the area you want to fill.
- To open the Gradient Settings dialog box, double-click the Bucket tool.

To enter bitmap text, do one of the following:

- Click the Text tool, then click in the Paint window, and begin typing.
- Select character formatting with the Modify > Font command.

Bitmap text is an image. Before you click outside the text box, you can edit text you've typed by using the Backspace key (Windows) or Delete key (Macintosh). After you click outside the text box, you can't edit or reformat bitmap text.

To draw a 1-pixel line in the current foreground:

- Click the Pencil tool, and drag it in the Paint window. To constrain the line to horizontal or vertical, Shift-Click and drag.

  If the foreground color is the same as the color underneath the pointer, the Pencil tool draws with the background color.

To spray variable dots of the foreground color:

- Click the Airbrush tool, and drag it in the Paint window.
- Click the Airbrush tool, and hold down the mouse button to select a new brush type from the pop-up menu. Select Settings to change the selected brush. For more information, see "Using the Airbrush tool" on page 107.

To brush strokes of the foreground color:

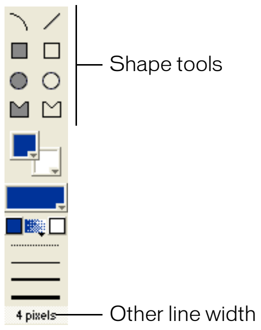
- Click the Brush tool, and drag it in the Paint window. To constrain the stroke to horizontal or vertical, Shift-click and drag.

To select a new brush type:

- Click the Brush tool, and hold down the mouse button to select a new brush type from the pop-up menu. Select Settings to change the selected brush. For more information, see "Using the Brush tool" on page 107.

To paint shapes or lines:

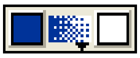
- Click and drag the shape tools. To constrain lines to horizontal or vertical, ovals to circles, and rectangles to squares, Shift-click and drag.



The filled tools create shapes that are filled with the foreground color and the current pattern. The thickness of lines is determined by the line-width selector.

To select a foreground and destination color for color-shifting inks:

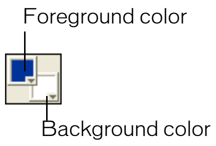
- Click the color box on the left to select a foreground color; click the color box on the right to select a destination color.



These colors affect the Gradient, Cycle, and Switch inks. Each of these inks uses a range of colors that shifts between the foreground color and the destination color. For more information, see "Using gradients" on page 117 and "Using Paint window inks" on page 121.

To select the foreground and background colors:

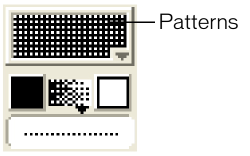
- Use the Foreground Color pop-up menu to select the primary fill color (used when the pattern is solid and the ink is Normal).



- Use the Background Color pop-up menu to select the secondary color (the background color in a pattern or text).

To select a pattern for the foreground color, do one of the following:

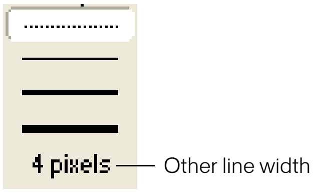
- To change the pattern palette, select Pattern Settings at the bottom of the Patterns pop-up menu.



- To define a tile—a pattern that is based on a rectangular section of an existing cast member—select Tile Settings from the Patterns pop-up menu. For more information, see "Editing patterns" on page 120 and "Creating a custom tile" on page 120.

To select a line thickness, do one of the following:

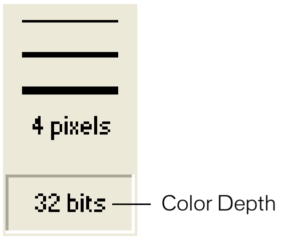
- Click the None, One-, Two-, or Three-Pixel Line button.



- Double-click the Other Line Width button to open the Paint Window Preferences dialog box, and assign a width to the line.

To change the color depth of the current cast member:

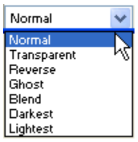
- Double-click the Color Depth button to open the Transform Bitmap dialog box.



The button displays the color depth of the current cast member. For more information, see "Changing size, color depth, and color palette for bitmaps" on page 113.

To select a Paint window ink:

- Select the type of ink from the Ink pop-up menu at the lower left of the window.



For more information, see "Using Paint window inks" on page 121.

Using the Lasso tool

You use the Lasso tool to select irregular areas or polygons. After you select artwork, it can be dragged, cut, copied, cleared, or modified with the effects on the Paint toolbar. The Lasso tool selects only those pixels of a color that are different from the color the Lasso tool was on when you first started dragging it. You use the Lasso pop-up menu to change settings.

To select an irregular area with the Lasso tool:

- Drag with the Lasso tool to enclose the pixels you want to select.

To select a polygon area with the Lasso tool:

1 Press Alt (Windows) or Option (Macintosh) while clicking the first point.

2 Click the remaining points.

3 Double-click the last point.

To change Lasso tool settings:

1 Hold down the mouse button while the pointer is on the Lasso tool.

2 Select one of the following options from the Lasso pop-up menu:

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

No shrink lets you select the entire area you drag around. The lasso selects whatever is inside the selected area.

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

Using the Marquee tool

The Marquee tool selects artwork in the Paint window. After you select artwork, it can be dragged, cut, copied, cleared, or modified with the commands on the Paint toolbar. Use the Marquee pop-up menu to change settings.

To select with the Marquee tool:

- Drag to select a rectangular area.

To select the entire bitmap:

- Double-click the Marquee tool.

To stretch or compress art that is selected with the Marquee tool:

- Hold down Control (Windows) or Command (Macintosh) while dragging a border of the selected area. Hold down Shift at the same time to maintain proportions.

To move a selection, do any of the following:

- Click and drag the selection.
- Shift-drag to limit the movement to horizontal or vertical.
- Use the arrow keys to move the selection one pixel at a time.

To make a copy of artwork that is selected with the Marquee tool:

- Hold down Alt (Windows) or Option (Macintosh) while dragging the selection.

To change marquee settings:

- Click the Marquee tool, hold down the mouse button, and select from the following options:

  Shrink causes the rectangle to shrink around the selected artwork.

  No Shrink lets you select everything within the marquee.

  Lasso tightens the marquee around the object in the same way as the Lasso tool and selects the pixels according to the color of the pixel beneath the cross hair when you started the drag.

  See Thru Lasso modifies the selection function so that pixels with the same color as the first pixel selected are not included in the selection.

Using the Airbrush tool

The Airbrush tool sprays the currently selected color, ink, and fill pattern. To modify the spray, you select the ink effects from the Ink pop-up menu in the Paint window. The longer you hold the Airbrush tool in one spot, the more it fills in the area.

If you hold down the mouse button when the pointer is positioned on the Airbrush tool, the Airbrush 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 Airbrush Settings dialog box.

To use the Airbrush tool:

- Click the Airbrush tool, and drag it in the Paint window.

To define airbrush settings:

1 Click the Airbrush tool, and hold down the mouse button.

2 Select the menu item for which you want to define settings.

3 Open the menu again, and select Settings from the Airbrush pop-up menu. Enter values for the options in the Airbrush Settings dialog box.

You can also double-click the Airbrush tool to open the Airbrush Settings dialog box.

4 Use the Flow Rate slider to control how quickly the airbrush covers an area with paint. To change the flow, drag the Flow Speed slider.

5 Use the Spray Area slider to set the size of the airbrush's spray area.

6 Use the Dot Size slider to set the size of the dots that the airbrush sprays.

7 Use the following Dot Options to specify how dots spray from the airbrush:

Uniform Spray causes the airbrush to spray drops of uniform size.

Random Sizes sprays randomly sized drops.

Current Brush sprays drops shaped the same as the current airbrush.

Using the Brush tool

You use the Brush tool to brush strokes with the current color, ink, and fill pattern. To select a different size and brush shape, you use the Brush Settings dialog box. The selections you make in the Brush Settings dialog box are assigned to the menu item in the pop-up menu and remain in effect until you change them. 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 Brush Settings dialog box.

To use the Brush tool:

- Click the Brush tool, and drag it in the Paint window.

To change brush settings:

1 Click the Brush tool, and hold down the mouse button.

2 Select the menu item for which you want to define settings.

3 Open the menu again, and select Settings from the Brush pop-up menu. Enter values for the options in the Brush Settings dialog box.

You can also double-click the Brush tool to open the Brush Settings dialog box.

4 To select from the default brush shapes, select Standard from the pop-up menu, and click a brush shape in the chart below the pop-up menu.

5 To create a new brush shape, select Custom from the pop-up menu, and select the brush shape you want to modify from the chart below the pop-up menu.

6 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. Clicking outside the Brush Shapes dialog box places the pixels on the screen at the point you click. Use the following editing controls to change the brush shape:

The right and left arrows move the brush shape one pixel to the right or left.

The up and down arrows move the brush shape up or down one pixel.

The black and white square reverses the colors of the brush shape (for example, black becomes white and white becomes black).

Copy copies the brush shape to the Clipboard.

Paste pastes the brush into the custom set of brush shapes.

Using rulers in the Paint window

The Paint window has vertical and horizontal rulers to help you align and size your artwork.

To hide or show the rulers in the Paint window:

- Select View > Rulers.

To change the location of the zero point, do one of the following:

- Drag along the ruler at the top or side of the window.
- Drag into the window to align the zero point with a specific point in the cast member.

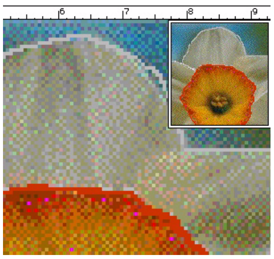
Zooming in and out in the Paint window

You can use the Magnify tool or the Zoom commands on the View menu to zoom in or out at four levels of magnification.

To zoom in or out, do one of the following:

- Click the Magnify tool, and click the image. Click again to increase the magnification. Shift-click to zoom out.
- Select Zoom In or Zoom Out from the Paint window's Options menu.
- Select View > Zoom, and select the level of magnification.
- Press Control + the Plus (+) key (Windows) or Command + the Plus (+) key (Macintosh) to zoom in, or Control + the Minus (-) key (Windows) or Command + the Minus (-) key (Macintosh) to zoom out.

- Control-click (Windows) or Command-click (Macintosh) the image to zoom in on a particular place.



To return to normal view, do one of the following:

- Click the normal-sized image in the upper right corner.
- Select View > Zoom > 100%.

Changing selected areas of a bitmap

After you select part of an image in the Paint window with the Lasso or Marquee tool, you can change the selected area.

To reposition the selected area:

1 Move the cross hair inside the selected area (the cross hair becomes an arrow pointer).

2 Drag the selected area.

To affect how the selected area behaves when you drag it, use the following key combinations:

- To make a copy of the selected area as you drag, Alt-drag (Windows) or Option-drag (Macintosh) the selection.
- To stretch the selection (Marquee tool only), Control-drag (Windows) or Command-drag (Macintosh) the selection.
- To stretch the selection proportionally (Marquee tool only), Control-Shift-drag (Windows) or Command-Shift-drag (Macintosh) the selection.
- To copy and stretch the selection (Marquee tool only), Control-Alt-drag (Windows) or Command-Option-drag (Macintosh) the selection.
- To constrain the movement of the selection to horizontal or vertical, Shift-drag the selection.
- To move the selection one pixel at a time, use the arrow keys.

Flipping, rotating, and applying effects to bitmaps

The toolbar at the top of the Paint window contains buttons to apply effects to bitmaps. Before using any of these options, you must select part of the bitmap with the Lasso or Marquee tool. Effects that change the shape of the selection work only when the selection is made with the Marquee tool. Effects that change colors within the selection work with the Marquee and the Lasso tools.

Lingo or JavaScript syntax can flip and rotate bitmaps by flipping and rotating bitmap sprites. For more information, see “Rotating and skewing sprites” on page 72 and “Flipping sprites” on page 75.

Note: To repeat any of these effects after using them, press Control+Y (Windows) or Command+Y (Macintosh).

To flip, rotate, skew, or apply effects to part of a bitmap:

1 Select part of a bitmap in the Paint window with the Marquee tool.

2 Use any of the following effects:

- To flip the selection, click the Flip Horizontal button to flip right to left, or click the Flip Vertical button to flip top to bottom.

- To rotate the selection 90° counterclockwise or 90° clockwise, click the Rotate Left or Rotate Right buttons, respectively.

- To rotate the selection by any amount in either direction, click the Free Rotate button, and drag the rotate handles in any direction. (You can rotate a sprite that contains a bitmap instead of the bitmap. For more information, see "Rotating and skewing sprites" on page 72.)

- To skew the selection, click the Skew button, and drag any of the skew handles.

- To warp the shape of the selected area, click the Warp button, and drag any handle in any direction.

- To create a perspective effect, click the Perspective button, and drag one or more handles to create the effect you want.

- To create an outline around the edges of the selected artwork, click the Trace Edges button.

To apply color effects to a selected area:

1 Select an area within a bitmap cast member by using either the Marquee or the Lasso tool.

2 Use any of the following effects:

- To soften the edges of the selected artwork, click the Smooth button. This effect works only with 8-bit cast members.

- To reverse the colors of the selected area, click the Invert button.

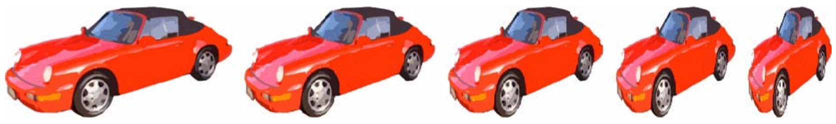
- To increase or reduce the brightness of the selected area, click the Lighten Color or Darken Color button. This effect works on 8-bit (256 color) images only.

- To fill the selected area with the current foreground color and pattern, click the Fill button.

- To change all pixels of the foreground color within the selection to the currently selected destination color, click the Switch Colors button.

Using Auto Distort

You can use Auto Distort to create animations that show bitmap cast members gradually changing from frame to frame. Auto Distort generates intermediate cast members for any cast member that is free-rotated, made into a perspective, slanted, distorted, or skewed.



Cast members created with Auto Distort after using the perspective effect.

To use Auto Distort:

1 Select the portion of a bitmap cast member you want to change.

2 Use the Free Rotate, Perspective, Skew, Distort, or Stretch button to change the image.

3 Without deselecting the changed image, select Xtras > Auto Distort.

4 In the Auto Distort dialog box, enter the number of cast members to create and click the Begin button.

Director generates new cast members with an intermediate amount of change applied to each one. The new cast members appear in the first available cast positions.

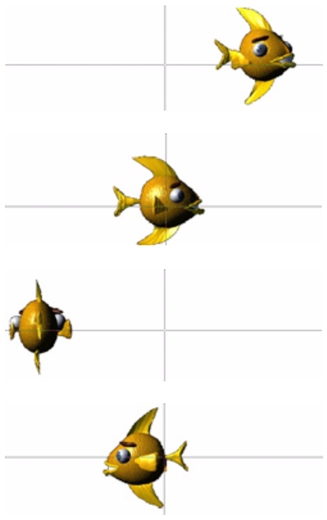
Changing registration points

A registration point is a marker that appears on a sprite when you select it with your mouse. (Registration points don't appear on unselected sprites or when a movie is playing.) Registration points provide a fixed reference point within an image, thereby helping you align sprites and control them from Lingo or JavaScript syntax. Registration points are crucial to precisely placing vector shapes, bitmaps, and all cast members that appear on the Stage.

By default, Director assigns a registration point in the center of all bitmaps, but for many types of animation, you might want to move the registration point. To do this, you can use the Registration Point tool.

You can edit a bitmap's registration point in the Paint window or using Lingo or JavaScript syntax.

Moving the registration point is useful for preparing a series of images for animation. When you use Cast to Time or exchange cast members, Director places a new cast member's registration point precisely where the previous one was located. By placing the registration point in the different locations, you can make a series of images move around a fixed position without having to manually place the sprites on the Stage. Use onion skinning to set registration points when images are placed in relation to each other. For more information, see "Using onion skinning" on page 125.



With the registration points set as shown, the series of fish swim in a circle without any tweening or manual placement of sprites.

To set a registration point:

1 Display the cast member you want to change in the Paint window.

2 Click the Registration Point 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 cross hair 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.

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

To set a bitmap's registration point with Lingo or JavaScript syntax:

- Set the `regPoint` cast member property. Set the `centerRegPoint` property to specify whether Director automatically centers the registration point if the bitmap is edited. For example, you can set the regPoint cast member property using the following:

  Lingo:

  ```
  member("bitmapMemberName").regPoint = point(10,10)
  ```

  JavaScript:

  ```
  member("bitmapMemberName").regPoint = point(10,10);
  ```

For more information, see the Scripting Reference topics in the Director Help Panel.

Changing size, color depth, and color palette for bitmaps

You can use Transform Bitmap to change the size, color depth, and palette of selected cast members. Any change you make to a cast member's color depth or palette affects the cast member itself—not only its appearance on the Stage. You can't undo changes to the color depth and palette. If you want to keep a cast member's original bitmap unchanged but temporarily apply a different palette, use the Member tab in the cast member's Property inspector. To change the size of only the sprite on the Stage, use the Sprite tab in the sprite's Property inspector.

You can also remap images to new palettes with an image-editing program such as Macromedia Fireworks.

The Transform Bitmap dialog box displays values for the current selection. If you select more than one cast member, a blank value indicates that cast members in the selection have different values. To maintain a cast member's original value, leave that value blank in the dialog box.

To use Transform Bitmap:

1 Select the bitmap cast members to change.

2 Select Modify > Transform Bitmap.

3 To change the size of the bitmap, do one of the following:

If multiple cast members are selected, you can resize all the cast members to the dimensions you enter.

- Enter new measurements (in pixels) in the Width and Height text boxes.
- Enter a scaling percentage in the Scale text box.

Select Maintain Proportions to keep the width and height of the selected cast member in proportion. If you change the width, the proportional height is automatically entered in the Height text box. If you use Transform Bitmap to change several cast members at once, be sure to deselect Maintain Proportions. If you don't, all cast members are resized to the values in the Width and Height text boxes.

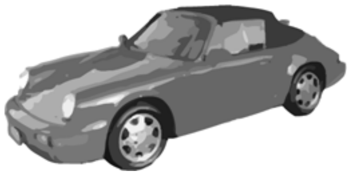
4 To change the color depth, select an option from the Color Depth pop-up menu.

For more information about the color depth of bitmap cast members, see "Controlling color" on page 145.

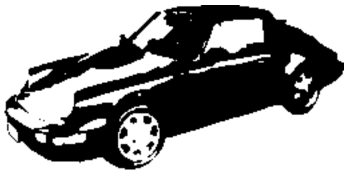
5 To change the palette, select a palette from the Palette pop-up menu and select one of the following remapping options:

Remap Colors replaces the original colors in the graphic with the most similar solid colors in the new palette. This is the preferred option in most cases.

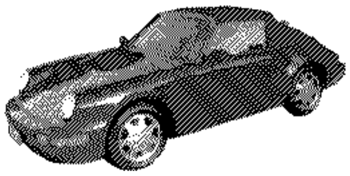
Dither blends the colors in the new palette to approximate the original colors in the graphic.



256 grays



Remapped to closest colors in black and white



Dithered in black and white

6 Click Transform to execute the changes.

The settings you select in the Transform Bitmap dialog box can't be undone.

Controlling bitmap images with Lingo or JavaScript syntax

Lingo or JavaScript syntax lets you control bitmap images in two ways. First, you can perform simple operations that affect the content of entire image cast members. These operations include changing the background and foreground colors in addition to switching the image that appears in a specific cast member with that of another cast member. Each of these operations manipulates a property of the entire image cast member.

Second, you can use Lingo or JavaScript syntax to perform fine manipulations of the pixels of an image or to create entirely new images. When you use script, you can be extremely flexible about which images you display. You can create images based on dynamic information, such as user input, or based on any other factors you want to include. To perform this kind of image operation, Lingo or JavaScript syntax works with image objects. For more information, see "Creating image objects" on page 115.

To change the image assigned to a bitmap cast member:

- Set the `picture` cast member property. For more information about this property, see the Scripting Reference topics in the Director Help Panel.

To specify the background or foreground of a bitmap sprite:

- Set the `backColor` or `foreColor` sprite property. For more information about these properties, see the Scripting Reference topics in the Director Help Panel.

To capture the current graphic contents of the Stage:

- Set a bitmap's `picture` cast member property to the Stage's `picture` property. For more information about these properties, see the Scripting Reference topics in the Director Help Panel.

  For example, the `member("Archive").picture = (the stage).picture` statement makes the current image of the Stage the image for the bitmap cast member Archive.

Creating image objects

An image object can be either a self-contained set of image data or a reference to the image data of a cast member or of the Stage. If an image object is created by referring to a cast member, the object contains a reference to the image of the member. The following statement creates an image object that contains a reference to the image of the cast member called Boat.

```
myImage = member("Boat").image
```

Because the image object `myImage` contains a reference to the cast member Boat, any changes you make to the object are reflected in the cast member. These changes are also reflected in any sprites made from that cast member.

You can also create an image object that contains a reference to the graphic contents of the Stage:

```
myImage = window("stage").image
```

Any changes to this image object are reflected on the Stage.

- To create an image object that is a self-contained set of image data instead of a reference to a cast member, you must tell the script what kind of image you want to create. You must provide the parameters that describe the size and bit depth of the image you are creating.

The following statement creates an image object that contains a 640 x 480 pixel, 16-bit image:

```
myImage = image(640, 480, 16)
```

Editing image objects

After you create an image object, its data can be edited with a variety of scripting commands that are designed to manipulate the pixels of the image. You can crop images, draw new pixels on them, copy sections of them, and work with mask and alpha channel information. For more information, see the Scripting Reference topics in the Director Help Panel.

To draw a line on an image object:

- Use the `draw()` method. You must specify the locations of each end of the line in addition to the line's color.

The following statement draws a line on the previously created 640 x 480 image object `myImage`, running from 20 pixels inside the upper left corner to 20 pixels inside the lower right corner, and colors it blue:

```
myImage.draw(20, 20, 620, 460, rgb(0, 0, 255))
```

To draw a rectangle on an image object:

- Use the `fill()` method. You provide the same information as for the `draw` method, but Director draws a rectangle instead of a line.

The following statement draws a red 40 x 40 pixel rectangle near the upper left corner of the image object `myImage`:

```
myImage.fill(rect(20, 20, 60, 60), rgb(255, 0, 0))
```

To determine the color of an individual pixel of an image object or set that pixel's color:

- Use the `getPixel` or `setPixel` method.

To copy part or all of an image object into a different image object:

- Use the `copyPixels()` method, which requires you to specify the image from which you are copying, the rectangle to which you are copying the pixels, and the rectangle from which to copy the pixels in the source image.

The following statement copies a 40 x 40 rectangle from the upper left area of the image object `myImage` and puts the pixels into a 40 x 40 rectangle at the lower right of the 300 x 300 pixel object called `myNewImage`:

```
myNewImage.copyPixels(myImage, rect(260, 260, 300, 300), rect(0, 0, 40, 40))
```

When using `copyPixels()`, you can specify optional parameters that tell the script to modify the pixels you're copying before drawing them into the destination rectangle. You can apply blends and inks, change the foreground or background colors, specify masking operations, and more. You specify these operations by adding a property list at the end of the `copyPixels()` method.

The following statement performs the same operation as the previous example and directs the script to use the Reverse ink when rendering the pixels into the destination rectangle:

```
myNewImage.copyPixels(myImage, rect(260, 260, 300, 300), rect(0, 0, 40, 40),
  [#ink: #reverse])
```

To make a new image object from the alpha channel information of a 32-bit image object:

- Use the `extractAlpha()` method, which can be useful for preserving the alpha channel information of a 32-bit image object that you plan to reduce to a lower bit depth. Reducing the bit depth can delete the alpha information.

The following statement creates a new image object called `alphaImage` from the alpha channel information of the 32-bit image object called `myImage`:

```
alphaImage = myImage.extractAlpha()
```

There are many more image-editing operations available through Lingo or JavaScript syntax. For a complete list, see the Scripting Reference topics in the Director Help Panel.

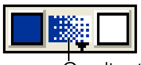
Using gradients

Director can create gradients in the Paint window. You can use gradients with the Brush tool, the Bucket tool, the Text tool, or any of the filled shape tools. Typically, a gradient consists of a foreground color at one side (or the center) of an image and another color, the destination color, at the other side (or outside edge) of the image. Between the foreground and destination colors, Director creates a blend of the two colors.

To use a gradient:

1 Select the Brush tool, the Bucket tool, or one of the filled shape tools.

2 Select the type of gradient from the Gradient pop-up menu.

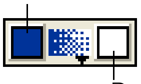


Selecting a gradient type automatically sets the current Paint window ink to Gradient. You can also select Gradient ink from the Ink pop-up menu at the bottom left of the Paint window to create a gradient with all the current settings.

To manually specify a gradient, select Gradient Setting from the pop-up menu. For more information, see “Editing gradients” on page 117.

3 Select a foreground color from Gradient Colors pop-up menu on the left.

The foreground color is the same color that is specified for the Paint window.



4 Select a destination color from the Gradient Colors pop-up menu on the right.

The destination color is the color of the gradient when it completes the color transition.

5 Use the current tool in the Paint window.

Director uses the gradient you’ve defined to fill the image.

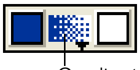
6 To stop using a gradient, select Normal from the Ink pop-up menu. For more information, see “Using Paint window inks” on page 121.

Editing gradients

You can change gradients before using them by changing the settings in the Gradient Settings dialog box. In the Gradient Settings dialog box, you set the foreground and background colors as well as the pattern to use with your gradient. There are several pop-up menus that control the style of your gradient fill. Each choice you make is immediately previewed on the left.

To edit gradient settings:

1 Select Gradient Settings from the Gradient Colors pop-up menu.



2 To determine whether the gradient is created with the pattern you select with the Patterns pop-up menu in the Paint window or with a dithered pattern, select a Type option, as described in the following list:

Dither produces a smooth transition between colors. If you select Dither, only dithering options appear in the Method pop-up menu.

Pattern uses the current pattern for the color transition. If you select Pattern, only pattern options appear in the Method pop-up menu.

3 To determine how a gradient shifts between colors, select an option from the Method pop-up menu:

If you select Dither as the Type option, the following choices are available:

Best Colors ignores the order of the colors in the palette. Instead, it uses only colors that create a continuous blend from foreground to background colors and blends them with a dithered pattern. Dithering is a technique that creates color from two or more colors of pixels interspersed together.

Adjacent Colors uses all colors between the foreground and background colors and blends them with a dithered pattern.

Two Colors uses only the foreground and background colors and blends them with a dithered pattern.

One Color uses only the foreground color and fades it with a dithered pattern.

Standard Colors ignores all colors between the foreground and background colors and adds several blended colors with a dithered pattern to create the gradient.

Multi Colors ignores all the colors between the foreground and background colors and adds several blended colors with a randomized dithered pattern to create a smooth gradient.

If you select Pattern as the Type option, the following options are available:

Best Colors ignores the order of the colors in the palette and uses only colors that create a continuous blend of the foreground and background colors.

Best Colors Transparent ignores the order of the colors in the palette and uses only colors that create a continuous blend of the foreground and background colors. White pixels in patterns created with this method are transparent.

Adjacent Colors uses all the colors in the palette between the foreground and background colors for the gradient.

Adjacent Colors Transparent uses all the colors in the palette between the foreground and background colors for the gradient. White pixels in patterns that are created with this method are transparent.

4 To determine the way the gradient fills an area in the Paint window, select one of the following options from the Direction pop-up menu:

Top to Bottom puts the foreground color at the top and the destination color at the bottom.

Bottom to Top puts the foreground color at the bottom and the destination color at the top.

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. You set the direction of the gradient in the Paint window with the paint tool used to fill the area.

Shape Burst creates a gradient that starts at the edge of the area and moves toward the center. The foreground color begins at the edge and the destination color appears in the center. This option works only on the Macintosh.

Sun Burst begins with the foreground color at the edge of the area and moves in concentric circles to the destination color at the center.

5 To control how colors cycle in a gradient, select a Cycles option, as described in the following list:

Sharp cycles have a banded appearance; smooth cycles go from foreground to destination and then back to foreground.

One cycles the gradient once through the range of colors you define.

Two Sharp cycles through the range of colors from foreground to destination twice.

Two Smooth cycles the gradient from foreground to destination and then from destination to foreground.

Three Sharp cycles the gradient from foreground to destination three times.

Three Smooth cycles the gradient from foreground to destination, destination to foreground, and foreground to destination.

Four Sharp cycles the gradient from foreground to destination four times.

Four Smooth cycles the gradient from foreground to destination, destination to foreground, foreground to destination, and destination to foreground.

6 To select how colors are distributed between the foreground and destination colors of the gradient, select a Spread option, as described in the following list:

Equal provides even spacing of colors between the foreground and 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.

7 To determine whether the full range of the gradient is created over the paint object, the cast member, or the entire Paint window, select a Range option, as described in the following list:

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 within 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 that corresponds to the object's location in the window.

8 To select a foreground, background, or destination color for the gradient, use the appropriate color picker.

The foreground color is the starting color of the gradient, and the destination color is the ending color. Background color has no effect unless you are using a pattern.

9 To select a pattern, use the Patterns pop-up menu.

Using patterns

You can select among three sets of patterns that are included with Director or create custom patterns. The patterns you change or edit in the Paint window don't affect the patterns that are available for shapes.

To use a pattern:

1 Select the Brush tool, the Bucket tool, or one of the filled shape tools.

2 Select the type of pattern from the Patterns pop-up menu.

To manually specify a pattern, select Pattern Settings from the pop-up menu. For more information, see "Editing patterns" on page 120.

Editing patterns

You can change patterns before using them by changing the settings in the Pattern Settings dialog box. Each change you make is immediately previewed.

To select a new set of patterns or create a custom pattern:

1 Select Pattern Settings from the bottom of the Patterns pop-up menu.

2 Select an option from the pop-up menu at the top of the Pattern Settings dialog box:

- To select one of the standard, noneditable sets of patterns, select QuickDraw, Grays, or Standard.
- To edit a pattern, select Custom. Custom is an editable copy of the Standard palette set.

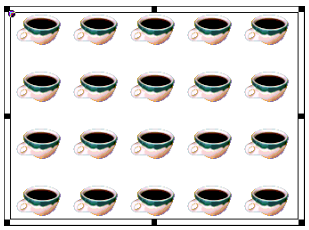
3 Select the pattern to edit or use the Copy and Paste buttons to move an existing pattern to one of the empty tile positions.

4 Use any of the following methods to edit the pattern:

- Click the magnified image of the pattern. Click a blank pixel to fill it, and click a filled pixel to make it blank.
- Click the right, left, up, and down arrows to move the pattern one pixel in any direction.
- Click the Black and White square to reverse the colors of the pattern (for example, black becomes white, and white becomes black).

Creating a custom tile

Custom tiles provide an effective way of filling a large area with interesting content without using much memory or increasing the downloading time. They are especially useful for large movies on the web. A custom tile uses the same amount of memory no matter what size area it fills.



To create a custom tile:

1 Create a bitmap cast member to use as a tile, and display it in the Paint window.

2 Click the pattern box in the Paint window, and select Tile Setting from the bottom of the Patterns pop-up menu.

3 Click an existing tile position to edit.

The existing tiles appear next to the Edit label. You must replace one of the built-in tiles to create a new one. To restore the built-in tile for any tile position, select it, and click Built-in.

4 Click Cast Member.

The cast member appears in the box at the lower left. The box at the right shows how the image appears when it is tiled. The dotted rectangle inside the cast member image shows the area of the tile.

To select a different cast member for the tile, use the arrow buttons to the right of the Cast Member button to move through the movie's cast members.

5 Drag the dotted rectangle to the area of the cast member you want tiled.

6 Use the Width and Height controls to specify the size of the tile.

The new tile appears in the tile position you selected. You can use it in the Paint window or from the Tool palette to fill shapes.

Using Paint window inks

You can use Paint window inks to create color effects for bitmap cast members. Paint window inks are different from sprite inks, which affect entire sprites and don't change cast members.

You can select an ink effect from the Ink pop-up menu at the bottom of the Paint window.

The result of the ink you select depends on whether you are working in color or in black and white. Also, some inks work better when painting with patterns, and others work better when painting with solid colors.

Ink	B&W	Color	Works with
Normal	✔	✔	Solids and patterns
Transparent	✔	✔	Patterns
Reverse	✔	✔	Solids and patterns
Ghost	✔	✔	Solids (B&W) and patterns (color)
Gradient	✔	✔	Brush, Bucket, shape tools
Reveal	✔	✔	Brush, shape tools
Cycle		✔	Solids and patterns
Switch		✔	Brush
Blend		✔	Solids and patterns
Darkest		✔	Patterns
Lightest		✔	Patterns
Darken		✔	Brush

Ink	B&W	Color	Works with
Lighten		✔	Brush
Smooth		✔	Brush
Smear		✔	Brush
Smudge		✔	Brush
Spread	✔	✔	Brush
Clipboard	✔	✔	Brush

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

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

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

Gradient lets you paint with the gradient fill that you selected in the Gradient Settings dialog box. For more information, see “Using gradients” on page 117. A gradient fill is one that progresses from one color, called the foreground color, to another color, called the destination color. You can paint with Gradient ink using the Brush tool, the Bucket tool, and the shape tools.

Reveal works indirectly with the art in the previous cast position. Imagine the previous cast member’s artwork covered with a white area. Reveal ink erases the white area to show the artwork in the previous window. Reveal ink can be used to create specific shapes from shades created with the Airbrush tool. Because it is impossible to mask certain shapes for the airbrush, spray an area with the airbrush first; then, in the next cast member, paint the shapes you need with a Reveal ink. As you paint your object, you expose the airbrush pattern in the previous window.

Cycle is a color ink. As you draw with Cycle 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, select white as the foreground color and black as the destination color. This ink works only when your computer is set to 256 colors.

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

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

Darkest is a useful ink for coloring black-and-white artwork. For example, if you paint yellow over black and white, black remains black because it is darker than yellow, and white becomes yellow because yellow is darker than white.

Lightest is a useful ink for coloring black-and-white artwork. For example, if you paint 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 makes colors darker. The more times you click with the Brush tool, the darker the area becomes. The colors of the foreground, background, and 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 change the rate of this ink effect in the Paint Preferences dialog box.

Lighten makes colors lighter. The more times you click with the Brush tool, the lighter the area becomes. The color of the foreground, background, and 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 change the lightness of this ink effect in the Paint Preferences dialog box.

Smooth blurs existing artwork when it is painted with the Brush tool. It is not directional as are Smear and Smudge. The color of the foreground, background, and destination inks have no effect on Smooth. Use it to smooth out jagged edges.

Smear works with the Brush tool and functions like mixing paint. Any area you drag across with a Smear ink spreads in the direction of the brush, fading as it gets farther from the source. The color of the foreground, background, and destination inks have no effect on Smear ink.

Smudge is a color ink for the Brush tool that is similar to Smear ink. It also functions like mixing paint. The colors fade faster as they are spread. The color of the foreground, background, and destination inks have no effect on Smudge ink.

Spread works with the Brush tool in color. Whatever is under the Brush tool 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 uses the current contents of the Clipboard as a pattern with which to paint. The Clipboard contents must originate in Director.

Using bitmap filters

Bitmap filters are plug-in image editors that apply effects to bitmap images. You can install Photoshop-compatible filters to change images within Director.

Original image

Filtered image

You can apply a filter to a selected portion of a bitmap image, to an entire cast member, or to several cast members at once.

To install a filter:

- Place the filter in the Xtras folder in the Director application folder. For more information about installing Xtra extensions, see the Getting Started topics in the Director Help Panel.

To apply a filter:

1 Open the cast member in the Paint window, or select the cast member in the Cast window.

  You can apply a filter to several cast members at once by selecting them all in the Cast window. To apply a filter to a selected portion of a cast member, use the Marquee or the Lasso tool in the Paint window to select the part you want to change.

2 Select Xtras > Filter Bitmap.

3 In the Filter Bitmap dialog box, select a category on the left and a filter on the right.

  To view all the filters at once, select All from the Categories list.

4 Click Filter.

  Many filters require you to enter special settings. When you select one of these filters, a dialog box or other type of control appears after you click Filter. When you finish selecting filter settings and proceed, the filter changes the cast member.

  Some filters have no changeable settings. When you select one of these filters, the cast member changes with no further steps.

Using filters to create animated effects

You can use Auto Filter to create dramatic animated effects with bitmap filters. Auto Filter applies a filter incrementally to a series of cast members. You can use it either to change a range of selected cast members or to generate a series of new filtered cast members based on a single image. When you define a beginning and ending setting for the filter, Auto Filter applies an intermediate filter value to each cast member.

You can tween a bitmap filter with Auto Filter.

Note: Although most filters don't support auto-filtering, the Auto Filter dialog box lists only those filters that support it.

To use Auto Filter:

1 Select a bitmap cast member or a range of cast members, and select Xtras > Auto Filter.

  If you want to change only a portion of a bitmap cast member, use the Marquee or the Lasso tool in the Paint window to select the part you want to change.

2 In the Auto Filter dialog box, select a filter.

3 Click Set Starting Values, and use the filter controls to enter filter settings for the first cast member in the sequence.

  When you finish working with the filter controls, the Auto Filter dialog box reappears.

4 Click Set Ending Values, and use the filter controls to enter filter settings for the last cast member in the sequence.

5 Enter the number of new cast members you want to create. The text box is not available if you selected a range of cast members.

6 Click Filter to begin the filtering.

A message appears to show the progress. Some filters are complex and require extra time for computing.

Auto Filter generates new cast members and places them in empty cast positions following the selected cast member. If you selected a range of cast members, no new cast members appear, but the cast members in the range you selected are changed incrementally.

Using onion skinning

Onion skinning derives its name from a technique used by conventional animators, who would draw on thin "onion skin" paper so that they could see through it to one or more of the previous images in the animation.

With onion skinning in Director, you can create or edit animated sequences of cast members in the Paint window using other cast members as a reference. Reference images appear dimmed in the background. When you work in the Paint window, you can view not only the current cast member that you're painting but also one or more cast members that are blended into the image.

You can use onion skinning to do the following:

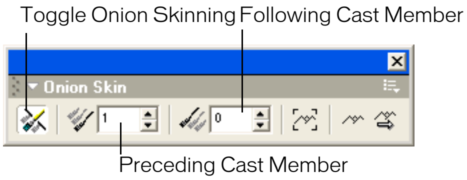
- To trace over an image or create a series of images all in register (aligned) with a particular image.
- To see previous images in the sequence, and use those images as a reference while you are drawing new ones.
- To create a series of images based on another parallel animation. The series of images serves as the background while you paint a series of foreground images.

Onion skinning uses registration points to align the current cast member with the previous ones you selected. Be careful not to move registration points for cast members after onion skinning. If you do, the cast members might not line up the way you want. For more information, see "Changing registration points" on page 112.

You must have created some cast members to use onion skinning.

To activate onion skinning:

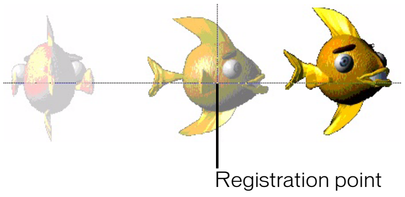
1 Open the Paint window, and select View > Onion Skin. The Onion Skin toolbar appears.



2 Click the Toggle Onion Skinning button at the far left of the toolbar to enable onion skinning.

To define the number of preceding or following cast members to display:

1 Open the Paint window, and select View > Onion Skin. The Onion Skin toolbar appears.
2 If necessary, click the Toggle Onion Skinning button on the Onion Skin toolbar to activate onion skinning.
3 Specify the number of preceding or following cast members you want to display.
   - To specify the number of preceding cast members to display, enter a number in the Preceding Cast Members text box.
   - To specify the number of following cast members to display, enter a number in the Following Cast Members text box.



Two preceding cast members shown with onion skinning and registration points

The specified number of cast members appear as dimmed images behind the current cast member. The order is determined by the position in the cast.

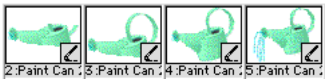
To create a new cast member by tracing over a single cast member as a background image:

1 Open the Paint window, and select View > Onion Skin. The Onion Skin toolbar appears.
2 In the Paint window, open the cast member that you want to use as the reference image or background.

3 If necessary, click the Toggle Onion Skinning button on the Onion Skin toolbar to activate onion skinning.
4 To set the background image, click the Set Background button on the Onion Skin toolbar.
5 To create a new cast member, click the New Cast Member button in the Paint window.
6 Click the Show Background button on the Onion Skin toolbar.
   The original cast member appears as a dimmed image in the Paint window. You can paint on top of the original cast member's image.
7 Paint the new cast member using the background image as a reference.

To use a series of images as a background while painting a series of foreground images:

1 In the Cast window, arrange the series of cast members you want to use as your background in consecutive order.



Cast members in the foreground and the background series must be adjacent to each other in the cast.

2 Open the Paint window, and select View > Onion Skin. The Onion Skin toolbar appears.

The Onion Skin toolbar appears.

3 If necessary, click the Toggle Onion Skinning button on the Onion Skin toolbar to activate onion skinning.

Make sure all values in the Onion Skin toolbar are set to 0.

4 Open the cast member you want to use as the first background cast member in the reference series. Click the Set Background button.

5 Select the position in the cast where you want the first cast member in the foreground series to appear. Click the New Cast Member button in the Paint window to create a new cast member.

The first cast member in the foreground series can be located anywhere in any cast.

6 Click the Show Background button to reveal a dimmed version of the background image.

7 Click the Track Background button on the Onion Skin toolbar.

8 Paint the new cast member using the background image as a reference.

9 When you finish drawing the cast member, click the New Cast Member button again to create the next cast member.

When Track Background is enabled, Director advances to the next background cast member in the series. Its image appears in the background in the Paint window.

10 Repeat step 8 until you finish drawing all the cast members in the series.

About the Paste as Pict option

You use the Paste as Pict option to paste a PICT image into the cast and have it remain in PICT format.

If you paste a PICT image into the cast using the Paste command in the Edit menu, Director converts it to a bitmap. If you want the artwork to remain in PICT format, use Paste as PICT when you paste it into the cast.

You might want to use Paste as PICT for several reasons. PICT cast members might occupy less memory than bitmap cast members. Some PICT cast members, such as compound images that consist of lines, shapes, and text, can stretch and scale more smoothly than bitmap cast members. PICT cast members also look better when printed on a laser printer.

However, PICT cast members animate more slowly than bitmap cast members, and they don't support ink effects. When you use color cycling or palette transitions, PICT cast members might have unexpected results.

Setting bitmap cast member properties

To view important information about cast members, change a cast member's name, select alpha settings, or turn on highlighting and dithering, you use bitmap cast member properties.

To view or change bitmap cast member properties:

1 Select a bitmap cast member, and click the Member tab in the Property inspector using the Graphical view.

  The Member tab displays the following:

  - A text box to view or change the cast member's name, a Comments text box to enter text that appears in the Comments column of the Cast List window, and an Unload pop-up menu that lets you determine how to remove a cast member from memory.
  - View-only fields that indicate when the cast member was created and modified as well as the name of the person who modified the cast member.

2 If the bitmap cast member is linked to an external file, the full path for the file appears in the Name text box. To choose a different file to link to the cast member, either type a new filename into the text box, or click the Browse button (...) and select the path to the new filename.

3 Click the Bitmap tab using the Graphical view.

4 To invert the current cast member when the user clicks it, select Highlight.

  Use this option to create buttons. Even if Highlight When Clicked is selected, the cast member does not do anything unless it is controlled by a behavior or a script.

5 To make Director approximate an original color in the bitmap if there is a palette problem, select Dither. When a color is not available because of a palette conflict, Dither displays a pattern of pixels of similar colors. If this option is off, Director uses the color in the current palette that is closest to the original.

6 If the imported bitmap has a white canvas that you want to remove, select Trim. If you want to retain the white canvas, deselect Trim.

7 To make Director use the alpha channel (transparency) data in the cast member, select Use Alpha.

  This option is on by default for all imported cast members with alpha channel data.

8 To determine how a transparent area receives a mouse click, use the Alpha Threshold slider to specify a value.

  Any area with a greater degree of opacity than the specified threshold can receive a mouse click.

9 To assign a different palette to an 8-bit cast member while maintaining the cast member's original palette references, select a new palette from the Palette pop-up menu.

Setting PICT cast member properties

You use PICT cast member properties to change the names of PICT cast members and set their properties.

To view or change PICT cast member properties:

1 Select a PICT cast member, and open the Property inspector in Graphical view.

2 To view or edit the cast member name, use the Name text box on the Member tab.

3 To specify how Director removes the cast member from memory if memory is low, select an option from the Unload pop-up menu. For more information, see "Controlling cast member unloading" on page 47.

Setting Paint window preferences

You can use Paint window preferences to modify the settings of several tools and drawing methods in the Paint window.

To change Paint window preferences:

1 Select Edit > Preferences > Paint.

Note: If you are using a Macintosh OS X operating system, select the Director menu, instead of the Edit menu, to access Preferences.

2 To make tools remember the last color or ink used, select the following options:

Remember Color remembers the last color used with a tool, which remains selected for the next time you use the Brush or Airbrush tools.

Remember Ink remembers the last ink used with a tool, which remains selected for the next time you use any tool.

3 To control the way colors cycle when you draw with Cycle ink, select one of the following options:

Repeat Sequence causes colors to cycle from the foreground color to the destination color and then repeat from foreground to destination.

Reverse Sequence causes colors to cycle from the foreground color to the destination color and then from destination to foreground.

4 To set a line width that is thicker than the widths available in the Paint window, use the Other Line Width slider to enter a value.

The width you set is the width that appears when you draw a line after selecting Other Line Width.

5 To set the opacity of a color when using the Blend ink effect in the Paint window, use the Blend slider to enter a value.

You can vary the blend value between 0 and 100%.

6 To set the rate at which artwork changes when you use the Darken or Lighten effects in the Paint window, use the Lighten or Darken slider to enter a value.

7 To determine how colors are used when using Smooth, Lighten, Darken, or Cycle effects, select one of the following Interpolate By options:

Color Value ignores the order of the colors in the palette and produces a continuous blend of the foreground and destination colors.

Palette Location uses all the colors in the palette between the foreground and destination colors.

Compressing bitmaps

If you plan to distribute your movie over the Internet, you can compress your bitmap images to ensure faster downloading. Director lets you compress images at the movie level and for individual cast members. Bitmap compression set at the cast member level overrides compression settings at the movie level.

In addition to Director standard compression, you can use JPEG compression and specify a range of image quality. If you have Fireworks installed, you can use the Optimize in Fireworks button to start Fireworks and then dynamically apply compression settings while viewing how your image looks at those settings. When you determine the most suitable compression level, Director remembers the settings you established in Fireworks.

To compress a bitmap at the cast member level:

1 Select bitmap cast members or sprites, and click the Bitmap tab in the Property inspector.

   If you selected multiple cast members or sprites, the Property inspector displays the compression setting if it is the same for each selected object. If the compression settings are not the same, the Compression pop-up menu is blank.

2 Click the Compression pop-up menu, and do one of the following:

   - To compress selected bitmaps using the same settings as those established for movie-level compression, select Movie Setting. For more information about setting bitmaps at the movie level, see the information about the Compression tab under "Changing Publish Settings" on page 453.
   - To use the standard Director compression, select Standard.
   - To use JPEG compression, select JPEG, and move the slider bar to the desired level of compression. The higher the number you specify, the less your bitmap is compressed (that is, 100 indicates no compression).

Movie Setting is usually the default compression setting, except under certain conditions when the compression feature is disabled or when Director controls image-compression choices.

For example, when the image is a JPEG, the compression setting defaults to JPEG compression. You can't select another compression option.

Similarly, the Compression setting defaults to Standard compression, and you can't change this setting when the cast member is any of the following:

- An 8-bit cast member created in the Paint window
- A GIF imported as a bitmap with no alpha channel information
- An 8-bit PNG
- A linked cast member or a cast member created with script

Note: If you open a Director 7 movie in Director MX, bitmap cast members are assigned Movie Setting as the default, and compression settings at the movie level, set in the Publish Settings dialog box, default to the Standard compression setting. This ensures the movie plays as it did in Director 7.

To compress bitmaps at the movie level:

1 Select File > Publish Settings.

   The Publish Settings dialog box appears.

2 On the Compression tab, make a selection from the Image Compression pop-up menu, and click OK.

   - To use the standard Director compression, select Standard.
   - To use JPEG compression, select JPEG, and move the slider bar to the desired level of compression. The higher the number you specify, the less your bitmap is compressed (that is, 100 indicates no compression).

   Note: Director saves your publish settings when you save your movie.

Working with Macromedia Fireworks

You can combine the power of Macromedia Fireworks and Director. Fireworks lets you export graphics and interactive content into Director. The export process preserves the behaviors and slices of the graphic. You can safely export sliced images with rollovers and even layered images. Director users can take advantage of the optimization and graphic design tools of Fireworks without compromising quality.

Placing Fireworks files into Director

Director can import flattened images from Fireworks, such as JPEGs and GIFs. It can also import 32-bit PNG images with transparency. For sliced, interactive, and animated content, Director can import Fireworks HTML.

Exporting graphics with transparency

In Director, transparency can be achieved by importing 32-bit PNG images. You can export 32-bit PNG graphics with transparency from Fireworks.

To export a 32-bit PNG with transparency:

1 In Fireworks, select Window > Optimize, change the export file format to PNG 32, and set Matte to transparent.

2 Select File > Export.

3 Select Images Only from the Save as Type pop-up menu. Name the file, and click Save.

Exporting layered and sliced content to Director

By exporting Fireworks slices to Director, you can export sliced and interactive content such as buttons and rollover images. By exporting layers to Director, you can export layered Fireworks content such as animations.

To export Fireworks files to Director:

1 In Fireworks, select File > Export.

Note: You can also click the Quick Export button and select Source as Layers or Source as Slices from the Director pop-up menu. Select Source as Layers if you are exporting an animation, and select Source as Slices if you are exporting interactive content such as buttons.

2 In the Export dialog box, type a filename, and select a destination folder.

3 Select Director from the Save As pop-up menu.

4 Select one of the following options from the Source pop-up menu:

Fireworks Layers exports each layer in the document. Select this option if you are exporting layered content or an animation.

Fireworks Slices exports the slices in the document. Select this option if you are exporting sliced or interactive content such as rollover images and buttons.

5 Select Trim Images to automatically crop the exported images to fit the objects on each frame.

6 Select Put Images in Subfolder to select a folder for images.

7 Click Save.

Importing Fireworks files into Director

In Director, you can import flattened images that you have exported from Fireworks, such as JPEGs, GIFs, and 32-bit PNGs. You can also import Fireworks layers, slices, and interactive elements by inserting Fireworks HTML.

To import a flattened Fireworks image:

1 In Director, select File > Import.

2 Navigate to the desired file, and click Import.

3 Change options, if desired, in the Image Options dialog box. For information about each option, see "Choosing import image options" on page 45.

4 Click OK.

The imported graphic appears in the cast as a bitmap.

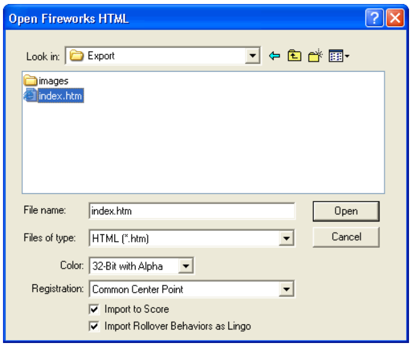
To import layered, sliced, or interactive Fireworks content:

1 In Director, select Insert > Fireworks > Images from Fireworks HTML.

Note: The location and name of this menu command might be different depending on your version of Director.

The Open Fireworks HTML dialog box appears.

2 Locate the Fireworks HTML file you exported for use in Director.



File name: index.htm
Files of type: HTML (*.htm)
Color: 32-Bit with Alpha
Registration: Common Center Point
☑ Import to Score
☑ Import Rollover Behaviors as Lingo

3 Change the following options if desired:

Color lets you specify a color depth for the imported graphics. If the graphics contain transparency, select 32-bit color.

Registration lets you set the registration point for the imported graphics.

Import Rollover Behaviors as Lingo converts Fireworks behaviors to Lingo code.

Import to Score places cast members into the Score when they're imported.

4 Click Open.

The graphics and code from the Fireworks HTML file are imported.

Note: If you are importing a Fireworks animation, drag keyframes in Director to offset the timing of each imported layer as necessary.

Editing Director cast members in Fireworks

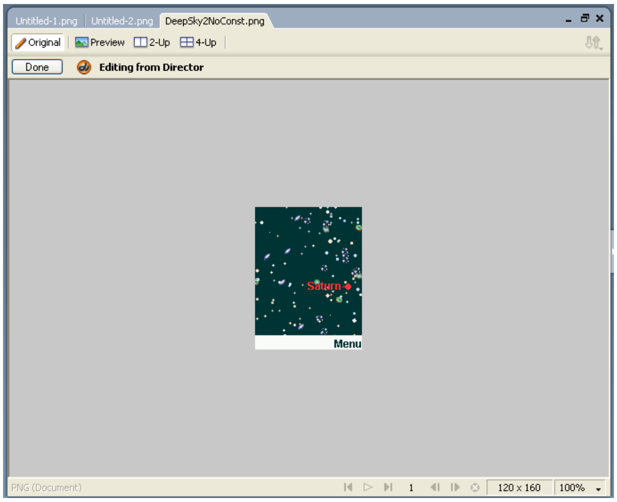
Using launch-and-edit integration, you can make changes to Director cast members by starting Fireworks and editing them from inside Director. You can also start Fireworks from inside Director to optimize cast members.

To start Fireworks and edit a Director cast member:

1. In Director, right-click (Windows) or Control-click (Macintosh) the graphic in the Cast window.
2. Select Launch External Editor from the context menu.

   Note: If Fireworks does not start as your external image editor, select Edit › Preferences › Editors in Director, and set Fireworks as the external editor for bitmap graphic file types. (If you are using a Macintosh OS X operating system, select the Director menu, instead of the Edit menu, to access Preferences.)

   The file opens in Fireworks, and the document window indicates that you are editing a file from Director.



3. Make changes to the image, and click Done when you finish.

   Fireworks exports the new graphic to Director.

Optimizing cast members in Director

You can start Fireworks from Director to make quick optimization changes to selected cast members.

To start Fireworks and change optimization settings for a Director cast member:

1. In Director, select the cast member in the Cast window, and click Optimize in Fireworks on the Bitmap tab in the Property inspector.
2. In Fireworks, change the optimization settings as desired.
3. Click Update when you finish. Click Done if the MIX Editing dialog box appears.

The image is exported back to Director using the new settings.

CHAPTER 6
Vector Shapes

Vector shapes and bitmaps are the two main types of graphics used with Macromedia Director MX 2004. A vector shape is a mathematical description of a geometric form that includes the thickness of the line, the fill color, and additional features of the line that can be expressed mathematically. A bitmap defines an image as a grid of colored pixels, and it stores the color for each pixel in the image. For more information about using bitmaps in Director, and how they compare to vector shapes, see Chapter 5, “Bitmaps,” on page 99.

You can create vector shapes in the Director Vector Shape window by defining points through which a line passes. The shape can be a line, a curve, or an open or closed irregular shape that can be filled with a color or gradient. You can also use Lingo or JavaScript syntax to dynamically create and control vector shapes. You can create a vector shape entirely with script or modify an existing one as the movie plays.

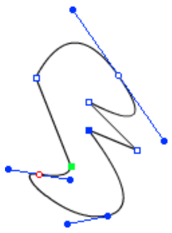
Because vector shapes are stored as mathematical descriptions, they require less RAM and disk space than an equivalent bitmap image and they download faster from the Internet.

Drawing vector shapes

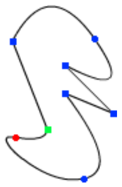
You create vector shapes with drawing tools in the Vector Shape window. You can use the Pen tool to create irregular shapes or use shape tools to create rectangles and ellipses. A vector shape can include multiple curves, and you can split and join the curves. Shape properties such as fill color, stroke color, and stroke width are set at the cast-member level and not for individual curves.

When you create vector shapes, you create vertices, which are fixed points. You can also create handles, which are points that determine the degree of curvature between vertices. These curves are known as Bézier curves. A vertex without a handle creates a corner.

As you draw vector shapes, control handles appear on the vertices: round curve points for vertices with handles and square corner points for vertices without handles.



- The first vertex in a curve is green.
- The last vertex in a curve is red.
- All other vertices are blue.
- Unselected vertices are solid.
- Selected vertices are unfilled.



To open the Vector Shape window:

- Select Window > Vector Shape.

Zooming in and out in the Vector Shape window

You can use the Magnify tool or the Zoom commands on the View menu to zoom in or out at four levels of magnification.

To zoom in or out, do one of the following:

- Select View > Zoom and then select the level of magnification.
- Right-click (Windows) or Control-click (Macintosh) and select Zoom In or Zoom Out from the context menu.
- Press Control + the Plus (+) key (Windows) or Command + the Plus (+) key (Macintosh) to zoom in, or Control + the Minus (-) key (Windows) or Command + the Minus (-) key (Macintosh) to zoom out.

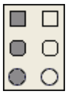
To return to normal view:

- Select View > Zoom > 100%.

Using vector shape drawing tools

You use the tools in the Vector Shape window to draw free-form shapes or geometric figures. You can define a shape with the Pen tool by creating curve or corner points through which a line passes.

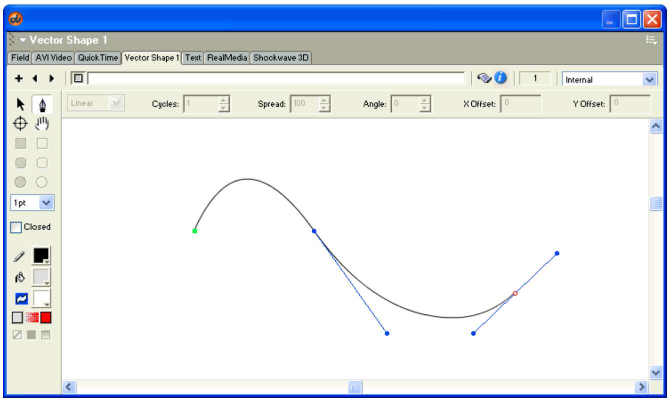
To draw regular shapes, you use the Rectangle, Rounded Rectangle, and Ellipse tools.



To create a vector shape using the Pen tool:

1 In the Vector Shape window, click the New Cast Member button.

2 Click the Pen tool and begin to draw:



- To create a corner point, click once.
- To create a curve point, click and drag. Dragging creates control handles that define how the line curves through the point that you define.
- To constrain a new point to vertical, horizontal, or a 45° angle, hold down Shift while clicking.

To draw using a basic shape tool:

1 In the Vector Shape window, click the New Cast Member button.

2 Select the Filled or Unfilled Rectangle, Rounded Rectangle, or Ellipse tool.

3 Hold down the mouse button to start a shape, drag to draw, and release the mouse button to end the shape.

To constrain a rectangle to a square, or to constrain an ellipse to a circle, hold down Shift while dragging.

To select a vertex or vertices, do one of the following:

- To select one vertex, select the Arrow tool and click the vertex.
- To select multiple vertices, either select the Arrow tool and hold Shift while clicking the vertices, or click and drag a selection rectangle over the vertices (marquee-select).
- To select all the vertices in a curve, select the Arrow tool and double-click one of the vertices in the curve.

To create multiple curves, do one of the following:

- If you use the Pen tool, double-click the last vertex drawn. The next vertex starts a new curve.
- With no vertices selected, use the Pen tool to start a new curve.
- To create two separate curves from one, select two adjacent vertices in a curve and select Modify > Split Curve.
- If the current shape is empty or closed, select one of the shape tools and draw a new shape.

Note: If you create multiple shapes in the Vector Shape window, Director treats all the shapes as one if you change shape attributes. If, for example, you create ten open shapes in one Vector Shape window and select Close, Director closes all ten shapes.

Choosing fill and line settings for vector shapes

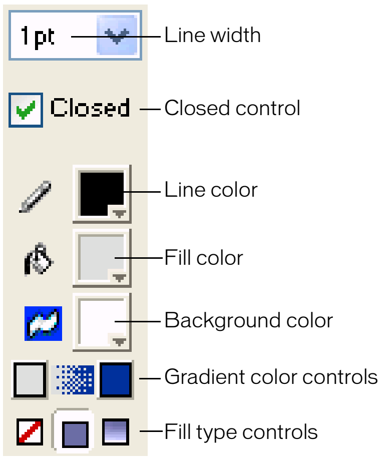
You can use either controls in the Vector Shape window or Lingo or JavaScript syntax to choose a vector shape's fill color, line width and color, and background color. The background is the area outside of a vector shape but within the cast member's bounding rectangle.

Because a vector shape is a single object, you don't need to select any part of the vector shape to make the following changes.

To select the fill and line settings:

1 Open a vector shape in the Vector Shape window.

2 Select fill and line settings using the appropriate controls at the left of the window.



- To set the line width, select a point size option from the Line Width menu.
- To close or open vector shapes, select or deselect the Closed option (for more information, see "Editing vector shapes" on page 139).
- To choose the line color, select a color from the Line Color menu.
- To choose the fill color, select a color from the Fill Color menu.
- To set the background color, select a color from the Background Color menu. Choosing a background color that matches the color of the background results in better performance than using Background Transparent ink.
- To set gradient fill colors, select colors from the Gradient Colors control. For more information about creating gradient fill, see "Editing vector shapes" on page 139.
- To set the fill type, select from the following Fill type control options: No Fill, Solid, or Gradient.

Specifying vector shape fills and strokes with Lingo or JavaScript syntax

You can script in Lingo or JavaScript syntax to specify a vector shape's fills and strokes.

To specify the strokes that form a vector shape in script:

- Set the `strokeColor` and `strokeWidth` cast member properties. For more information about these properties, see the Scripting Reference topics in the Director Help Panel.

To specify a vector shape's fill in script:

- Set the `fillColor`, `fillMode`, `fillOffset`, and `fillScale` cast member properties. For more information about these properties, see the Scripting Reference topics in the Director Help Panel.

Editing vector shapes

To edit vector shapes, you use the Vector Shape window. You change vector shapes by moving, adding, or deleting control points and changing the way they control curves. You can also change the way a vector shape is placed on the Stage by moving its registration point, using either the Vector Shape window or Lingo or JavaScript syntax.

To adjust the outline of a vector shape:

1 Open a vector shape in the Vector Shape window.

2 Click the Arrow tool and make any of the following changes:

- To move a curve or corner point, drag it to any location.
- To move multiple points, Shift-click or marquee select all the points you want to move, then drag any one of the selected points.
- To drag a single curve within a shape, select the Arrow tool and drag the curve. If the curve is filled, you can click anywhere within the filled area and drag the curve.
- To adjust a curve, select a curve point and drag a control handle.

By default, the two control handles remain at a 180° angle from each other. If you want to drag one control handle independently from the other one, hold down Control (Windows) or Command (Macintosh) when you drag it. To constrain the control handles to vertical, horizontal, or a 45° angle, hold down Shift as you move them.

- To change a corner point to a curve point, Alt-click (Windows) or Option-click (Macintosh) and drag away from the handle to extend a control handle.
- To change a curve point to a corner point, drag the control handles directly over the curve point.
- To delete a point, select the point and press Backspace (Windows) or Delete (Macintosh).
- To move the window view without using the scroll bars, click the Hand tool and drag anywhere inside the shape.

To add a point in the middle of a shape:

1 Open a vector shape in the Vector Shape window.

2 Click the Pen tool.

3 If the shape is closed, move the pointer over a line until it changes and then click the mouse button. If the shape is open, hold down Alt (Windows) or Option (Macintosh) and move the pointer over a line until it changes; then click the mouse button.

To add a new point that is connected to a certain end point:

1 Click the Arrow tool and select an end point.

2 Click the Pen tool, and click the location where you want to add the next point.

To join two curves:

1 Select a vertex in each curve.

  If you select two endpoint vertices, you join them. If you select points in the middle of the curve, you join the start of the second curve to the end of the first curve.

2 Select Modify > Join Curves.

To split two curves:

- Select two adjacent vertices, and select Modify > Split Curves.

To change the registration point:

1 Click the Registration Point tool.

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

  The pointer changes to a cross hair when you move it to the window.

2 Click to set the new registration point.

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

3 To reset the default registration point at the center of the cast member, double-click the Registration Point tool.

To change a vector shape cast member's registration point in Lingo or JavaScript syntax:

- Set the `regPoint` or `regPointVertex` cast member property. You can test the `centerRegPoint` property to determine whether Director automatically recenters the registration point when the cast member is edited. (If you specify a value for `regPointVertex`, any values in the `regPoint` and `centerRegPoint` properties are ignored.) For more information about these properties, see the Scripting Reference topics in the Director Help Panel.

To close or open vector shapes:

- Select or deselect the Closed check box at the left side of the window.

  If the shape is closed, Director draws a line between the last and first points defined; if it is open, Director removes the line between the last and first points.

To close a shape with Lingo or JavaScript syntax:

- Set the `closed` cast member property to `true`. For more information about this property, see the Scripting Reference topics in the Director Help Panel.

To scale a vector shape:

- Control-Alt-drag (Windows) or Command-Option-drag (Macintosh) to proportionally resize a vector shape.

  You can also enter a scaling percentage for a vector shape using the Cast Member Properties dialog box. For more information, see "Setting vector shape properties" on page 142.

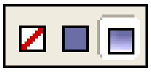
Defining gradients for vector shapes

You can use controls in the Vector Shape window or Lingo or JavaScript syntax to specify the type of gradient, how it is placed within a shape, and how many times it cycles within the shape. A gradient for a vector shape shifts between the fill color and the end color you define. You can create linear or radial gradients. Changes you make to vector shape gradients have no effect on gradients for bitmaps in the Paint window. You can fill only closed vector shapes with gradients.

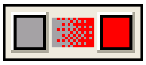
To define a gradient for a vector shape:

1 Create a closed vector shape in the Vector Shape window.

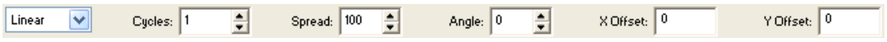
2 Click the Gradient button in the Fill type controls.



3 To select colors for the gradient, click the color box on the left side of the Gradient Colors control and select a starting color from the Color menu. To select the ending color, repeat this step using the color box on the right side of the Gradient Colors control.



4 Select Linear or Radial from the Gradient Type pop-up menu at the top of the window.


Linear Cycles: 1 Spread: 100 Angle: 0 X Offset: 0 Y Offset: 0

5 To define the number of times the gradient should change colors within the shape, use the Cycles control.

6 To specify the rate at which the gradient shifts between colors, use the Spread control to enter a percentage.

A setting of 100% uses the entire width or height of the shape to gradually shift colors. Lower settings make the shift more abrupt. For settings over 100%, the end color is reached at a theoretical location beyond the edges of the shape.

7 To rotate the gradient within the shape, use the Angle control to enter the number of degrees.

This setting affects only linear gradients.

8 To offset the gradient within the shape, enter X Offset (horizontal) and Y Offset (vertical) values.

To specify a gradient in Lingo or JavaScript syntax:

- Set the `fillColor`, `fillDirection`, `fillMode`, `fillOffset`, `fillScale`, `gradientType`, and `endColor` cast member properties. For more information about these properties, see the Scripting Reference topics in the Director Help Panel.

Controlling vector shapes with Lingo or JavaScript syntax

You can use script to modify a vector shape by setting properties and using methods related to the shape's vertices. For more information about the following properties, expressions, and methods, see the Scripting Reference topics in the Director Help Panel.

- To display a list that contains the location of each vertex and control handle in a vector shape, test the `vertexList` property.
- To access a vertex directly, use the `vertex` chunk expression.
- To add or delete a vertex, use the `addVertex()` or `deleteVertex()` method.
- To move a vertex or a vertex handle, use the `moveVertex()` or `moveVertexHandle()` method.
- To display the vertex list for a vector shape, test the `curve` property.
- To add a new shape to the vector shape, use the `newCurve()` method.
- To display or specify the registration point for the vector shape's cast member, test or set the `regPointVertex` property.
- To display or specify the point around which a vector shape scales and rotates, test or set the `originMode` property.

Setting vector shape properties

You can use the Property inspector to view and change settings for selected vector shape cast members. In addition to setting standard name and unload properties, you can specify anti-aliasing based on system performance and how the shape fits within the bounding rectangle.

To view or change vector shape cast member properties:

1. Select a vector shape cast member and click the Member tab on the Property inspector.
2. To specify how Director removes the cast member from memory if memory is low, be sure you're in Graphical view and select an option from the Unload pop-up menu. For more information, see "Controlling cast member unloading" on page 47.
3. To set specific vector shape settings, click the Vector tab.
4. To set the stroke color, choose a color from the Color menu, or enter a color value in the Stroke Color text box.
5. To set the width of the stroke, use the Width slider.
6. To set the fill color, select a color from the Color menu, or enter a color value in the Fill Color text box.
7. To set the type of fill, select one of these three options: No Fill, Solid, or Gradient.
8. To change the setting for anti-aliasing, click Anti-alias.

   A check mark indicates that Anti-alias is on.

9 To specify how vector shapes are scaled on the Stage, select an option from the Scale Mode pop-up menu.

Show All maintains the vector shape's aspect ratio and, if necessary, fills in any gap along the horizontal or vertical dimension using the vector shape's background color.

No Border maintains the vector shape's aspect ratio by cropping the horizontal or vertical dimension as necessary without leaving a border.

Exact Fit stretches the vector shape to fit the sprite exactly, disregarding the aspect ratio.

Auto Size adjusts the vector shape's bounding rectangle to fit the movie when it is rotated, skewed, or flipped.

No Scale places the vector shape on the Stage with no scaling. The movie stays the same size no matter how you resize the sprite, even if it means cropping the vector shape.

10 To change the size of the cast member, either enter a percentage in the Percentage text box (Graphical view) or use the Scale slider (List view) to determine a percentage.

Using shapes

Shape cast members are the same non–anti-aliased shapes that were available in older versions of Director. Shapes are different cast member types than vector shapes. Similar to vector shapes, they are very memory-efficient.

Shapes are images you can create directly on the Stage with the Line, Rectangle, Rounded Rectangle, and Ellipse tools on the Tool palette. You can fill shapes with a color, pattern, or custom tile. Shapes require even less memory than vector shapes, but Director does not anti-alias shapes, so they don't appear as smooth on the Stage as vector shapes. You can use shapes for creating simple graphics and backgrounds when you want to keep your movie as small as possible. Shapes are especially useful for filling an area with a custom tile to create an interesting background that downloads quickly from the Internet. For more information, see "Creating a custom tile" on page 120.

To create a shape:

1 Select a frame in the Score where you want to draw a shape.

2 Select a shape, color, line thickness, and pattern setting with the controls in the Tool palette. (To open the Tool palette, select Window > Tool Palette, and select either Classic or Default view; shapes are not available in the Flashcomponent view.)

3 Click a tool and then drag on the Stage to draw the shape.

The new shape appears on the Stage and in the Cast window.

Setting shape cast member properties

You can use cast member properties to view and change settings for selected shape cast members. You can change the type of shape and choose a new fill color or pattern. You can also use Lingo or JavaScript syntax to control shape cast member properties.

To view or change shape cast member properties:

1. Select a shape cast member and open the Property inspector in Graphical view.
2. Use the Name text box on the Member tab to view or edit the cast member name.
3. To specify how Director removes the cast member from memory if memory is low, select an option from the Unload pop-up menu. For more information, see "Controlling cast member unloading" on page 47.
4. To change the type of shape, click the Shape tab and select an option from the Shape pop-up menu.
5. To fill the shape with the current color and pattern, select Filled.

To specify a shape's type in Lingo or JavaScript syntax:

- Set the `shapeType` cast member property. For more information about this property, see the Scripting Reference topics in the Director Help Panel.

To specify a shape's fill in Lingo or JavaScript syntax:

- Set the `filled` and `pattern` shape cast member properties. For more information about these properties, see the Scripting Reference topics in the Director Help Panel.

To specify the line size for a shape in Lingo or JavaScript syntax:

- Set the `lineSize` cast member or sprite property. For more information, see the Scripting Reference topics in the Director Help Panel.

CHAPTER 7
Color, Tempo, and Transitions

Several behind-the-scenes functions in Macromedia Director MX 2004 are important to the appearance and performance of a movie.

To control the way Director manages colors, it's important to understand the difference between RGB and index color, and how to assign colors to various elements in your movie. See the next section.

To control the speed at which your movie plays, you use settings in the tempo channel. For more information, see "About tempo" on page 157.

To make scenes in your movie flow together without creating the animation yourself, you can use predefined transitions. For more information, see "Using transitions" on page 160.

All these features involve using the channels at the top of the Score.

Controlling color

Selecting colors for movie elements is as simple as making a selection from a menu. To make sure that the colors you select are displayed correctly on as many systems as possible, it helps to understand how Director controls color.

Director provides a variety of color controls. The following list describes the most important:

- Use the Movie tab in the Property inspector to change modes for selecting colors. Click either the RGB or Index radio button. (RGB assigns to the movie all color values as absolute RGB values. Index assigns color to the movie according to its position in the current palette.)
- Use the pop-up Color menu to select colors for movie elements. The Color menu is available throughout the Director application—for example, in the Tool palette.
- Use Transform Bitmap to remap bitmap images to new palettes and change their color depth. You can also make the same changes when you import a bitmap. For more information, see "Changing size, color depth, and color palette for bitmaps" on page 113, and "About importing bitmaps" on page 100.
- Use the Score's palette channel to change the movie's color palette as a movie plays.
- Use the Color Palettes window to change the colors in a color palette or to create a custom color palette cast member.

Specifying palette index and RGB color

Director can use either palette index values or RGB values to specify colors. RGB values are much more reliable and accurate for specifying colors than palette index values. RGB is the system that most web pages use.

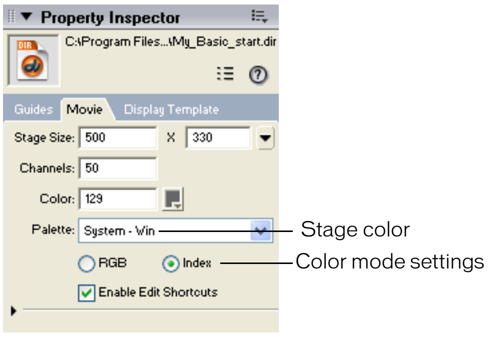
Director identifies a palette index color by the number of its position in a set of colors called a color palette. Color number 12, for example, might be blue. If a different palette is active, color number 12 might be red. When a computer is set to display 256 colors or fewer, it can display only the colors in the palette currently active in the system. This means that images created to display with the colors of one palette do not appear correctly when a different palette is active. If you use palette index color in a movie and then switch palettes during the movie, or never make sure that the correct palette is active, the images in your movie might appear with the wrong colors.

Director identifies an RGB color as a set of hexadecimal numbers that specify the amounts of red, green, and blue required to create the color. When a computer is set to display thousands or millions of colors, Director always displays RGB colors accurately. When a computer is set to 256 colors, Director finds the closest color in the current color palette to approximate the RGB color.

To choose the color mode for the current movie, you use the color mode settings on the Movie tab of the Property inspector. When you select RGB, all the colors you select from the Color menu in Director are specified in RGB values. When you select Index, the colors you choose are specified according to their position in the current palette. The Color menu indicates which method is being used.

To change the color mode of a movie:

1 Display the Movie tab of the Property inspector.



Stage Size: 500 X 330
Channels: 50
Color: 129
Palette: System - Win
RGB [✓ Index]
[✓] Enable Edit Shortcuts

2 Select either RGB or Index.

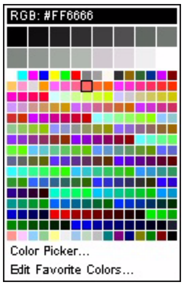
Changing the color depth of a movie

When you save a Director movie, it is set to the same color depth as the system on which you are authoring. You can use Lingo or JavaScript syntax to reset the system color depth to match the color depth of a movie. For more information, see the Scripting Reference topics in the Director Help Panel.

If you want to set the color depth of a movie without using script, you can use system utilities to change the color depth of your system before you save the movie file. On the Macintosh, you can also make the movie reset the system color depth by selecting Edit > Preferences > General and selecting Reset Monitor to Movie's Color Depth. (If you are using a Macintosh OS X operating system, select the Director menu, instead of the Edit menu, to access Preferences.)

Choosing colors for movie elements

Use the Color menu to select colors for movie elements such as the Stage, vector shapes, and the foreground and background of sprites. For some elements, such as Stage and sprite colors, you can also enter hexadecimal values for any RGB color. The Color menu displays the colors in the current palette; the 16 larger color boxes at the top of the menu identify your favorite colors.



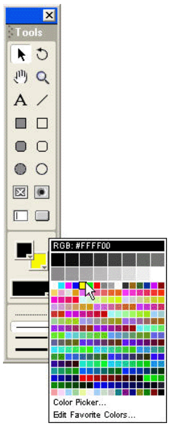
If the movie is set to specify colors as RGB values, selecting a color from the Color menu specifies the RGB value of the color, not its index value. (For an explanation of the difference between index and RGB color, see "Specifying palette index and RGB color" on page 146.) The bar at the top of the Color menu indicates whether the movie is set to RGB or index color.

If you want to select a color that is not in the current palette (and therefore not available on the Color menu), you can use the system color picker to specify any color. You can also change the set of colors available on the Color menu by displaying a different color palette.

To open the Color menu:

1 Do one of the following:
  - Select a sprite and display the Sprite tab of the Property inspector.
  - Select Window > Tool Palette.

2 Click and hold the mouse button while pointing at the Foreground Color and Background Color buttons.



Note: To open the Color menu in the opposite mode (RGB or index), hold down the Alt key (Windows) or Option key (Macintosh) while clicking the color box.

To select colors not on the Color menu:

1 Open the Color menu.

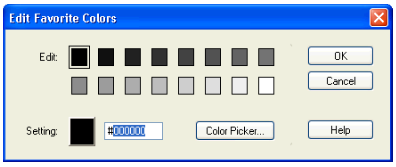
2 Click Color Picker.

3 Use the color picker that appears to select colors.

To edit the favorite colors on the Color menu:

1 Open the Color menu.

2 Select Edit Favorite Colors.



3 Select the color box you want to change.

4 Select a new color for the box using one of the following options:
  - Click the color box to open the Color menu and select a color from the current palette.
  - Enter an RGB value for a color in the box to the right of the color box.
  - Click Color Picker and then use the system color picker to specify a new color.

5 Click OK.

To change the color palette displayed on the Color menu:

1 Select Window > Color Palettes or double-click the mouse button on the Foreground Color and Background Color buttons in the Tool palette.

2 Select a color palette from the Palette pop-up menu.

Changing color palettes during a movie

The palette channel in the Score determines which palette is active for a particular frame in a movie. To define the palette that is active in a particular frame of a movie, use Modify > Frame > Palette. When the playhead reaches the frame with the palette change, Director switches to the new palette.

The settings in the palette channel have no effect on a movie playing in a web browser. Do not use any of these settings for movies on the web.

For a stand-alone disk-based movie that takes over the entire screen, changing palettes during a movie is a viable option for displaying 8-bit graphics with the best possible colors.

If you place a cast member that has its own custom palette on the Stage—and if it's the first cast member that has a different palette in the frame—Director automatically assigns the new palette to the palette channel. The new palette becomes the active palette unless you clear it from the palette channel or replace it with a different palette, and it remains in effect until you set a different palette in the palette channel.

Only one palette can be active at any time. If an 8-bit image appears with the wrong colors, it requires a different palette.For more information, see "Solving color palette problems" on page 154.

Director contains several color palettes. The Windows and Macintosh system palettes are the default selections. Web216 is nearly identical to the palettes used by Netscape Navigator and Microsoft Internet Explorer. Use it for any movie you plan to play in a browser. Any additional palettes you create or import appear as cast members.

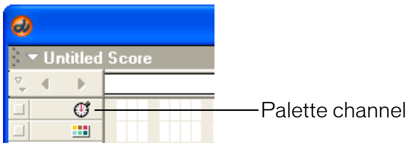
While working on a movie, you can change the active palette in the authoring environment by selecting a new palette in the Color Palettes window. The palette that is active in the authoring environment while you work does not change the palette in the movie on which you're working. Any settings in the palette channel reset the active palette as soon as the movie plays.

To specify a palette:

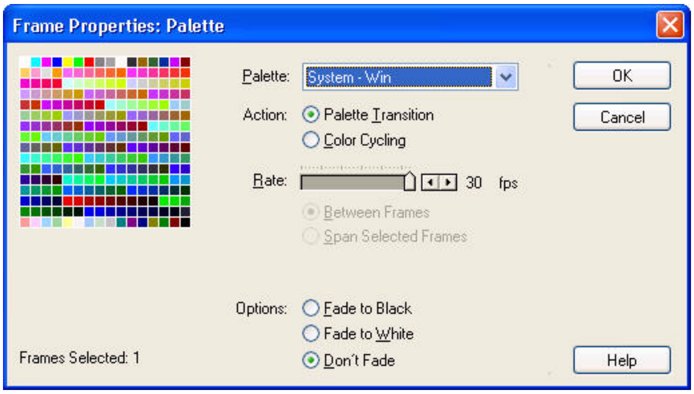
1 In the Score, do one of the following:

- Double-click the cell in the palette channel where you want the new palette setting to appear.
- Right-click (Windows) or Control-click (Macintosh) the cell in the effects channel where you want the new palette setting to appear, and then select Palette from the Context menu.
- Select the cell in the effects channel where you want the new palette setting to appear, and then select Palette from the Score window's Options menu.
- Select a frame in the palette channel, and select Modify > Frame > Palette.

(If you don't see the palette channel, the effects channel is hidden. To display it, click the Hide/ Show Effects Channel tool in the upper right of the score window.)



2 Select the options you want to use in the Frame Properties: Palette dialog box.


Frame Properties: Palette

Palette: System - Win

Action: (●) Palette Transition ( ) Color Cycling

Rate: 30 fps

(●) Between Frames ( ) Span Selected Frames

Options: ( ) Fade to Black ( ) Fade to White (●) Don't Fade

Frames Selected: 1

- Select a new palette.
- Specify how you want Director to manage the palette change. For example, to hide a palette change within a fade, first select a new palette from the pop-up menu. Select the Palette Transition option, and then select Fade to Black or Fade to White. Use the Rate slider to set the speed of the fade.

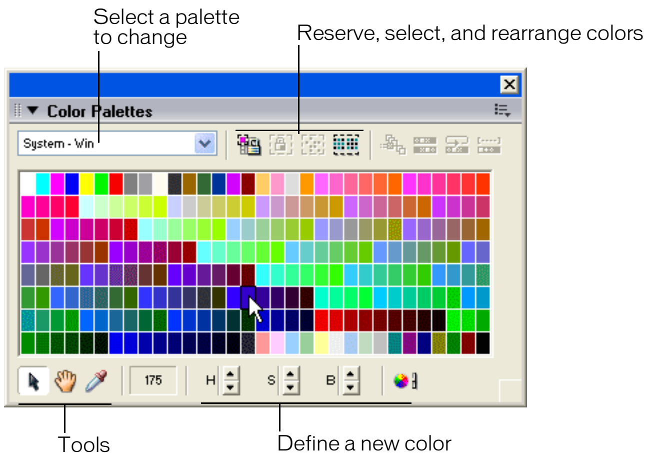
To stop the movie while the palette changes, first select a new palette from the Palettes pop-up menu. Select the Palette Transition option, and then select Between Frames. Use the Rate slider to set the speed of the transition.

3 Click Set.

The palette you select now appears in the cell that you selected in the Score's palette channel. The setting remains in effect in the movie until you set a different palette in the palette channel.

Using the Color Palettes window

Use the Color Palettes window to change and rearrange color palettes and to determine which colors in a palette are used in an image. This section explains basic features of the Color Palettes window.


Select a palette to change
Reserve, select, and rearrange colors
Color Palettes
System - Win
175
H
S
B
Tools
Define a new color

If you add new palettes to your movie from other graphics applications, those palettes appear in the palette list and in the Cast window.

The row of buttons on the right side of the Color Palettes window are for reserving, selecting, and rearranging colors in the current palette. If you attempt to change one of the nine built-in palettes, Director creates a copy of the palette for you to modify.

Note: Selecting a new palette in the Color Palettes window does not change the palette for the movie or any frame in the movie. Use the Movie tab in the Property inspector to select the movie color palette, or select Modify > Frame Palette to change the color palette at a particular frame.

When you modify a palette, all the cast members using the palette also change, so make sure you always keep a copy of the original palette.

To open the Color Palettes window:

- Select Window > Color Palettes.

To edit a palette already used in a movie:

1 Select Window > Color Palettes.

2 Select the palette you want to edit from the Palettes pop-up menu.

3 Double-click any color within the palette.

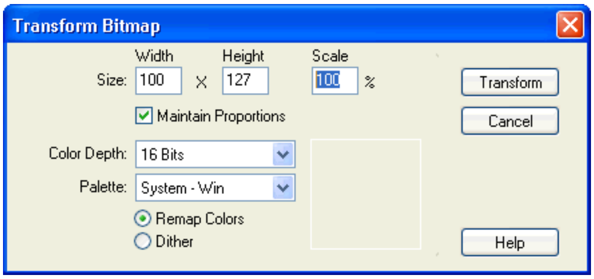
Director makes a copy of the palette and prompts you to enter a name.

4 Enter a name, and press OK.

5 Edit the palette using any of the methods discussed later in this section.

6 Select all the cast members that use the old version of the palette, or use Find to locate all the cast members using a particular palette.

7 Select Modify > Transform Bitmap and select the desired options.


Transform Bitmap

Size: Width 100 × Height 127 Scale 100 %

[✓] Maintain Proportions

Color Depth: 16 Bits

Palette: System - Win

[✓ Remap Colors] Dither

Note: Be sure to select Remap Colors, not Dither.

8 Click Transform to remap all the cast members to the new palette.

To select one or more colors:

1 Click a color in the Color Palettes window. If the selection arrow is not active, click the Arrow tool at the bottom of the window.

2 To select a range, drag across colors or click the first color in the range, and then Shift-click the last.

3 Control-click (Windows) or Command-click (Macintosh) to select multiple discontiguous colors.

To match the color of any pixel on the Stage with the same color in the palette:

1 Click the Eyedropper tool.

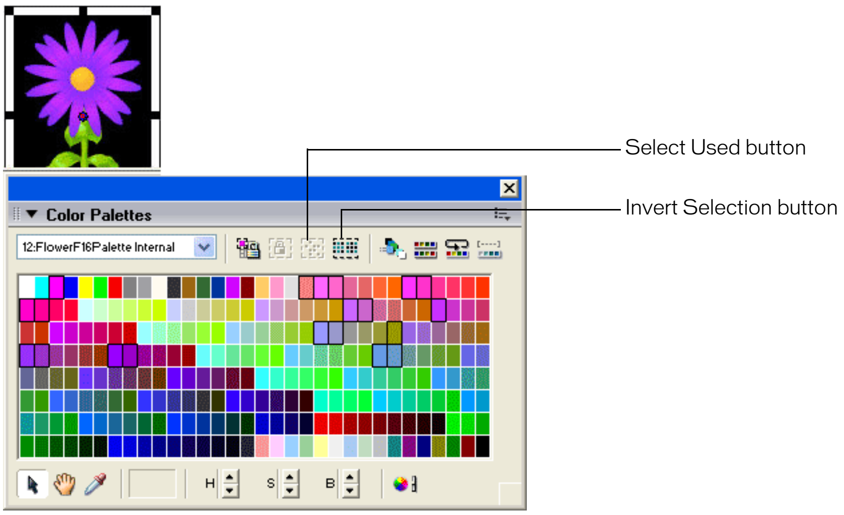
2 Drag any color in the Color Palettes window to any point on the Stage.

The selection in the Color Palettes window and the foreground color in the Tool palette changes to the color at the pointer location.

To select colors in the palette used by the current cast member:

1 In the Cast window, select the cast member.

2 Select Window > Color Palettes.



3 Click the Select Used Colors button in the Color Palettes window.

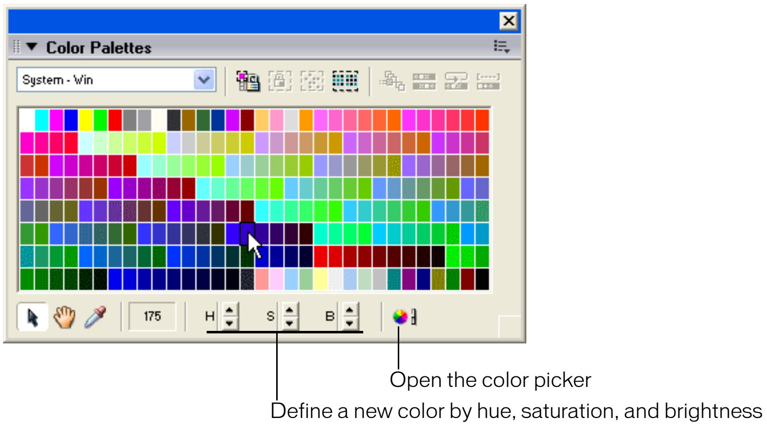
4 In the Select Colors Used In Bitmap dialog box, click Select.

To select all colors not currently selected:

- Click the Invert Selection button in the Color Palettes window.

Changing colors in a color palette

You can define a new color for a color palette by selecting a color you want to change and then using either the controls at the bottom of the Color Palettes window or the system color.



To edit selected colors in the Color Palettes window:

1 Select Window > Color Palettes.
2 Select the palette you want to change from the Palette pop-up menu.
3 Select a color within the palette to change.

   If you attempt to change one of the default palettes, Director makes a copy of the palette and prompts you to enter a name.
4 To change the color using the H, S, and B (hue, saturation, and brightness) controls, click the arrows next to the controls.

   Hue is the color that is created by mixing primary colors.

   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, in the case of black, a shade of gray.

   Brightness controls how much black is mixed in with a color. Colors that are very bright have little or no black. As more black is added, the brightness is reduced, and the color gets darker. If brightness is reduced to 0, then no matter what the values are for hue or saturation, the color is black.
5 To change the color using the system color picker, click the Color Picker button.

   For instruction on using the Windows or Macintosh color picker, see your system documentation.

Controlling color palettes with Lingo or JavaScript syntax

By using the `puppetPalette` method, you can change the current palette and specify how quickly a new palette fades in. This method is useful when you want to change the palette to suit changing conditions in the movie without entering a new frame. For example, you can change the palette when you switch a cast member assigned to a sprite.

The new palette remains in effect until a new `puppetPalette` method is issued, a new palette is set in the palette channel, or a new movie starts.

For more information about this method, see the Scripting Reference topics in the Director Help Panel.

Solving color palette problems

When images in your movie appear with the wrong colors, you probably have the wrong color palette active. Color palette problems occur only if you are using 8-bit bitmaps and you want your movie to be displayed correctly on 256-color systems (8-bit bitmaps always appear correctly on computers that are set to display thousands or millions of colors).

Eight-bit bitmaps don't store information about actual colors; they identify colors by referring to positions in the current color palette. When saving an 8-bit bitmap, a graphics program creates a palette with the colors required for that particular image. This palette is saved with the file and must be active when the bitmap appears in a Director movie for the bitmap to appear with the proper colors. Only one palette can be active. Whenever it's necessary to display more than one 8-bit bitmap on the screen at one time, as is often the case in Director movies, all the images must refer to the same palette.

To solve color palette problems, follow these guidelines:

- To avoid color problems in movies for the web, map all 8-bit bitmaps in your movie to the Web216 color palette that is built in to Director. This is essentially the same palette used by Netscape Navigator and Microsoft Internet Explorer.
- Do not attempt to change palettes while a movie is playing in the browser. The browser, not the Director movie, controls the palette. Browsers ignore all palette channel settings.
- Make sure all the 8-bit images that are on the Stage at the same time refer to the same palette.
- If bitmaps are not dithering or remapping to the current palette, make sure that the Remap Palettes If Needed option on the Movie tab of the Property inspector is selected.
- Make sure there are no palette changes in the palette channel of which you are unaware. For example, when a cast member you are placing on the Stage has a palette different from the currently active palette, Director adds the new palette to the palette channel. If you don't realize that this has happened, you might find the palette changing unexpectedly when the movie plays.
- For disk-based movies, simplify your work and avoid frequent palette changes by mapping all the images in your movie to as few palettes as possible.
- Remap existing cast members to a new color palette using the Modify > Transform Bitmap command.
- If the Import option for Palette in the Image Options dialog box is not available while you are importing an image, the image's palette might not meet standard system requirements.

  Note: Use an image editor to make sure the image's palette meets the following requirements: The palette must contain exactly 16 or 256 colors. The first and last colors in the palette must be black or white, and there must be only one black and one white in the entire palette.

- Don't change colors that are used by your system software for interface elements. In Windows, these colors always appear as the first ten and the last ten colors in the palette.

Setting palette cast member properties

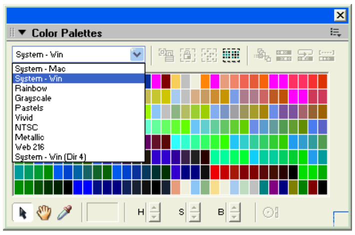
When you create a color palette in the Color Palettes window or import a bitmap with its own palette, the palette appears in a cast as an ordinary cast member. You use cast member properties to name the palette and to specify how it is unloaded from memory.

To create a color palette as a cast member:

1 If the Cast window is not already open, select Window > Cast.

2 Select Window > Color Palettes.

The Color Palettes window appears.

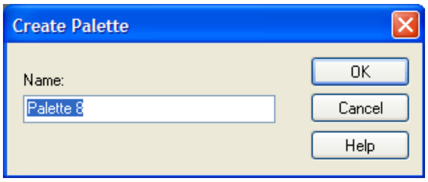


3 From the Palette pop-up menu, select the palette that you want to use to create a new palette cast member.

4 Double-click any color within the palette.

Director makes a copy of the palette and prompts you to enter a name.

5 In the Create Palette dialog box, enter a name and click the OK button.



Palette 8

6 The Color dialog box opens. Make any changes you want and click the OK button.

7 The new color palette appears in the Cast window bearing the name you gave it.

To view or change color palette cast member properties:

1 Select a color palette cast member.

2 To display the Property inspector, select Modify > Cast Member > Properties, or select Window > Property Inspector.

3 If necessary, click the Member tab and display the Graphical mode.

The following non-editable settings appear:

- The cast member size in kilobytes
- The cast member creation and edit dates
- The name of the last person who modified the cast member

4 To view or edit the cast member name, use the Name text box.

5 To add comments about the cast member, use the Comments text box.

6 To specify how Director removes the cast member from memory if memory is low, select one of the following options from the Unload pop-up menu:

3–Normal sets the selected cast members to be removed from memory after any priority 2 cast members have been removed.

2–Next sets the selected cast members to be among the first removed from memory.

1–Last sets the selected cast members to be the last removed from memory.

0–Never sets the selected cast members to be retained in memory; these cast members are never unloaded.

7 To modify the colors in the palette, click Edit.

About tempo

Tempo is the number of frames per second that Director tries to play. You can control tempo by using the Score tempo channel or the `puppetTempo` method.

Director tempo settings control the maximum speed at which the playhead moves from frame to frame. The tempo doesn't affect the duration of any transitions set in the transition channel, nor does it control the speed at which a sound or digital video plays. Tempo settings don't always control animated GIFs; for more information, see "Using animated GIFs" on page 101.

Settings in the tempo channel can also make a movie pause and wait for a mouse click or key press. For information about making a movie wait for a cue point in a sound or video, see "Synchronizing media" on page 240.

For simple movies, using the tempo channel is often the best way to define tempos. For more sophisticated control of the speed of a movie, use the `puppetTempo` method to control tempo.

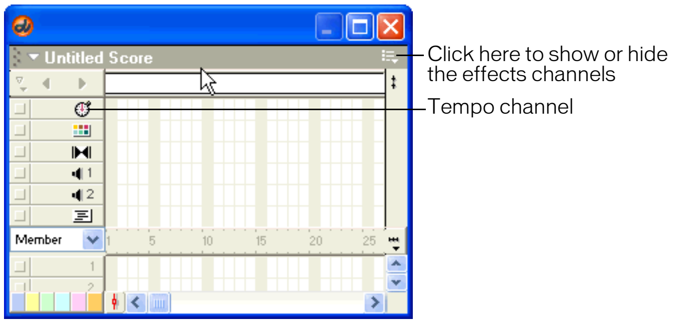
You can't make a movie go faster than the computer allows. Many factors can make movies play more slowly than the specified tempo, such as the following:

- Playing the movie on a slower computer
- Making the movie wait for cast members to download from a slow Internet connection
- Animating several large sprites at the same time
- Animating stretched sprites
- Color depth differences between the movie and monitor
- Animating sprites that have blend values

Specifying tempo properties

It's best to begin a movie with a tempo setting in the first cell of the tempo channel. If you don't set a tempo until later in the movie, the beginning tempo is determined by the setting in the floating Control panel. Director plays a movie at the tempo you've set until it encounters a new tempo setting in the tempo channel or a `puppetTempo` method is issued.

Enter tempo changes in the tempo channel at the top of the Score. (If you don't see the tempo channel, the effects channel is hidden. To display it, click the Hide/Show Effects Channel tool in the upper right of the Score window.)



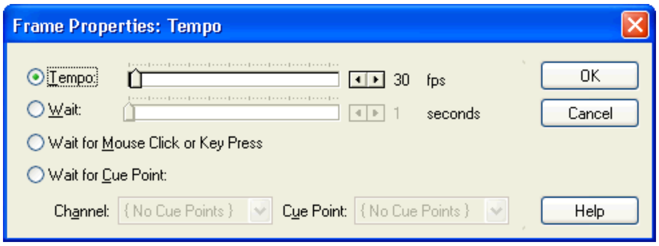
To specify a tempo setting:

1 In the Score, do one of the following:

- Double-click the cell in the tempo channel where you want the new tempo setting to appear.
- Right-click (Windows) or Control-click (Macintosh) the cell in the effects channel where you want the new tempo setting to appear, and then select Tempo from the Context menu.
- Select a frame in the tempo channel, and select Modify > Frame > Tempo.

If you don't see the tempo channel, the effects channel is hidden. To display it, click the Hide/Show Effects Channel tool in the upper right of the Score window.

2 Select the option you want to use in the Frame Properties: Tempo dialog box.



Frame Properties: Tempo

- [✓ Tempo:] 30 fps
- Wait: 1 seconds
- Wait for Mouse Click or Key Press
- Wait for Cue Point:

Channel: { No Cue Points } Cue Point: { No Cue Points }

- To set a new tempo for the movie, select Tempo, then use the Tempo arrows or drag the slider.
- To pause the movie at the current frame for a certain length of time, select Wait, then use the Wait arrows or drag the slider.
- To pause the movie until the user clicks the mouse or presses a key, select Wait for Mouse Click or Key Press.
- To pause the movie until a sound or digital video cue point passes, select Wait for Cue Point and select a channel and cue point. For more information, see "Synchronizing media" on page 240.

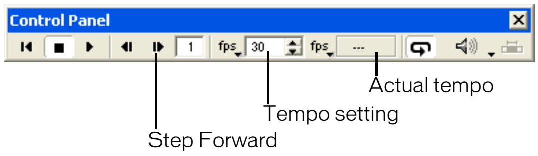
3 Click OK.

A number that matches the setting you've selected appears in the tempo channel. If you can't read the number, you might need to zoom the score. To do so, click the Zoom Menu button at the right edge of the sprite channel, or select View > Zoom. Then select a percentage from the pop-up menu.

Comparing actual speed with tempos

It's good practice to test the performance of your movie on a system that is similar to that of your users. Make sure the movie plays well on the slowest systems likely to be used.

The tempo you've set and the actual speed of a movie both appear in the floating Control panel.



Note: The Control panel attached to the bottom of the Stage does not include tempo settings. Tempo settings are available only from the floating (detached) version of the Control panel. To detach the Control panel from the Stage, Right-click (Windows) or Control-click (Macintosh) the Control panel. In the context menu, select Detach Control Panel.

To compare the actual speed of a movie with the tempos you've set:

1 Play the movie from start to finish, and then rewind it to the beginning.

2 Use the Step Forward button to step through the movie frame by frame.

3 In each frame, compare the tempo setting shown in the floating Control panel with the actual speed shown there.

   If you haven't recorded the actual speed of a movie in a particular frame, the actual tempo field displays two dashes (--).

Locking frame durations

To make Director play a movie at the same tempo on all types of computers, use the Lock Frame Durations option in the Movie Playback Properties dialog box (see "Setting movie playback options" on page 471). For frames without tempo settings, Director uses the current tempo. Lock Frame Duration prevents a movie from playing too fast on a fast system, but it cannot prevent a movie from playing slowly on a slow system.

To turn on Lock Frame Durations:

1 Select Modify > Movie > Playback.

2 Select Lock Frame Durations.

Controlling tempo with Lingo or JavaScript syntax

To override the tempo set in the movie's tempo channel, you use the `puppetTempo()` method. This approach is useful when you want to change the movie's tempo in response to conditions that you can't control, such as the type of computer the movie is playing on or a user's action.

The `puppetTempo()` method doesn't retain control of the tempo channel. If the movie encounters any tempo settings in the tempo channel, the `puppetTempo()` settings are overridden.

For more information about `puppetTempo()`, see the Scripting Reference topics in the Director Help Panel.

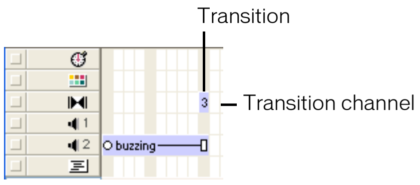
Using transitions

Transitions create brief animations that play between frames to create a smooth flow as sprites move, appear, or disappear or as the entire Stage changes. Director provides dozens of transitions built into the application, and many third-party Xtra extensions also include transitions. For example, you can dissolve from one scene to the next, display a new scene strip by strip, or switch to a scene as though revealing it through venetian blinds. You can also use many of the transitions to make individual elements appear or disappear from the screen.

After they are defined, transitions appear in the Cast window as cast members. You can place them in the transition channel by dragging them from the cast to the Score.

Creating transitions

The same as tempos, palettes, sounds, and behaviors, transitions have a channel set aside for them in the Score.



A transition always takes place between the end of the current frame and the beginning of the frame where the transition is set. If you want to create a dissolve between two scenes, set the transition in the first frame of the second scene, not in the last frame of the first scene.

To add a transition:

1 In the transition channel, select the frame in which you want the transition to occur.

2 Select Modify > Frame > Transition, or double-click the frame in the transition channel.

3 In the Frame Properties Transition dialog box, select a category if desired, and then select the transition you want. You can quickly scroll through transitions by typing the first letter of the transition's name.

   Many transitions have default settings for Duration and Smoothness. You can adjust the sliders to change the settings.

   For many transitions, you can also select whether the transition affects the entire Stage or only the area that's changing.

   Xtra transitions might offer additional options provided by the developer. If the Options button is available when you select an Xtra transition, click it to view and change the transition options.

4 Click OK.

   Director displays the cast member number that corresponds to the transition in the transition channel. The transition also appears in the cast.

Tips for using transitions

Here are some points to remember when working with transitions:

- To play a sound while a transition occurs, place the sound in the frame immediately before the transition.
- The Dissolve Pixels, Dissolve Pixels Fast, or Dissolve Patterns transitions might look different on Windows and Macintosh systems. Test to ensure satisfactory results.
- If you export a movie that contains transitions as a digital video or PICS file, the transitions might not be preserved.
- A transition that occurs while a sound or digital video is decompressing might require more system resources than are available on less powerful systems. This might cause the sound to stop playing. If you notice this behavior while testing on low-end systems, try making the transition shorter, and avoid complex transitions such as Dissolve.
- Avoid looping on a frame that contains a transition. Playing a transition continuously might cause performance issues.
- Options become available only when transition Xtra extensions are available.

Using transition Xtra extensions

You can add custom transitions that are available as transition Xtra extensions. Transition Xtra extensions appear in the Frame Properties: Transitions dialog box. Transition Xtra extensions are often more complex than the transitions that are provided with Director and might include an additional dialog box for specialized settings.

To install a transition Xtra:

- Place the transition Xtra in the Xtras folder in the Director application folder. The transition Xtra must be present when the movie runs.

Controlling transitions with Lingo or JavaScript syntax

To set a transition with script, you use the `puppetTransition()` method. This method gives you the flexibility to select a transition that is appropriate for current movie conditions or to apply a transition to sprites before the playhead exits the current frame.

For example, use the `puppetTransition()` method to specify one of several transitions, depending on which sprites are on the Stage when the playhead enters a new frame, or apply a transition to a new sprite when it appears but the playhead doesn't exit the frame.

The `puppetTransition()` method applies only to the frame in which you issue the method. You do not need to explicitly return control of the transition channel to the Score after the transition occurs.

The `puppetTransition()` method's parameters perform the same functions as the options in the Frame Properties: Transition dialog box.

For more information about `puppetTransition()`, see the Scripting Reference topics in the Director Help Panel.

Setting transition cast member properties

You use the Property inspector to set values for the transition cast member.

To view or change transition cast member properties:

1 Select a transition cast member.

2 To display the Property inspector, select Modify > Cast Member > Properties, or select Window > Property Inspector.

3 If necessary, click the Member tab and display the Graphical mode.

The following noneditable settings appear:

- The cast member size in kilobytes
- The cast member creation and edit dates
- The name of the last person who modified the cast member

4 To view or edit the cast member name, use the Name text box.

5 To add comments about the cast member, use the Comments text box.

6 To specify how Director removes the cast member from memory if memory is low, select one of the following options from the Unload pop-up menu:

3–Normal sets the selected cast members to be removed from memory after any priority 2 cast members have been removed.

2–Next sets the selected cast members to be among the first removed from memory.

1–Last sets the selected cast members to be the last removed from memory.

0–Never sets the selected cast members to be retained in memory; these cast members are never unloaded.

7 If you are using an Xtra transition, click Options to set values that are specific to the Xtra transition. The contents of the Options dialog box is determined by the developer of the Xtra. Refer to any documentation supplied with the Xtra.

CHAPTER 8
Text

Macromedia Director MX 2004 creates text that is editable, anti-aliased, and compact for fast downloading of outline fonts on both the Macintosh and Windows platforms. Combine these features with any of the animation capabilities of Director, such as rotation, and you can create wonderful text effects in your Director movies.

You can embed fonts in a movie to ensure that text appears in a specific font when a movie is delivered, regardless of which fonts are available on the user's computer.

Because Director renders text in the display font, and anti-aliases, or smooths, it as the movie plays, text in Director is very compact and downloads quickly from the Internet. Most of the high-quality text you see in web browsers is actually a GIF or JPEG graphic, and takes longer to download than Director text.

Director provides many ways to add text to a movie. You can either create new text cast members within Director or import text from an outside source such as a document stored on the Internet. You can import plain text, RTF, or HTML documents. After text is part of your movie, you can format the text in a variety of ways by using the Director formatting tools. Director offers standard professional formatting functions, including alignment, tabs, kerning, spacing, subscripts, superscripts, color, and so on. You can also create hypertext links for any text.

Text in Director is editable when you are working on your movie and, optionally, while a movie plays.

You can also script in Lingo or JavaScript syntax to control text. For example, you can use script to edit the text in existing cast members, specify text formatting such as font and size, and interpret strings that users enter.

To create the smallest possible text cast members, use field text. Field text is standard text controlled by your system software, the same as the text you see in dialog boxes and menu bars. Director does not anti-alias field text or support paragraph formatting and tabs for fields. As with regular text, script can control field text and specify whether field text is editable while a movie plays.

Whereas regular text is best suited for large type that you want to look as good as possible, field text is an excellent choice for large blocks of smaller text in standard fonts (such as Times or Helvetica) that don't need to be anti-aliased.

Embedding fonts in movies

Before creating text or field cast members, it's good practice to embed the fonts you want to use in the movie. Embedding fonts makes Director store all font information in the movie file so that a font displays properly even if it is not installed in a user's system. Because embedded fonts are available only to the movie, there are no legal obstacles to distributing fonts in Director movies.

Embedded fonts appear in a movie as cast members and work on Windows and Macintosh computers. Director compresses embedded fonts so they usually only add 14 to 25K to a file.

For the best display at smaller sizes, include bitmap versions of a font when you embed a font. For small font sizes, usually from about 7 to 12 points, bitmap fonts often look better than anti-aliased outline fonts (for more information, see "About anti-aliased text" on page 170). Adding a set of bitmap characters does, however, make the font cast member larger. Examine the text display quality of your movie to find out if this option is worthwhile.

To speed up movie downloading, you can keep a file size small by specifying a subset of characters to include. You can also specify which point sizes to include as bitmaps and which characters to include in the font package. If you do not embed fonts in a movie, Director substitutes available system fonts.

If you create embedded fonts by using the original font name followed by an asterisk (*), such as Arial* for the Arial font, Director uses the embedded font for all the text in the movie that uses the original font. This saves you the trouble of manually reapplying the font to all the text in existing movies.

After you embed a font in a movie file, the font appears on all the movie's font menus, and you can use it as you would any other font.

To embed a font in a movie:

1 Select Insert > Media Element > Font.

2 From the Original Font pop-up menu, select a font that is currently installed on your system.

   You can't embed a font that is not installed on your system. In other words, only fonts that appear in the Original Font pop-up menu are available to be embedded.

   In the New Font Name text box, the name of the font is followed by an asterisk (*). This name appears on all font menus in Director. In most cases, you should not change the name of a font.

3 To include bitmap versions of the font in specified sizes, click the Sizes button for Bitmaps, and enter the point sizes you want to include, separated by spaces or commas. For example, you might enter **9, 10, 14**.

4 To include bitmap versions of bold or italic characters with the font, select Bold or Italic.

   This option provides better-looking bold and italic fonts if you are including a bitmap version of the font, but it increases the file size.

5 To specify the characters that the font includes, select an option for Characters:

   Entire Set includes every character (symbols, punctuation, numbers, and so on) with the font.

   Partial Set lets you select exactly which characters are included. To select a group of characters, select Punctuation, Numbers, Roman Characters, or Other. If you select Other, enter the characters to be included in the text box on the right. In some double-byte languages, other groups of characters might appear.

To embed a font in a movie with Lingo or JavaScript syntax:

- Use the `recordFont` method. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

Creating text cast members

You can create text within Director or import text from external files.

Creating text in Director

Director provides two ways to create text cast members: directly on the Stage or in the Text window.

To create text cast members directly on the Stage:

1 Select the Text tool in the Tool palette.

Note: The Text tool is available when the Tool palette is in Classic or Default view. When in Flashcomponent mode, the tool becomes a textInput Flash component.

2 Drag the text pointer across the Stage and release the mouse button to create a text cast member.

You can't adjust the height of the text object at this point; the height adjusts automatically when you add text.

When you release the mouse button, a text insertion point appears in the area you defined.

3 Enter text.

The new text cast member appears in the first available position in the current cast, and the sprite is placed in the first open cell in the current frame in the Score.

To create text cast members in the Text window:

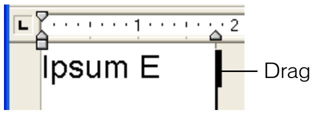
1 Select Insert > Media Element > Text.

If the Text window is already open, click the New Cast Member button to create a new text cast member.

2 Enter text in the Text window.

The text you enter appears in the first available cast position, but it is not automatically placed on the Stage.

3 To change the width of the cast member, drag the bar at the right edge of the cast member.



Importing text

You can import text from any application that saves text in rich text format (RTF), in plain text (ASCII), or from HTML documents. Use the standard importing procedure with File > Import to import any RTF, ASCII, or HTML document. To import an HTML document from the Internet, click the Internet button in the Import dialog box (File > Import) and enter a URL in the File URL text box.

Text and RTF files are always imported and stored inside the movie file, even if you select Link to External File.

When you import text from an HTML document, Director recognizes many standard tags and parameters, including tables, and approximates the formatting. Director does not recognize embedded objects other than tables, and it does not support nested tables. It also does not recognize `APPLET`, `FORM`, `FRAME`, `INPUT`, or `IMAGE` tags.

Director ignores any tags it does not recognize. For HTML files that are updated frequently, make sure you're satisfied with the formatting when importing.

When you import text from an RTF file, Director recognizes most standard RTF formatting, but it does not import pictures embedded in the file.

The amount of text in a cast member is limited only by the memory that is available in the playback system.

Importing text with Lingo or JavaScript syntax

You can import text in several ways by scripting in Lingo or JavaScript syntax:

- To import text from a URL, use the `getNetText()` method. For more information about this method, see the Scripting Reference topics in the Director Help Panel.
- To import text from an external file from a URL or the local computer, select or create a text cast member and set its `fileName` property to the name of the external file that contains the text. For more information about this property, see the Scripting Reference topics in the Director Help Panel.
- To import text from a file on disk, use the `getPref()` method. If no `setPref` method has already written such a file, the `getPref()` method returns `VOID`. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

Editing and formatting text

Director offers several ways to edit and format text. You can edit text directly on the Stage and format it with the Text inspector, or use the Text window to work in a more traditional text editing environment. Many of the same formatting controls are in the Font and Paragraph dialog boxes, as well as in the Text window and the Text inspector. Select the most convenient option for your work style.

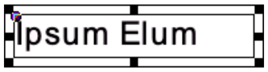
Selecting and editing text on the Stage

For basic text editing, it's fastest to edit text directly on the Stage.

To edit text on the Stage:

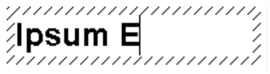
1 Click a text cast member on the Stage to select it as a sprite.

The text sprite appears as a normal sprite with double borders.



2 Click twice to edit the text.

An insertion point appears in the text, and you can begin editing.



3 Use the Text inspector (Window > Text Inspector) to reformat the text.

You can also use the Modify > Font and Modify > Paragraph commands to reformat selected text.

When you make a change, Director updates all sprites that display the text cast member.

Note: If you're changing the background color of text, you have two options. To change the background color of the cast member, double-click the text sprite on the Stage and assign a value from the Color box on the Tool palette. You can also tint the sprite's background, which blends the background color of the cast member with the background color of the sprite. To apply this effect, select the sprite, and select a background color on the Property inspector's Sprite tab.

To edit text on the Stage during playback:

1 Select a text sprite, and click Editable on the Property inspector's Sprite tab. For more information, see "Displaying and editing sprite properties in the Property inspector" on page 59.

2 Begin playing back the movie.

3 On the Stage, double-click to edit the text.

Formatting characters

After you create text cast members for your movie, you can format them in several ways: You can set the font, style, size, line spacing, and color. The following procedure uses the Font dialog box, but many of the same options are available in the Text inspector and the Text window.

To format characters:

1 Double-click a text sprite or cast member.

2 Select Modify > Font to open the Font dialog box.

3 Select from the following options in the Font dialog box:

- To specify the font, select a font from the list of available fonts. Be sure to use embedded fonts for movies that you intend to distribute (for more information, see "Embedding fonts in movies" on page 164).
- To use bold, italic, underline, superscript, subscript, or strikeout for text, click the appropriate box.
- To change the point size of text, increase or decrease the size with the Size option.
- To change the distance between lines of text, increase or decrease the spacing with the Spacing option.
- To specify kerning between selected characters, use the Kerning option to specify the number of points. This setting supplements the standard kerning that is applied to the entire cast member in the Text tab of the Property Inspector. For more information, see "About kerning" on page 170.
- To change the text color, click the Color box and select a color from the Color menu.

Formatting paragraphs

You can specify the alignment, indentation, tabs, and spacing for each paragraph in a text cast member. The following procedure explains how to format paragraphs while you work in the Text window, but many of the same formatting options are available in the Text inspector and the Paragraph dialog box.

To make formatting changes to a paragraph:

1 Double-click the text sprite in the Score or the text cast member in the Cast window to open the Text window.

2 If the ruler is not visible, select View > Rulers.

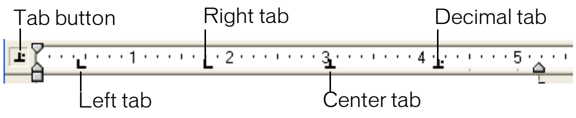
   To change the unit of measure on the text ruler, select Edit > Preferences > General, and select Inches, Centimeters, or Pixels from the Text Units pop-up menu.

   Note: If you are using a Macintosh OS X operating system, select the Director menu, instead of the Edit menu, to access Preferences.

3 Place the insertion point in the paragraph you want to change, or select multiple paragraphs.

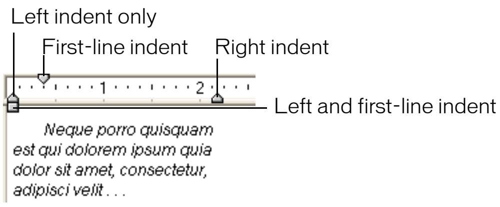
4 To define tabs, use any of the following options:

   - Set a tab by clicking the Tab button until the type of tab you want appears. Then click the ruler to place the tab.

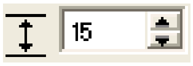


   - Move a tab by dragging the tab marker on the ruler.
   - Remove a tab by dragging the tab marker up or down off the ruler.

5 To set margins, drag the indent markers on the ruler.


Left indent only
First-line indent
Right indent
Left and first-line indent
Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit . . .

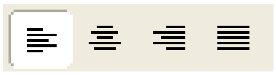
6 To set line spacing, change the setting with the Line Spacing control.



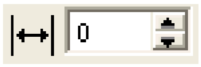
   Director adjusts line spacing to match the size of the text you are using.

   If you change the line spacing setting, Director stops making automatic adjustments. To resume automatic adjustments of spacing, enter `0` in the Line Spacing text box.

7 To set paragraph alignment, click one of the alignment buttons.



8 To change the kerning of selected characters, change the value of the Kerning option.



9 Set spacing before and after paragraphs by selecting Modify > Paragraph and by using the Spacing Before and After options.

Formatting entire cast members

Director can apply formatting changes to entire cast members. This process is much faster than manually opening each cast member and applying changes. Any change you apply to a cast member affects all the text within the cast member.

To format text cast members:

1 In a Cast window or on the Stage, select the cast members you want to change.

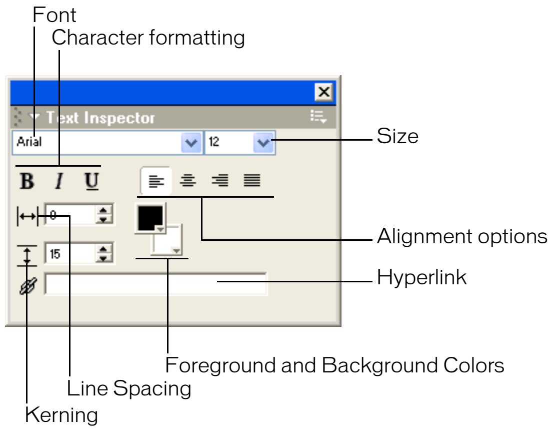
  You can select as many cast members as you want to change.

2 Use the Text inspector, Modify > Font, or Modify > Paragraph to make formatting changes.

  The change affects all the text in the selected cast members.

Formatting with the Text inspector

The Text inspector provides many useful formatting controls in a compact window for use on the Stage or with entire cast members in the Cast window.



Most of the formatting controls also appear at the top of the Text window and in the Font and Paragraph dialog boxes.

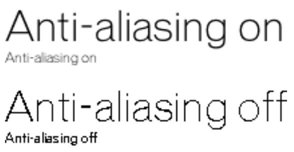
To display the Text inspector:

- Select Window > Text Inspector, or press Control+T (Windows) or Command+T (Macintosh).

About anti-aliased text

Anti-aliased text is text that uses color variations to make its jagged angles and curves look smoother. Director activates anti-aliasing by default. You can change this setting on the Text tab of the Property inspector (for more information, see "Setting text or field cast member properties" on page 174). Anti-aliasing functions the same way for embedded fonts and for system fonts that have not been embedded (see "Embedding fonts in movies" on page 164).

Using anti-aliased text dramatically improves the quality of large text on the Stage, but it can blur or distort smaller text. Experiment with the size settings to get the best results for the font you are using.



Director can anti-alias all outline (TrueType, PostScript, and embedded) fonts but not bitmap fonts. When you select a font that can't be anti-aliased, the message "This font can't be anti-aliased" appears in the Font dialog box below the font list. (Display the Font dialog box by selecting text or a text sprite and then selecting Modify > Font.)

About kerning

Kerning is a specialized form of spacing between certain pairs of characters that look best when they overlap slightly, such as A and W (AW). Kerning dramatically improves the appearance of large text for headlines, but it often does not improve the appearance of text at small font sizes.

If the Kerning option is selected on the Property inspector's Text tab, Director kerns all the characters in the cast member according to standard kerning tables (for more information, see "Setting text or field cast member properties" on page 174). The setting you enter in the Kerning text box in the Text window or Font dialog box (for more information, see "Formatting characters" on page 167) supplements the standard kerning.

Finding and replacing text

You can use the Find > Text command to quickly search for and replace text in the Text, Field, or Script window. All searches start at the insertion point and search forward.

To search and replace text:

1 Select Window > Text, Window > Field, or Window > Script to open the window in which you want to search.
2 Place the insertion point at the position where you want the search to begin.
3 Select Edit > Find > Text.
4 Enter the text for which you want to search in the Find box.
5 Enter the text you want to use in place of the found text in the Replace box.

6 To specify the cast members in which to search, select one of the following Search options:

Cast Member [Cast Member Name] limits the search to the current cast member.

Cast [Cast Name] limits the search to cast members in the current cast.

All Casts extends the search to all cast members in all casts.

7 To set additional search options, select Wrap-Around, Whole Words Only, or Case Sensitive.

Wrap-Around specifies whether Director returns to the beginning of text after it reaches the end. If you select this option but not All Casts, Director continues searching from the top of the current text after it reaches the bottom of the window. If you select both options, Director searches all cast members of the same type (either text, field, or script, depending on where you initiated the search), beginning with the currently selected cast member and returning to the first cast member of that type if necessary.

Whole Words Only searches only for occurrences of the specified whole word.

Case Sensitive searches only for text with the same capitalization as the text in the Find box.

Creating a hypertext link

In the Text inspector, you can turn any selected range of text into a hypertext link that links to a URL or initiates other actions. Director automatically adds standard hypertext link formatting to the selected text so that it initially appears with blue underlining. You can turn off this formatting in the Property inspector's Text tab. For more information, see "Setting text or field cast member properties" on page 174.

The following procedure describes how to add a hypertext link to selected text. To make a hypertext link active, you must write an `on hyperlinkClicked` event handler. For more information, see the Scripting Reference topics in the Director Help Panel.

You can enter any string in the Hyperlink text box; it does not have to be a URL. The string can't contain a double quotation mark or the script continuation character.

To define a hypertext link:

1 Select the text you want to define as a hypertext link.

2 Select Window > Text Inspector to open the Text inspector.

3 In the Hyperlink Data text box, enter the URL to which you want to link, or enter any message you want to send to the `on hyperlinkClicked` handler. Then press Enter (Windows) or Return (Macintosh).

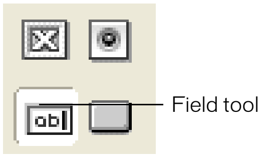
Working with fields

Working with field cast members is similar to working with text. As with text cast members, you edit fields on the Stage or in a window, and apply formatting with the Text inspector. Not all text formatting options are available for fields: you can't apply spacing, tabs, or indents to individual paragraphs within fields. Alignment settings apply to every paragraph in the field.

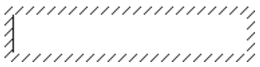
To create a field cast member:

1 Perform one of the following actions:

- Select Insert > Control > Field.
- While in Classic mode, click the Field tool in the Tool palette, and drag on the Stage to define the area of the field.



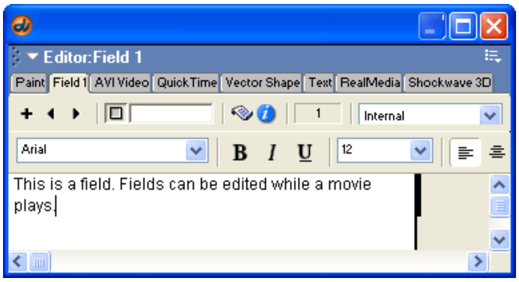
The field is created, and an insertion point is placed at the beginning of the field.



2 Enter the text for the field. When you finish, click outside the field to exit the field.

To specify field settings:

- Select Window > Field, or double-click a field cast member in the Cast window.



This is a field. Fields can be edited while a movie plays.

If necessary, use the Previous Cast Member and Next Cast Member buttons to navigate to the field you want to edit. For more information, see "Setting text or field cast member properties" on page 174.

Using editable text

Editable text lets users enter text on a web page, customize a game, and so on. When text is editable, editing the text changes the text cast member and all the text in the sprites where the cast member appears.

You can make text editable and let users tab between editable sprites from script or the Property inspector (for more information, see "Setting text or field cast member properties" on page 174).

You can make a text sprite editable in only a certain range of frames in the Score.

To make a text sprite editable in a range of frames:

1 Select a range of frames within a sprite.

You can select an entire sprite, or Shift-Alt-click (Windows) or Shift-Option-click (Macintosh) to select frames within a sprite.

2 Click the Property inspector's Text or Field tab by using the Graphical view.

3 Click Editable.

To control whether text is editable with Lingo or JavaScript syntax:

- Set the `editable` property. For more information about this property, see the Scripting Reference topics in the Director Help Panel.

To specify whether pressing the Tab key opens the next sprite for editing:

- Set the `autotab` property. For more information about this property, see the Scripting Reference topics in the Director Help Panel.

Converting text to a bitmap

You use the Convert to Bitmap command to change a text or field cast member to a bitmap. The converted graphic can then be edited in the Paint window. After you convert a cast member to a bitmap graphic, you can't undo the change.

This command works only with text and field cast members. You can't convert a shape to a bitmap.

To convert text to a bitmap:

1 In the Cast window, select the cast members to convert.

2 Select Modify > Convert to Bitmap.

Director converts the cast members to bitmaps.

Mapping fonts between platforms for field cast members

Director uses a file named Fontmap.txt to map fonts in fields between the Windows and Macintosh platforms. When you create a new movie, Director looks for Fontmap.txt in the same folder as the Director application.

The version of Fontmap.txt that is included with Director assigns fonts as shown in the following table. These settings provide the best equivalents of common system fonts on both platforms.

Windows font	Macintosh font
Arial	Helvetica
Courier	Courier
Courier New	Courier
MS Serif	New York
MS Sans Serif	Geneva
Symbol	Symbol
System	Chicago

Windows font	Macintosh font
Terminal	Monaco
Times New Roman	Times (because Times New Roman is larger than Times, Fontmap.txt assigns a smaller point size.)

Fontmap.txt also determines the scaling of fonts and how special characters such as bullets and symbols are translated between platforms. Again, the default settings are correct for nearly all applications, but you can edit the settings if necessary

You can also load and save fontmaps by using the Property inspector.

To save fontmaps:

1 Click on the Stage and select Window > Property Inspector. Make sure that the Property inspector is in graphical view.
2 Select the Movie tab from the Property inspector.
3 In the Font Map area near the bottom of the Movie tab, click the Save button.
4 A dialog box opens that prompts you to enter a name for the fontmap and to select a folder in which to save the fontmap file.

To load fontmaps:

1 Click on the Stage and select Window > Property Inspector. Make sure that the Property inspector is in graphical view.
2 Select the Movie tab from the Property inspector.
3 In the Font Map area near the bottom of the Movie tab, click the Load button.
4 A dialog box opens that prompts you to specify the font mapping file to load.
5 Enter a name for the fontmap or browse for the file, and click Open.

Setting text or field cast member properties

Use the Property inspector to view and change settings for selected text cast members. In addition to standard Name and Unload properties, you can specify whether text is editable while the movie plays, improve performance with pre-rendering, and control anti-aliasing and kerning.

To view or change text or field cast member properties:

1 Select a text cast member in the Cast window.
2 To display the Property inspector, perform one of the following actions:
   - Select Modify > Cast Member > Properties.
   - Select Window > Property Inspector.
3 Click the Member tab, if it's not already selected, using the Graphical view.

   The following noneditable settings appear:
   - The cast member size in kilobytes
   - The cast member creation and edit dates
   - The name of the last person who modified the cast member
4 To view or edit the cast member name, use the Name text box.
5 To add comments about the cast member, use the Comments text box.

6 To specify how Director removes the cast member from memory if memory is low, select an option from the Unload pop-up menu. For more information, see "Controlling cast member unloading" on page 47.

7 To change the text of the cast member, click Edit.

8 Click the Property inspector's Text or Field tab by using the Graphical view.

9 To determine how Director places text within the boundaries of the cast member, select one of the following Framing options:

Adjust to Fit expands the text box vertically when text that is entered extends beyond the current size of the box.

Scrolling attaches a scroll bar to the right side of the text box. This is useful when there is a large amount of text. The scroll bar is drawn direct to Stage, which means that even if another cast member is in front of a cast member that contains a scroll bar, the scroll bar appears in the front.

Fixed retains the original size of the text box. If you enter text that extends beyond the limits of the box, the text is stored but doesn't appear. You can set up scrolling with some script in Lingo or JavaScript syntax (for more information, see "Controlling scrolling text with Lingo or JavaScript syntax" on page 179).

Limit to Field Size (available only for field cast members) displays only the amount of text that fits within the field's bounding rectangle.

10 To set editing and display options, select from the following options:

Editable makes the cast member editable while the movie plays (for more information, see "Using editable text" on page 172).

Wrap increases the vertical size of the text box or field on the Stage so that all text is visible.

Tab advances the text insertion point to the next editable sprite on the Stage when the user presses Tab.

DTS (Direct to Stage) (text cast members only) makes text display more quickly by rendering it directly to the Stage without composing it with other sprites. Selecting the DTS option prevents other sprites from appearing over the text, and limits the ink options to Copy.

Use Hypertext Styles (text cast members only) makes hypertext links appear the same as in a web browser, initially using blue underlining, and then red after the link has been visited. (For more information, see "Creating a hypertext link" on page 171.)

11 To make text of a text cast member appear on the Stage more quickly, select a pre-render option. Pre-rendering controls when text buffers are created.

Without pre-rendering, large amounts of anti-aliased text can take a while to load. This can cause a noticeable delay when the text is displayed for the first time. When a pre-render option is selected, text buffers are created when the current text member is loaded, instead of when the member first appears on the Stage.

Select one of the following pre-render options from the Pre-Render pop-up menu:

None provides no pre-rendering.

Copy Ink optimizes the pre-rendering for Copy Ink (this option renders text more quickly than Other Ink).

Other Ink pre-renders the text for all other ink types.

If you select a pre-render option, you can make text appear on the Stage even more quickly by selecting Save Bitmap.See the next section.

12 To control how Director anti-aliases text for a text cast member, select one of the following Anti-Alias options:

All Text anti-aliases all the text in the text block.

Larger Than anti-aliases only text that is larger than the point size entered in the Points text box.

None turns off anti-aliasing for the current cast member.

Anti-aliasing dramatically improves the appearance of large text, but it can blur or distort smaller text. Experiment with the size setting to get the best results for the font you are using. (For more information, see "About anti-aliased text" on page 170.)

13 To control how Director kerns text, select a Kerning option.

Kerning often does not improve the appearance of text at small point sizes. For more information, see "About kerning" on page 170.

All Text kerns all the text in the cast member according to the standard kerning table.

Larger Than kerns only text that is larger than the point size entered in the Points text box.

None turns off kerning for the current cast member.

14 To add borders and shadings (field cast members only), select options from the Box Shadow, Border, Drop Shadow, and Margin pop-up menus.

Using the Save Bitmap feature for pre-rendered text

The Save Bitmap feature works with pre-render options to display a buffer image of your text while your user waits for the actual text to load. This feature is useful when you're working with a large amount of anti-aliased text. (The Save Bitmap feature is different from the Convert to Bitmap menu command, which converts a text cast member into a bitmap image.)

You can also use the Save Bitmap feature with pre-render options if you're using special text characters for an audience that is not equipped to display them. For example, using Save Bitmap enables a non-Japanese system to display a text sprite that contains Japanese characters. Using the Save Bitmap option adds to the file size. The feature works with static text but not with editable or scrolling text.

To use the Save Bitmap feature for pre-rendered text:

1 Select the text sprite.

2 On the Property inspector's Text tab, select from the Pre-Render pop-up menu:

- If the text sprite's ink is Copy Ink, select Copy Ink.
- If the text sprite's ink is any type of ink other than Copy Ink, select Other Ink.

For this procedure to work, you must make the correct selection from the Pre-Render pop-up menu. You can determine the Sprite's ink on the Sprite tab of the Property inspector.

3 Select Save Bitmap.

Formatting chunks of text with Lingo or JavaScript syntax

The Director interface lets you format a variety of text characteristics, such as the font, size, style, and line spacing. Using Lingo or JavaScript syntax, you can format text dynamically as the movie plays. You can also use script to rapidly format text during authoring.

Formatting text with Lingo or JavaScript syntax

Lingo or JavaScript syntax can format text in an entire cast member or any specific chunk of text. Use the following properties:

- To select or identify a chunk of text in a field cast member, use the `selStart` and `selEnd` cast member properties. These properties identify the first and last characters of a text selection.
- To refer to a selected chunk of text, use the `selection` cast member property.
- To specify the font for a text cast member, field cast member, or chunk expression, set the `font` cast member property.
- To specify the character size for a text cast member, field cast member, or chunk expression, set the `fontSize` property.
- To specify the line spacing for a field cast member, set the `lineHeight` property.
- To specify the style for a text cast member, field cast member, or chunk expression, set the `fontStyle` property.
- To specify the drop shadow size for the characters in a field cast member, set the `boxDropShadow` property.
- To specify additional spacing to be applied to a chunk expression in a text cast member, set the `charSpacing` property.
- To specify the foreground color for a field cast member, set the `foreColor` property.

For more information about these properties, see the Scripting Reference topics in the Director Help Panel.

Applying paragraph formats with Lingo or JavaScript syntax

Lingo or JavaScript syntax can control paragraph formatting such as alignment and indenting for a chunk expression:

- To set text alignment for a text or field cast member, set the `alignment` property.
- To set line spacing in points for a text cast member, set the `fixedLineSpace` property.
- To add pixels below paragraphs in a text cast member, set the `bottomSpacing` property.
- To add pixels above paragraphs in a text cast member, set the `topSpacing` property.
- To specify line spacing in a field cast member, set the `lineHeight` property.
- To add pixels to the first indent in a chunk expression in a text cast member, set the `firstIndent` property.
- To set the left indent (in pixels) of a chunk expression in a text cast member, set the `leftIndent` property.
- To set the right indent (in pixels) of a chunk expression in a text cast member, set the `rightIndent` property.
- To specify or obtain a list of tabs that are in a chunk expression in a text cast member, set or test the `tabs` property.

For more information about these properties, see the Scripting Reference topics in the Director Help Panel.

Formatting text or field cast members with Lingo or JavaScript syntax

In addition to formatting text in any chunk expression, Lingo or JavaScript syntax can specify anti-aliasing and kerning for an entire text cast member and control the appearance of the text's bounding rectangle.

Setting anti-aliasing and kerning with Lingo or JavaScript syntax

You can use Lingo or JavaScript syntax to specify anti-aliasing and kerning for a text cast member:

- To specify whether Director anti-aliases text in a text cast member, set the `antiAlias` cast member property.
- To specify the size at which anti-aliasing in a text cast member takes effect, set the `antiAliasThreshold` cast member property.
- To specify automatic kerning for a text cast member, set the `kerning` cast member property.
- To specify the size at which automatic kerning for a text cast member takes effect, set the `kerningThreshold` cast member property.

For more information about these properties, see the Scripting Reference topics in the Director Help Panel.

Formatting text boxes with Lingo or JavaScript syntax

Lingo or JavaScript syntax can specify the type of box that surrounds a text or field cast member. For field cast members, script can also specify box characteristics such as borders, margins, drop shadows, and height.

- To specify the type of box around a text or field, set the `boxType` cast member property.
- To specify the size of the border around a field, set the `border` field cast member property.
- To specify the size of the margin inside a field's box, set the `margin` field cast member property.
- To specify the size of the drop shadow for a field's box, set the `boxDropShadow` field cast member property.
- To specify the height of a field's box on the Stage, set the `pageHeight` field cast member property.

For more information about these properties, see the Scripting Reference topics in the Director Help Panel.

Setting text autotabbing and wrapping with Lingo or JavaScript syntax

Lingo or JavaScript syntax can set text autotabbing and wrapping.

- To specify autotabbing for text or field cast members, set the `autoTab` cast member property.
- To specify whether lines wrap in a field cast member, set the `wordWrap` cast member property.

For more information about these properties, see the Scripting Reference topics in the Director Help Panel.

Controlling scrolling text with Lingo or JavaScript syntax

You can script in Lingo or JavaScript syntax to scroll text and determine the location of specific text within the text box for text and field cast members. For example, this statement sets the `scrollTop` value for the text cast member called Discussion to 0, which makes its first line appear at the top of its scrolling field:

```
member("Discussion").scrollTop = 0
```

This procedure can be useful for making a scrolling field automatically scroll back to the top. For more information about the following methods and properties, see the Scripting Reference topics in the Director Help Panel.

- To scroll up or down by a specific number of pages in a text or field cast member, use the `scrollByPage` method.
- To scroll up or down by a specific number of lines in a text or field cast member, use the `scrollByLine` method.
- To determine the number of lines that appear in a field cast member on the Stage, set the `lineCount` cast member property. (This property doesn't apply to text cast members.)
- To determine a line's distance from the top edge of a field cast member, use the `linePosToLocV()` method.
- To determine the number of the line that appears at a specific vertical position in a field cast member, use the `locVToLinePos()` method. (This measures the distance from the top of the cast member, not what appears on the Stage.)
- To determine the point in a field cast member that is closest to a specific character, use the `charPosToLoc()` method.
- To determine the character that is closest to a specific point in a field cast member, use the `locToCharPos()` method.
- To check or set the distance from the top of the line that is currently visible to the top of the box for a scrolling field cast member, test or set the `scrollTop` cast member property.

Checking for specific text with Lingo or JavaScript syntax

The script operators `contains` and the equals symbol (=) are useful for checking strings. The `contains` operator compares two strings to see whether one string contains the other. The equals symbol operator can determine whether a string is exactly the same as the contents of a field cast member. Use these operators to check whether a specified string is in a field cast member. For more information, see the Scripting Reference topics in the Director Help Panel

You can also use script to evaluate strings returned by the `text` property of a text or field cast member. For more information about this property, see the Scripting Reference topics in the Director Help Panel.

Modifying strings with Lingo or JavaScript syntax

As time passes or other conditions change, you might want to update and change text. For example, you might want to frequently update a text sprite that displays the user's name or a description of a musical selection that the user is currently streaming from a website. For more information, see the Scripting Reference topics in the Director Help Panel.

- To set the entire content of a text or field cast member, set the `text` cast member property to a new chunk of text. The chunk can be a string or another text cast member.
- To combine character strings, use the `&` and `&&` operators. The `&` operator attaches the second string to the end of the first string. The `&&` operator includes a space between two strings when they are combined.
- To insert a string of characters into another string, use the `text` property. For example, `sprite(1).text = sprite(1).text + "someWord"`.
- To delete a chunk expression from a string of text, use the `delete` method.

CHAPTER 9
Using Flash, Flash Components, and Other Interactive Media Types

To add complex media and new capabilities to your Macromedia Director MX 2004 movie, you can use Macromedia Flash content (MX or later), Flash components, other Director movies, and ActiveX controls. Each of these multimedia formats has interactive capabilities that are preserved by Director.

Flash content in a Director movie provides a vector-based, scalable, interactive animation that is optimized for use on the web.

Director provides you with a set of Flash built-in components, which are movie clips with defined parameters. You can use these components to add user interface elements, such as buttons and check boxes to your movies. You can use these components and set properties and events without having Flash installed. For more information, see "Using Flash components" on page 207.

Director movies within other Director movies simplify complex productions. A linked movie appears within another movie as a single cast member, saving you the trouble of managing extra cast members and Score data. Using discrete movies also helps you manage file size for easier downloading.

ActiveX controls in Director can manage ActiveX application resources from within a movie. ActiveX controls provide a variety of features, including web browsing, spreadsheet functions, and database management. ActiveX controls function as normal sprites in a movie. ActiveX controls work only in Director for Windows and only in projectors.

Using Flash Content

You can incorporate Flash vector-based animation in your Director movies and projectors simply by importing Flash content into Director and using it like any other cast member. Effects that once required multiple versions of a bitmap cast member—such as blending one shape into another—can now be accomplished with a single, small Flash file.

Director can import Flash 2 files or later. Director supports the new features of Macromedia Flash MX and Flash MX 2004, including access to Macromedia Flash Communication Server MX.

In Director, you can control nearly every Flash property—including playing, rewinding, and stepping forward and backward through all Flash content, adjusting quality settings, and turning sound on or off—using Lingo or JavaScript syntax.

In Flash, you can create cross-platform Windows and Macintosh movies and then play or manipulate them in Director. You can create Flash content that communicates with your Director movie by sending events that Director scripts can capture and process. You can store entire Flash files in the Director cast file, or you can link to external Flash content. Director automatically loads the Flash content it encounters in the Score into memory from disk, from a network drive, or from anywhere on the Internet.

Flash content is particularly effective for use in Macromedia Shockwave content because, as vector-based media, they are extremely small and therefore load much more quickly than most other media types. Because Flash content is vector-based, you can scale and rotate them while still maintaining their sharpness. For example, you can create splash screens for your Shockwave content that loads quickly and entertains your users while the rest of the Director movie streams into memory, or you can create interactive maps in Flash that users can pan across or zoom in on to reveal details with vector-based precision.

Director MX 2004 uses the Flash asset commonPlayer. The commonPlayer is a property that applies to all Flash assets, Vector shapes, and Flash components. It lets you load multiple Flash sprites into one instance of the Flash Player rather than requiring one Flash Player for each Flash sprite on the Stage. The commonPlayer feature is designed to provide better Flash playback performance in Director projects that use large numbers of Flash assets.

Adding a Flash content cast member

All Flash cast members added to a Director movie must have been created with Flash 2.0 or later and saved in the Flash format (SWF).

The following procedure explains how to create a Flash cast member and set properties for it at the same time. You can also import a cast member by using the Import command or by dragging and dropping a SWF file to the Director Cast window.

To add Flash content as a cast member:

1 Select Insert > Media Element > Flash Movie.

2 In the Flash Asset Properties dialog box, select the Flash movie (SWF) file you want to add to your Director cast.

- To add a file from your computer or from a network drive, click Browse, select the file, and click Open. Director creates a relative link to the file, so it must maintain the same location relative to the Director file, or the link becomes invalid.
- To add a file from a location on the Internet, click Internet, type the URL of the file, and click OK.
- Type the pathname or the URL of the file in the Link File box.

3 Set Media options:

Linked leaves the actual media of the Flash content stored in an external file. When a sprite created with this cast member appears on the stage in a Director movie, Director automatically loads the file into memory by looking in the location specified in the Link File box. Deselect Linked to have Director copy the Flash content into the cast.

Preload requires Director to load the all the Flash content into memory before playing the movie's first frame. Deselect Preload to have Director start playing the movie immediately, while continuing to stream the cast member into memory. This option is only available if you selected Linked. If the Flash file is internal instead of linked, it must load completely into memory before beginning to play.

4 Select Playback options to control how a Flash content sprite plays in a Director projector, in Shockwave content, and while you are authoring in Director:

Image displays the image of the Flash content when it plays. When Image is deselected, the Flash movie is invisible.

Sound enables any sound in the Flash content to play. When Sound is deselected, the movie plays without sound.

Paused displays only the first frame of the movie without playing the movie. When Paused is deselected, the movie begins playing immediately when it appears on the Director Stage.

Loop makes the movie play again from frame 1 once it finishes. When Loop is deselected, the movie plays once and stops.

Direct to Stage displays the movie when it appears on the stage with the fastest, smoothest playback. Deselect Direct to Stage to have Director apply ink effects and perform compositing of the sprite with other sprites in a memory buffer before actually displaying it. The disadvantage of the Direct to Stage option is that the movie always appears on top of other sprites, regardless of the channel in which it appears in the Score, and ink effects don't work.

5 Specify a Scale value by typing the percentage to scale the cast member.

6 Specify a Quality value:

- Select a high-quality setting to have the Flash content play with anti-aliasing turned on, which slows down performance; select Auto-High to have Director start playing the movie with anti-aliasing on but turn it off if it can't play the movie at the required frame rate.
- Select a low-quality setting to turn off anti-aliasing but speed up performance; select Auto-Low to have Director start playing the movie without anti-aliasing but turn on anti-aliasing if it can do so and continue playing the movie at the required frame rate.

7 Select a Scale Mode value to control how the Flash content sprites are scaled on Stage:

Show All maintains the movie's aspect ratio and, if necessary, fills in any gaps along the horizontal or vertical dimension using the movie's background color.

No Border maintains the movie's aspect ratio by cropping the horizontal or vertical dimension as necessary without leaving a border.

Exact Fit stretches the movie to fit the sprite exactly, disregarding the aspect ratio.

Auto-Size adjusts the sprite's bounding rectangle to fit the movie when it is rotated, skewed, or flipped. This option always sets the scale to 100% in the Director score.

No Scale places the movie on the Stage with no scaling. The movie stays the same size no matter how you resize the sprite, even if it means cropping the movie.

8 Select a Rate value to control the tempo at which Director tries to play Flash content:

Normal plays the Flash content at the tempo stored in Flash content.

Fixed plays the movie at a rate you specify by typing a value in the entry box.

Lock-Step plays a frame of the Flash content for each Director frame.

Note: Flash content does not play faster than the frame rate set in the Director movie.

9 When you have finished selecting options, click OK.

Director adds the Flash content to the cast.

Note: You can also use Lingo or JavaScript syntax to adjust these and other properties of the Flash content. For more information, see "Controlling Flash content with Lingo or JavaScript syntax" on page 185.

About using Flash content in a Director movie

Once you have added Flash content to the Director cast, using it in your movie is as simple as dragging it to the stage and positioning it where you want it. You can then use the Flash content sprite in much the same way that you use other sprites.

When working with Flash content on the Stage, keep these points in mind:

- Flash content animation plays only as long as the Flash content sprite is actually on the Stage. (Flash content resembles digital video and sound sprites in this regard.)
- Because Flash content uses a vector graphics format, you can stretch the application's sprite without loss of clarity.
- You can rotate, skew, scale, or flip Flash content just as you would a vector shape or bitmap.
- If the movie is set up to play direct-to-Stage, the movie always appears on top of other sprites, regardless of the channel in which it is placed, and ink effects are ignored.
- Only the Copy, Transparent, Background Transparent, and Blend ink effects work with Flash content, and only when the sprite is not played Direct-to-Stage.
- Blend and color settings are supported for Flash sprites just as they are for vector shapes.

Editing a Flash cast member

If you have Macromedia Flash MX or later installed, you can launch the Flash authoring tool from within Director to edit your Flash cast members.

To edit a Flash cast member, do one of the following:

- Double-click the Flash cast member in the cast or a Flash sprite in the Score or on the Stage.
- Select Edit > Launch External Editor while a Flash cast member is selected.

If Flash MX or later is installed and the Flash source file (FLA) is specified in the Property inspector, Flash starts with the source file open for editing, and changes are immediately reflected in Director when the Flash file is saved in Flash. If the source file is not specified in the Property inspector, a file dialog box appears so you can locate the source file. If Flash MX or later is not installed, the Flash properties dialog box opens instead.

To specify a source file (FLA) path for a SWF cast member before initiating a launch and edit session:

1. Select the SWF cast member in the cast.
2. Open the Property inspector.
3. Select the Flash tab.
4. Enter the path in the Filename field or Click the Browse button to browse to the file.

If you install Flash MX or later before installing Director, Flash is added automatically to the list of external editors when you install Director. If you install Flash after installing Director, you can enable Flash launch and edit by adding Flash to the list of external editors.

To add Flash to the list of external editors:

1 Select Edit > Preferences > Editors.

   Note: If you are using a Macintosh OS X operating system, select the Director menu, instead of the Edit menu, to access Preferences.

2 Select Flash, and click Edit.

3 Select Use External Editor, and click Browse.

4 Browse to the location of your Flash application.

5 Select the Flash application file, and click Open.

6 Click OK, and then click OK again.

Flash starts when you double click a Flash cast member.

Controlling Flash content with Lingo or JavaScript syntax

Lingo or JavaScript syntax gives you precise control over the way Director streams and displays Flash content. You can use these scripts to check and control cast member streaming, zoom and colorize the Flash asset, and pan the Flash image.

When a movie is playing, Lingo or JavaScript syntax can change the Flash cast member's properties. Some cast member properties, such as the `flashRect` and `frameRate` cast member properties, are valid only after the Flash content header has streamed into memory.

Director provides the following Lingo or JavaScript syntax that lets you manage how Director uses Flash content. For more information, see the Scripting Reference topics in the Director Help Panel.

- To control whether changes to a Flash cast member immediately appear in sprites that use the cast member, set the cast member's `broadcastProps` property.
- To control whether Flash content is stored in an external file, set the `linked` property.
- To control which frame of Flash content that Director uses for the Flash contents thumbnail image, set the `posterFrame` property.
- To display a list of Flash content's current property settings in the Message window, use the `showProps` method.

Controlling Flash content appearance with Lingo or JavaScript syntax

Lingo or JavaScript syntax can control how a Flash content appears on the Stage and which part of the Flash content appears in its sprite's bounding rectangle. Lingo or JavaScript syntax can also skew, rotate, scale, and flip the Flash content.

Director supports only the Copy, Transparent, Background Transparent, and Blend inks for Flash sprites, and only when the sprite is not played Direct-to-Stage.

Flipping, rotating, and skewing Flash sprites

Lingo or JavaScript syntax can flip, rotate, and skew Flash sprites as the movie plays. For more information, see the Scripting Reference topics in the Director Help Panel.

- To flip a Flash sprite, set the `flipH` and `flipV` sprite properties.
- To skew a Flash sprite, set the `skew` sprite property.
- To rotate a Flash sprite, set the `rotation` property. Set the `obeyScoreRotation` property to specify whether a Flash sprite obeys the rotation specified in the Score.

  If `obeyScoreRotation` is set to `TRUE`, Director ignores the cast member's `rotation` property and obeys the Score rotation settings instead.

Colorizing and blending Flash sprites

You can use Lingo or JavaScript syntax to change a sprite's color and blend as the Director content plays. For more information, see the Scripting Reference topics in the Director Help Panel.

To specify the color of a Flash sprite:

- Set the `color` sprite property.

To specify the blend for a Flash sprite:

- Set the `blend` sprite property.

Scaling Flash content

You can use Lingo or JavaScript syntax to scale Flash cast members and sprites. For more information, see the Scripting Reference topics in the Director Help Panel.

To control the scaling of Flash content:

- Set the `scale` and `scaleMode` properties.

To set the scale percentage of Flash content within its sprite's bounding rectangle:

- Set the `viewScale` property.

Obeying the Flash content cursor settings

Flash content can be designed to use different cursors depending on which part of the Flash content the mouse pointer is rolled over. To allow a Flash sprite to use the cursor settings defined in the Flash content, attach the Flash Cursor behavior to the Flash sprite.

To write your own script to test the cursor settings of the Flash content, use the `getFlashProperty()` method and test for the `#cursor` property. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

Controlling the Flash content bounding rectangle and registration points

You can use Lingo or JavaScript syntax to control the Flash content bounding rectangle and to Flash content registration points. For more information, see the Scripting Reference topics in the Director Help Panel.

- To control which part of the Flash content appears within its sprite's bounding rectangle, set the `viewH`, `viewpoint`, `viewScale`, and `viewV` properties.
- To control the default size for all new Flash sprites, set the `defaultRect` property. Use the `defaultRectMode` property to control how the default size is set.
- To determine the original size of a Flash cast member, test the `flashRect` property.
- To specify the Flash content registration point around which scaling and rotation occurs, set the `originH`, `originMode`, `originPoint`, and `originV` properties.
- To center a Flash cast member's registration point after resizing the cast member, set the `centerRegPoint` property to `TRUE`.

Placing Flash content on the Stage

Lingo or JavaScript syntax can set whether Flash content appears at the front of the Stage and whether specific areas of Flash content and the Stage overlap. For more information, see the Scripting Reference topics in the Director Help Panel.

- To determine whether Flash content plays in front of all other layers on the Stage and whether ink effects work, set the `directToStage` property.
- To determine which Stage coordinate coincides with a specified coordinate in Flash content, use the `flashToStage()` method.
- To determine which Flash content coordinate coincides with a specified coordinate on the Director Stage, use the `stageToFlash` method.
- To improve performance for a Director movie that uses static (not animated) Flash content, set the `static` property.
- To control whether the Flash content graphics are visible, set the `imageEnabled` property.
- To control whether the Flash content plays sounds, set the `sound` property.
- To control whether Director uses anti-aliasing to render Flash content, set the `quality` property.

Streaming Flash content with Lingo or JavaScript syntax

In addition to the Lingo or JavaScript syntax that lets you stream many of the Director media types, Director lets you control and monitor streaming Flash content. For general information about using script to stream media in Director, see Chapter 24, "Using Shockwave Player," on page 469. For more information about specific Lingo or JavaScript syntax methods and properties, see the Scripting Reference topics in the Director Help Panel.

- To specify whether a linked movie streams or not, set the `preLoad` property.
- To specify how much of a Flash cast member streams into memory at one time, set the `bufferSize` cast member property.
- To check how many bytes of Flash content have streamed into memory, test the `bytesStreamed` property.

- To determine how much of Flash content is currently streamed, test the `percentStreamed` property or check the `streamSize` method.
- To set when Director attempts to stream part of Flash content, set the `streamMode` property.
- To clear an error setting for streaming Flash content, use the `clearError` method.
- To determine whether an error occurred while streaming Flash content, use the `getError()` method.
- To check the current state of a streaming file, test the `state` property.
- To attempt to forcibly stream a specified number of bytes of Flash content, use the `stream` method.

Playing back Flash content with Lingo or JavaScript syntax

Lingo or JavaScript syntax lets you control how Flash content plays back and whether the Flash content retains its interactivity.

Controlling Flash content playback with Lingo or JavaScript syntax

You can use Lingo or JavaScript syntax to control the Flash content tempo, to specify which frame plays, and to start, stop, pause, and rewind the Flash content. For more information, see the Scripting Reference topics in the Director Help Panel.

- To control the tempo of Flash content, set the `fixedRate` and `playBackMode` properties.
- To determine the original frame rate of Flash content, test the `frameRate` property.
- To determine the number of frames in Flash content, test the `frameCount` property.
- To determine the frame number associated with a label in Flash content, use the `findLabel()` method.
- To play Flash content starting from a specified frame, set the `frame` property or use the `goToFrame()` method.
- To set whether Flash content starts playing immediately when the Flash sprite appears on the Stage, set the `pausedAtStart` property.
- To check whether Flash content is playing or paused, test the `playing` property.
- To rewind Flash content to frame 1, use the `rewind sprite` method.
- To stop Flash content at its current frame, use the `stop` method.
- To stop Flash content at its current frame but let any audio continue to play, use the `hold` method.
- To specify a separate time line within a Flash cast member as the target of subsequent script sprite methods, use the `tellTarget()` and `endTellTarget()` methods.
- To call a series of actions that reside in a frame of a Flash application sprite, use the `callFrame()` method.

Controlling Flash content interactivity with Lingo or JavaScript syntax

Lingo or JavaScript syntax can control whether Flash content remains interactive. For more information, see the Scripting Reference topics in the Director Help Panel.

- To control whether the actions in Flash content are active, set the `actionsEnabled` property to `TRUE`.
- To control whether buttons in Flash content are active, set the `buttonsEnabled` property.
- To control when Flash content detects mouse clicks or rollovers, set the `clickMode` property.
- To control whether clicking a button in Flash content sends events to sprite scripts, set the `eventPassMode` property.
- To determine which part of Flash content is directly over a specific point on the Stage, use the `hitTest` method.
- To check whether the mouse pointer is over a button in Flash content, test the `mouseOverButton` property.

Using Lingo or JavaScript syntax to set and test Flash variables

In previous Director releases it was necessary to use the `getVariable()` and `setVariable()` methods to access Flash variables. Director MX 2004 now allows you to access Flash variables and execute methods directly on the Director sprite.

To set a Flash variable:

- Use the following syntax:

  ```
  spriteReference.myFlashVariable = "newValue"
  ```

  For example, the following expression sets the Flash variable called myColorSwatch to red on the sprite called myFlashSprite:

  ```
  sprite("myFlashSprite").myColorSwatch = "red"
  ```

To get the value of a Flash variable:

- Use the following syntax:

  ```
  put spriteReference.myFlashVariable
  ```

  For example, the following expression gets the value of the Flash variable called myColorSwatch on the sprite called myFlashSprite:

  ```
  put sprite("myFlashSprite").myColorSwatch
  ```

To execute a Flash method:

- Use the following syntax:

  ```
  spriteReference.myFlashMethod()
  ```

  For example, the following expression calls the method setColorSwatch on the sprite called myFlashSprite:

  ```
  sprite("myFlashSprite").setColorSwatch("blue")
  ```

For more information, see “Using Flash objects in script” on page 193.

You can also use the following two sprite methods to access ActionScript variables in Flash sprites: `getVariable()` and `setVariable()`. For more information, see the Scripting Reference topics in the Director Help Panel.

- To return a string that contains the current value of a Flash sprite variable, use the following statement:

  ```
  spriteReference.getVariable("variableName", TRUE)
  ```

  The parameter `TRUE` is the default, and is therefore optional.
- To return a reference to the value of a Flash variable instead of the variable's literal value, add a value of `FALSE` to the end of the method. This lets you get or set the value of the variable simply by using the reference.

  ```
  myVariableReference = spriteReference.getVariable( "variableName", FALSE)
  ```

  Once you have created the reference to the variable, you can test it with the following statement:

  ```
  put myVariableReference
  -- value
  ```
- To set the current value of a Flash sprite variable to a specified string, use the following statement:

  ```
  spriteReference.setVariable( "variableName", "newValue" )
  ```

  Note: Be sure to pass the Flash variable's name and value as strings in both the `getVariable()` and `setVariable()` methods. Failure to do so results in script errors when the methods are executed.

Sending messages from Flash content using getURL

A Flash sprite can send messages to Director in the form of a string by using the Flash ActionScript `getUrl()` method. The string can be an event message sent either to the scripting engine of Director at the movie level (for example, a movie script), or at the sprite level received only by the behavior located on the Flash sprite sending the message. The message can also be a simple string such as "Hello Director" received by an `on getURL` handler in a movie script.

In Flash, you create a button or frame and then assign it a `getURL()` action in which you specify the message you want the Flash sprite to send to Director.

The section covers the following topics:

- "Sending simple messages and script statements" on this page.
- "Sending script statements with arguments" on page 192

Sending simple messages and script statements

You can send simple strings or script statements from Flash content to a Director movie. To pass more complicated strings, see "Sending script statements with arguments" on page 192.

To get Flash content to generate a message for Director:

1 In Flash, add the `getURL` function to your ActionScript code.

2 As the URL parameter of the `getURL` function, enter the Director script statement you want Flash to send to the movie.

To handle a message string passed by Flash content:

1 Specify the message string in the Flash content, as described above.

  For example, in Flash, you could specify the following string as the URL parameter of the `getURL` function:

```
Hello World
```

  In Flash, the ActionScript would look like this:

```
getURL("Hello World");
```

2 In Director, include an `on getURL` handler to receive and read the string passed by the Flash content.

  For example, in Director, you could enter the following handler in a movie script:

```
on getURL me, stringFromFlash
  _movie.go("stringFromFlash")
end
```

  When the `on getURL` handler receives the text string ("Hello World"), it reads the string and then jumps to the frame labeled Hello World in the Director Score.

To set up a script statement for execution at the Movie level:

1 In Flash, specify an event message by specifying the word lingo followed by a colon, the name of a handler you write in Director, and a parameter (if any) to pass with the event.

  For example, in Flash, you could specify the following statement as the URL parameter of the getURL function:

```
lingo: myDirectorScript
```

  In Flash, the ActionScript would look as follows:

```
getURL("lingo:myDirectorScript");
```

  Note: Using lingo: with getURL fully supports sending Lingo, not JavaScript, syntax. However, you can send JavaScript syntax method calls to Director since the syntax for calling a method is the same syntax as calling Lingo handlers.

2 In Director, include an event handler to execute the statement passed by the Flash content.

  For example, in Director, you could write a corresponding handler as follows:

```
on myDirectorScript
  _movie.go("Flash Message Received")
end
```

  When the Director script receives the myDirectorScript message, the movie executes the movie-level Director script handler and jumps to the frame specified in the handler.

To set up a script statement for execution at the sprite/Behavior level:

- In Flash, specify a script statement to be sent to a behavior attached to the Flash sprite from which the script message is sent.

  You specify the statement by specifying the word event, followed by a colon, followed by the script statement that you want the behavior to execute.

  For example, in Flash, you could specify the following statement as the URL parameter of the getURL function:

```
event: alertMessageReceived
```

In Flash, the ActionScript would look as follows:

```
getURL("event:alertMessageReceived");
```

When Director receives the getURL message from the Flash sprite, the Director movie immediately executes the script statement.

Sending script statements with arguments

It is possible to send script statements that are more complicated than a string or statements that call a single script handler reference. Sending handlers with arguments or sending arguments that contain double quote characters takes more effort to make sure that the script statement is properly formed.

Note: To send simple strings and statements, see "Sending simple messages and script statements" on page 190.

Suppose you have a Director handler in a movie script that adds two numbers and presents an alert with the sum, as follows:

```
on addTwoNumbers number1, number2
  myResult = number1 + number2
  _player.alert(string(myResult))
end
```

To pass arguments to this script (`number1` and `number2`), you would include the following statement in your Flash ActionScript:

```
getURL("lingo:addTwoNumbers 100, 200");
```

If the arguments are strings, you need to escape the double quote characters using a backslash character on each double quote character that you want to be passed through to Director. For example, if you want to send a script statement from the Flash sprite which tells Director to launch a browser and open macromedia.com, you would do the following:

```
getURL("lingo:gotonetpage(\"http://www.macromedia.com\")");
```

It may be easier to fabricate the script statement in a temporary variable inside the Flash method and then use getURL on that variable as follows:

```
var theString = getURL("lingo:gotonetpage(\"http://www.macromedia.com\")");
getURL(theString);
```

Additionally, you may want to send the value of a Flash variable as part of the script statement. For example, if `myFlashVar` in Flash represents a Macromedia product such as Director (`myFlashVar="director"` in ActionScript), you could form a script statement as follows in your Flash script:

```
var theString = getUrl("lingo:gotonetpage(\"http://www.macromedia.com/
  software/" + myFlashVar + "\")");
getURL(theString);
```

The resulting URL would be "http://www.macromedia.com/software/director" and would be passed to the browser through Director's `goToNetPage` command.

Sending XML Data from Flash to Director

You can send XML data from a Flash sprite or object to a script.

To send XML data from a Flash sprite or a global Flash object to a script:

1 On the Flash side, use the `XMLobject.send` ActionScript method.

The `XMLobject.send` method has a URL and a `targetWindow` parameter.

2 On the Director side, include an `on sendXML` handler in your script to handle rhe XML data.

This handler takes the same parameters as the `XMLobject.send` method, as follows:

```
on sendXML me, URL, targetWindow, xmlData
```

The `xmlData` parameter is the XML data contained in the original XMLObject. You can then add script to the handler to process the XML data. A common action is to send the XML data to the URL and await the response of the server located at the URL, as in this example:

```
on sendXML me, URL, targetWindow, xmlData
  gotoNetPage(URL, targetWindow)
  postNetText(URL, xmlData)
end
```

Using Flash objects in script

With Director, you can create Flash ActionScript objects and access all of their properties and methods. You can create a wide variety of Flash objects, including arrays, dates, Booleans, XML objects, and net connection objects for use with Macromedia Flash Communication Server MX. If you have authored Flash content that contains ActionScript classes that generate custom objects, you can access those objects in script as well. You can also create references to existing ActionScript objects with the `getVariable()` method. For more information about accessing these objects, see “Using Lingo or JavaScript syntax to set and test Flash variables” on page 189.

When creating Flash objects, you can choose to create an object within a Flash sprite, or you can create a global Flash object.

- To create a Flash object within a Flash sprite, you must have Flash content in the cast and a Flash sprite on the Stage. Do not use a Flash cast member created with the method `new(#Flash)`, as this creates a Flash cast member shell, which contains no actual internal Flash data. When you create a Flash object within a sprite, the object uses the same instance of the Flash Player that Director loads when the Flash cast member appears on the Stage.

  In your script, use the `newObject()` method with a sprite reference. If the object you want to create requires parameters, include them with the `newObject()` method:

  ```
  myNewFlashObject = sprite(1).newObject("Array", "apple", "orange", "banana")
  ```

  In this example, the specified sprite, `sprite(1)`, is the Flash sprite. The object is an array. (In Lingo, arrays are called lists.) The array contains three strings: `"apple"`, `"orange"`, and `"banana"`.

- To create a global Flash object, use the `newObject()` method without a sprite reference. When you do this, Director loads a special instance of the Flash Player into memory. This way, you can use Flash objects without the need for a Flash cast member or sprite.

  In your script, use the `newObject()` method without a sprite reference:

  ```
  myNewFlashObject = newObject("Array", "apple", "orange", "banana")
  ```

The Flash asset commonPlayer feature gives you the ability to load multiple Flash sprites into one instance of the Flash Player; this provides better Flash playback performance in Director objects that have a great deal of Flash assets.

Note: If you do not import any Flash cast members, you must manually add the Flash Asset Xtra to your movie's Xtra list for global Flash objects to work correctly in Shockwave and projectors. For more information about the movie Xtra list, see "Managing Xtra extensions for distributed movies" on page 450.

The `newObject()` method creates the object you specify and a reference to it. In the preceding examples, the object is an array and the reference is stored in the variable named `myNewFlashObject`. The first parameter included with the `newObject()` method is the type of object to create. The subsequent parameters are the values to put into the array. In this case, the values are a list of names of fruit.

- To access a property of the object, such as the array's length, you only need to refer to the property as a property of the object reference you created:

```
put myNewFlashObject.length
-- 3
```

- To access a part of the object, such as the value of the third item in the array, use the following syntax:

```
put myNewFlashObject[2]
-- "banana"
```

Note: The items in an ActionScript array are numbered beginning with zero, while the items in a script list are numbered beginning with one. Be sure to use the correct number when referring to items in arrays or lists.

- To access a method of the object, use the same syntax and specify the method name after the object reference:

```
myNewFlashObject.sort()
```

For more information about the types of objects supported by Flash and the methods and properties used to control them, see the Flash documentation.

Setting callbacks for Flash objects

Some kinds of Flash objects generate events that must be routed to an appropriate script handler. For example, a Flash Communication Server connection object generates an event each time an incoming message is received from the server. You can set script to route events like this to a specific handler in a specific script object by using the `setCallback()` method.

- To set up a callback for an event from a Flash object, use script similar to this:

```
myConnection = sprite(1).newObject("NetConnection")
sprite(1).setCallback(myConnection, "onStatus", #dirOnStatusHandler, me)
```

In the first line of this example, a new `netConnection` object is created along with a reference to it named `myConnection`. The second line calls the handler named `dirOnStatusHandler` whenever the `myConnection` object generates an `onStatus` event. The argument `me` indicates that the handler is located in the same script object that the `setCallback()` method is being issued in. In this case, that script object is attached to sprite 1.

The callback handler could look like the following:

```
on dirOnStatusHandler (me, aInfoObject)
  if (aInfoObject[#level] = "error") then
    member("chat input").text = "Error sending last message."
  end if
end
```

Using the Flash local connection object

Flash includes an object type called local connection. The Flash local connection object can be very useful for allowing separate movies on the same computer to connect to and communicate with each other. Because the local connection object is a Flash object supported in Director, it can allow communication between separate Flash content, Director movies, or combinations of the two. You can use the messaging capability of the local connection object for simple tasks such as exchanging sprite property data, or for more complex ones such as exchanging chat messages when used in conjunction with the Flash Communication Server. For more information, see "Using Flash Communication Server MX 2004" on page 198.

To use the local connection object, you can either create a global local connection object or associate the object with a Flash sprite in the Score. Once you have created the object, you can control it entirely through script. The following are examples of a script that is attached to a Flash sprite in channel 1 of the Score. The script contains a `beginSprite` handler and other handlers that manage the local connection object.

Initialize properties

The first thing to do is declare some properties that you will use throughout the local connection script to store references to the local connection object and 2 connections, one outgoing and one incoming.

- The property `pCon_name` stores the name of an outgoing connection
- The property `pOtherCon_name` stores the name of an incoming connection
- The property `pLocalCon` stores a reference to the local connection object

The beginning of the script might look like this, including the start of the `beginSprite` handler:

```
property pCon_name
property pOtherCon_name
property pLocalCon

on beginSprite (me)
  pCon_name = "userA"
  pOtherCon_name = "userB"
```

Creating the local connection object

The next step is to create a new local connection object. Once the object is created, you can use the `setCallback()` method to set up handlers to respond to the events that the local connection object generates. You can then also use the methods of the local connection object to connect to other movies and send messages.

- To create the new local connection object, use the `newObject()` method:

  ```
  pLocalConn = sprite(1).newObject("LocalConnection")
  ```

  This script assigns the property `pLocalConn` to be a reference to the newly created object.

Setting up callbacks

The next step is to set up callback handlers with the `setCallback()` method. You should set up a callback for each event you expect the object to generate. Local connection objects generate `onStatus` and `allowDomain` events, as well as an event for each incoming message received. These incoming message events are named by the string that is passed as the subject, or first parameter, of the message.

To set up each callback, use the `setCallback()` method and include the name of the local connection object, the name of the event to respond to, the script handler name, and the script object that contains the handler as arguments with the method.

To set up localConnection callbacks:

1 Set up the `onStatus` callback. An `onStatus` event is generated each time a message is sent by the local connection object and contains information about the success or failure of the send operation.

```
sprite(name).setCallback(pLocalConn, "onStatus", #myOnStatus, me)
```

This script statement sets a callback for the event named `onStatus` generated by the object `pLocalConn`. The script handler name is `myOnStatus` and the script object that contains it is the same script object that contains this `setCallback()` method, referred to as `me`. The quotes around the event name and the pound sign (#) preceding the handler name. The pound sign converts the handler name to a symbol.

The actual callback handlers, such as the `myOnStatus` handler, are illustrated later in this section.

2 Set up the `allowDomain` callback. An `allowDomain` event is generated each time the local connection object receives an incoming message.

```
sprite(name).setCallback(pLocalConn, "allowDomain", #myAllowDomain, me)
```

In this case, the event name is `AllowDomain` and the handler name is `myAllowDomain`.

If the `myAllowDomain` handler determines that the message is coming from an allowed domain, an event named with the message subject is generated. When a movie is running from a user's computer and not in a browser, the domain is `localHost`. When the movie is running in a browser, the domain is determined by the server that hosts the movie, such as macromedia.com.

3 Set up the callback for these user-defined messages. Decide what string you want to use as your message subjects, so that you know what the event name will be. In the following example, the user-defined message subject and event name are `incomingMessage`:

```
sprite(name).setCallback(pLocalConn, "incomingMessage", #myIncomingMessage,
  me)
```

Writing callback handlers

Now the callbacks are set up. In order for them to work, however, you must write the callback handlers themselves. While the `setCallback()` methods are all inside a `beginSprite` handler in this example, the callback handlers must be outside the `beginSprite` handler because they are handlers by themselves. In this example, the callback handlers are located in the same script attached to the Flash sprite, just after the `beginSprite` handler.

The onStatus event is generated each time the local connection object sends an outgoing message. The myOnStatus handler might look like this:

```
on myOnStatus (me, aInfoObject)
  if (aInfoObject[#level] = "error") then
    member("chat input").text = "Error sending last message."
  else
    member("chat output").text = member("chat output").text & RETURN & \
      member("chat input").text
  end if
end myOnStatus
```

Two arguments are passed with the onStatus event by the local connection object. The me argument tells the script that the event was generated on the same script object that contains the myOnStatus handler. The aInfoObject argument contains a Flash array object containing information about the status of the message sending operation. The handler checks to see if the aInfoObject argument contains an error. In this example, if there is an error, an error message is displayed in a chat input field. If no error occurs, a text output field is updated with the contents of the chat input field. For more information about the Flash infoObject, see the Flash Communication Server MX documentation.

The allowDomain event is generated each time the local connection object receives an incoming message. This provides an opportunity for the myAllowDomain callback handler to determine whether the message is coming from a trusted domain. The myAllowDomain callback handler must return TRUE in order for the incoming message to be processed.

The myAllowDomain handler might look like this:

```
on myAllowDomain (me, aSendingDomain)
  if aSendingDomain = "myDomain.com" then
    return TRUE
  else
    member("chat output").text = & RETURN & "Message received from \
    unapproved domain."
    return FALSE
  end if
end myAllowDomain
```

The handler checks that the domain is the one that is expected, and returns TRUE if it is. If the message comes from another domain, the return value is FALSE.

Once the allowDomain callback has returned TRUE, the local connection object forwards the incoming message to the callback handler set up for the event. In this example, the subject of the message is incomingMessage and the callback handler is myIncomingMessage.

The myIncomingMessage handler might look like the following:

```
on myIncomingMessage (me, aObject, aMessage)
  member("chat output").text = & RETURN & aMessage
end myIncomingMessage
```

This handler simply appends the incoming message aMessage to the end of a chat output field.

Sending messages and closing the connection

In order to complete your script, you must finish the `beginSprite` handler and write handlers for sending messages from the object and closing the connection when you are finished using it.

- To finish the `beginSprite` handler that already contains the `newObject()` method and all the `setCallback()` methods, you must add a `connect()` method. This is actually a Flash ActionScript method that you send to the local connection object you created.

  ```
  pLocalConn.connect(pCon_name)
  ```

  The argument `pCon_name` gives the actual connection a name, “userA”, that was declared at the beginning of the script.
- To send messages, write a handler that uses the `send()` method of the local connection object:

  ```
  on sendMessage (me, aMessage)
    tMessage = pCon_name && ":" && aMessage
    pLocalConn.send(pOtherCon_name, "incomingMessage", tMessage)
  end sendMessage
  ```

  The `send()` method requires three arguments: the recipient of the message, the event to be triggered when the message is received, and the message itself.
- To close the connection, write a handler that uses the `close()` method of the local connection object:

  ```
  on closeConnection (me)
    pLocalConn.close()
  end closeConnection
  ```

  You can call this handler from any other handler by using the statement:

  ```
  sendSprite (1, #closeConnection)
  ```

  You might also use the `close()` method in an `endSprite` handler:

  ```
  on endSprite (me)
    pLocalConn.close()
  end endSprite
  ```

Now that the local connection object is set up and it has handlers for callbacks, sending messages, and closing the connection, it is ready to be used by the movie.

Using Flash Communication Server MX 2004

Macromedia Flash Communication Server MX allows Flash content on separate computers to share information, including sound, video, text and other data in real time. You can use Flash Communication Server MX in Director by using Flash cast members that are designed to work with the server, or by creating `NetConnection` and `NetStream` objects in script that you use to communicate with the server.

As with any Flash ActionScript object you create by using script, you use the exact same methods and properties to manipulate the object as you would in ActionScript. For a detailed example, see the previous section. The Macromedia Director MX 2004 installation CD for Windows includes Flash Communication Server MX Personal Edition, the Flash Communication Server MX authoring components for Flash, and documentation. The Macromedia Director MX 2004 installation CD for Macintosh includes the Flash Communication Server MX authoring components for Flash and documentation.

The steps required to communicate with Flash Communication Server MX are identical to those you would use in ActionScript.

To create a NetConnection object:

- Use the `newObject()` method.

```
myNetConObject = sprite(1).newObject("NetConnection")
```

To create a NetStream object:

- Use the `newObject()` method and include the `NetConnection` object as a parameter:

```
myStream = sprite(1).newObject("NetStream", myNetConObject)
```

The `NetStream` object can send text messages without the need for a Flash sprite on the Stage.

To create a global NetConnection object that does not require a sprite reference:

- Use the `newObject()` method and omit the sprite reference.

```
myNetConObject = newObject("NetConnection")
```

To create a global NetStream object that does not require a sprite reference:

- Use the `newObject()` method and include the `NetConnection` object as a parameter, and omit the sprite reference:

```
myStream = newObject("NetStream", myNetConObject)
```

To send text messages with the NetStream object:

- Use the send method.

```
myStream.send(handlerName {,p1, ...,pN})
```

To send audio or video, you need to associate a camera and microphone with the `NetStream` object.

To associate a video camera with the NetStream object:

- Use the ActionScript attachVideo method.

```
myStream.attachVideo(source)
```

To associate a microphone with the NetStream object:

- Use the ActionScript attachAudio method.

```
myStream.attachAudio(source)
```

To publish a video, audio, or other data stream with the NetStream object:

- Use the ActionScript publish method.

```
mystream.publish(whatToPublish)
```

To play a non-video data stream from the server with the NetStream object:

- Use the ActionScript play method

```
mystream.play(whatToPlay)
```

To receive a video stream from the server, the stream must be attached to a video clip instance in a Flash sprite. Sample Flash content containing a video clip object is included in the Macromedia/ Support/Flash/ folder on the Director installation CD.

To create a script reference to the video clip object in the Flash sprite:

- Use the `getVariable()` method.

```
videoRef = sprite(1).getVariable(nameOfFlashVideoClip, FALSE)
```

To play a video stream through the video clip object:

- Use the `attachVideo()` method with the reference to the video clip.

  ```
  videoRef.attachVideo(source)
  ```

For detailed examples of how to use Flash Communication Server in Director, see the Director Support Center (www.macromedia.com/go/director_support).

Using the Flash Settings panel

When you use a Flash cast member in your Director movie, you may want to display the Flash Settings panel. The Flash Settings panel lets users choose privacy, storage, camera, and microphone settings that affect playback of Flash content that communicate with a Flash Communication Server. For more information about the options in the Settings panel, see the Macromedia Flash Player Help section of the Macromedia website (www.macromedia.com/go/flashplayer_help_en).

Opening the Settings panel

You must have a Flash sprite on the Stage to display the Settings panel. The sprite must be at least as large as the Settings panel (214x137 pixels). The Flash sprite's Direct-to-Stage property must be set to `TRUE`. You can set this property in the Flash tab of the Property inspector. If the Flash sprite is not set to display Direct-to-Stage or is too small, the Settings panel does not appear, but no error occurs.

You display the Settings panel in a Director movie by using the `settingsPanel()` method. Once the panel is displayed, the user can choose the desired settings and then close the panel.

To display the Flash Settings panel, use the following script:

```
sprite(flashSpriteReference).settingsPanel(integerWhichTabToDisplay)
```

For more information about the parameters for this method, see the Scripting Reference topics in the Director Help Panel.

You might decide to invoke the Settings panel by allowing the user to click a button on the Stage that opens the panel. In this case, you would use the `settingsPanel()` method in a `mouseUp` or `mouseDown` handler attached to the button sprite. You could also use the `settingsPanel()` method at any other time to open the panel, depending on how you want to make the panel available.

Emulating the Flash Player context menu in Shockwave

You might decide that you want to allow users to access the Settings panel by right-clicking (Windows) or Control-clicking (Macintosh) on your Flash sprite when your movie is playing in a browser. To do this, you must first disable the context menu that is built into the Macromedia Shockwave Player. After the Shockwave context menu is disabled, you can trigger the Flash Settings panel by right-clicking or Control-clicking.

To disable the Shockwave context menu:

1 Select File > Publish Settings.
2 Click the Shockwave Save Tab.
3 Deselect the Display Context Menu in Shockwave option. This sets a parameter named `swContextMenu` in the Shockwave `<Object>` and `<Embed>` tags to `FALSE`.

4 Click OK.

5 Save your movie. Your Publish Settings are saved with the movie.

Once you have disabled the Shockwave context menu, you can attach a `mouseUp` handler to your Flash sprite that tests for the `rightMouseDown` property or the `controlDown` property. In Windows, you can choose just to write an `on rightMouseUp` handler.

The handler might look like the following:

```
on mouseUp
    if the rightMouseDown or the controlDown then
        sprite(1).settingsPanel(0)
    end if
end
```

or

```
on rightMouseUp
    sprite(1).settingsPanel(0)
end
```

You can also choose to use ActionScript in your Flash content to enable the Settings panel. For more information, see the Macromedia Flash MX 2004 documentation.

Playback performance tips for Flash content

Performance of Flash content can vary greatly, depending on the options in effect and the playback environment. The following are tips for getting optimal playback performance from Flash:

- If adequate for your needs, use the Low quality setting rather than High. Using Low turns off anti-aliasing, which speeds up Flash animation rendering. A handy technique is to switch the quality of the sprite to Low while displaying a fast-moving animation sequence (such as a spinning logo), and then switch the quality back to High on the fly as the animation slows down or comes to a stop. This way, performance can be improved during the part of the sequence where it would be more difficult to perceive the improved quality anyway, without sacrificing quality in the end result.
- Experiment with different system color depths to see what provides the best performance. For example, some graphics, such as gradients, display faster at 16 bits.
- Use Copy ink if possible. Transparency, using Background Transparent ink, requires much more processing time. If your Flash sprite is in the background (no other Director sprites are behind it), use Copy instead of Background Transparent, and author your Flash content in such a way that its background color is the same as the background color you chose for your Director Stage.
- Use Direct to Stage if possible. Layering and transparency are not supported in this mode; however, if you just want to play Flash content within a box with the best performance possible, this may be the way to go.
- Make sure that the Director movie tempo is set high enough. Unless you're using Direct-to-Stage, your Flash content does not play faster than the Director movie frame rate, regardless of the `frameRate` or `fixedRate` setting. For smoothest playback, set the Director frame rate to at least 30 frames per second (fps).

- Use Lock-Step or Fixed playback mode to adjust the Flash content frame rate. Lock-Step gives the best performance, because playback of the Flash content is synchronized to the Director movie frame for frame.
- Set the `static` property of the sprite to `TRUE` if your sprite contains no animation (such as a static block of text) and doesn't overlap other moving Director sprites. This keeps Director from redrawing the sprite every frame unless it moves or changes size.
- When modifying Flash properties using script, set the properties for the sprite rather than for the cast member. Setting the properties for the cast member modifies values at the cast member level and broadcasts the change to all sprites on the Stage. This overhead can affect performance. If you have only a single sprite for the cast member, modify the sprite property directly.
- Limit the amount of script that executes while the Flash content plays. Avoid tight repeat loops between frames. The usual Director performance optimizations apply when using Flash movies.
- If you import compressed SWF files, be aware that Director will use memory for both the compressed and decompressed versions of the file until the file has been completely decompressed into memory.

Using Director movies within Director movies

You can import a Director movie into another movie as an internal or linked cast member, with the Import command. As with other media types, you can link to an external movie file or import the file so that it becomes internal media. The way you choose to import a movie affects its properties:

- For linked movies, cast member scripts and behaviors (sprite scripts) work as before; select Enable Scripts in the Linked Movie tab of the Property inspector. Frame and movie scripts do not work. As with other types of linked media, the external movie file must be present on the system when the host movie plays.
- For movies imported as internal media, the movie appears as a film loop, and interactivity does not work. Use this mainly for animations.

For both types of imported movies, the host movie controls the tempo settings, palette settings, and transitions. Settings for these methods in the imported movie are ignored.

Once it is imported, the movie appears as a cast member in the Cast window. The cast members of a movie imported as internal media also appear in the Cast window. You can animate the cast member just as you would any graphic cast member, film loop, or digital video.

To import a Director movie:

1. Select File > Import.
2. From the Files of Type pop-up menu, select Director Movie.
3. Select a Director movie.
4. To determine whether the movie is imported into the current movie file or linked externally, select a Media option.

   Standard Import imports all the movie's cast members into the current cast and creates a film loop that contains the Score data. Scripts in the imported movie will not work.

   Link to External File creates a cast member that references the external movie file. A linked movie appears as a single cast member.
5. Click Import.

To place a Director movie cast member in the current movie:

1 Do one of the following:
   - For an internal movie, drag the film loop cast member to the Stage or Score.
   - For a linked external movie, drag the movie cast member to the Stage or Score.

2 Extend the sprite through all the frames in which you want it to appear.

3 To change any of the movie's properties, use the Movie tab of the Property inspector.

   See the next section.

Setting linked Director movie properties

To determine whether a linked Director movie is cropped or scaled to fit within a sprite's bounding rectangle, you use the Property inspector. You can also use the Property inspector to enable the movie's scripts, mute its sounds, and specify whether it loops.

To set linked movie properties:

1 Select a linked movie cast member.

2 To display the Property inspector, select Modify > Cast Member > Properties, or select Window > Property Inspector.

3 If necessary, display the Member tab in Graphical mode.

   The following noneditable settings are displayed:
   - The cast member size in kilobytes
   - The cast member creation and edit dates
   - The name of the last person who modified the cast member

4 To view or edit the cast member name, use the Name field.

5 To add comments about the cast member, use the Comments field.

6 To specify how Director removes the cast member from memory if memory is low, select one of the following options from the Unload pop-up menu:

   3–Normal sets the selected cast members to be removed from memory after any priority 2 cast members have been removed.

   2–Next sets the selected cast members to be among the first removed from memory.

   1–Last sets the selected cast members to be the last removed from memory.

   0–Never sets the selected cast members to be retained in memory; these cast members are never unloaded.

7 Click the Linked Movie tab in Graphical mode.

8 To determine how the linked movie appears within the sprite bounding rectangle, select a Framing option:

   Crop displays the movie image at its default size. Any portions that extend beyond the sprite's rectangle are not visible.

   Center is available only if Crop is selected. It keeps the movie centered within the sprite's bounding rectangle if you resize the sprite.

   Scale resizes the movie to fit inside the bounding rectangle.

9 To determine how the linked movie plays back, set the following options

Play Sound enables the sound portion of the linked movie. Turn this option off to mute sounds.

Loop replays the linked movie continuously from the beginning to the end and back to the beginning.

Enable Scripts makes all the scripts in the linked movie work the same way they do when the movie plays by itself.

If you import a Director movie internally, it is imported as a film loop. In this case, the Linked Movie tab in the Property inspector is replaced by the Film Loop tab, and the Enable Scripts option is not available.

Using ActiveX controls

In Director, movies that you intend to publish only as projectors for Windows, you can embed ActiveX (formerly known as OLE/OCX controls) controls that let you take advantage of the technology and adapt ActiveX controls to make them function as sprites in Director. You can use ActiveX controls to manage application resources for the hosted ActiveX control—for instance, to manage properties, events, and windows and filing properties. You can also manage resources used by the ActiveX control within the Director movie. ActiveX controls are not allowed in Shockwave.

The range of uses for ActiveX in Director is as limitless as the variety of ActiveX controls available. Using the Microsoft Web Browser control (installed with Microsoft Internet Explorer 3.0 or later), you can browse the Internet from within a multimedia production; using the FarPoint Spreadsheet control, you can create and access spreadsheets; using the InterVista VRML control, you can explore virtual worlds; using the MicroHelp extensive library of Windows widget controls, you can build and simulate complete Windows applications.

Note: Not all ActiveX controls expose their methods and properties in all hosts. Test the controls you want to use to see how they work in Director. Because ActiveX controls are non-Macromedia software, Macromedia Technical Support does not support them.

Inserting an ActiveX control

You can place ActiveX controls in a Director movie and have them function as sprites. Note that this procedure is designed only for Director for Windows.

To insert an ActiveX control on the Stage:

1 Make sure that the ActiveX control you want to use in Director is installed on your system.

Most controls have their own installation utilities provided by the manufacturer of the control.

2 Select Insert > Control > ActiveX.

3 In the dialog box that appears, select the desired ActiveX control and then click OK. The ActiveX Control Properties dialog box appears.

(If the desired ActiveX control does not appear in the list, it may not have been installed properly by the system. You can attempt to verify this by viewing the list of ActiveX controls in another application such as Visual Basic.)

The ActiveX Control Properties dialog box lets you edit each ActiveX control and view information regarding each method the control supports and each event the control can generate.

4 Set the values for each property in the ActiveX control and then click OK. The ActiveX control now appears in the cast.

5 Drag the ActiveX control from the cast to the Stage.

   Once the ActiveX control appears on the Stage, it can be repositioned and resized just like any other sprite Xtra. When you pause the movie, the ActiveX control stays in authoring mode and does not react to mouse or keyboard events. When you play the movie, the control responds to user input.

Setting ActiveX control properties

An ActiveX control describes its information using properties—named characteristics or values such as color, text, font, and so on. Properties can include not only visual aspects but also behavioral ones. For example, a button might have a property that indicates whether the button is momentary or push-on/push-off. An ActiveX control's properties define its state—some or all of which properties may persist. Although the control can change its own properties, it is also possible that the container holding the control might change a property, in response to which the control would change its state, user interface, and so on.

When an ActiveX control is inserted into a Director movie, the properties that the control exposes can be viewed and edited by clicking the Properties tab of the Control Properties dialog box for the ActiveX Xtra. Each property exported by the ActiveX control is identified along with the current value of the property. The user edits a property value by simply clicking over the existing value with the mouse. For most properties, such as numeric or string values, the new value can be directly entered into the list using the keyboard.

In Director, all properties that an ActiveX control exports are properties of the corresponding sprite. This is the generic syntax for setting an ActiveX control property:

```
sprite(X).propertyName = value
```

The generic syntax for getting an ActiveX control property is as follows:

```
value = sprite(X).propertyName
```

As an example, if the Microsoft Access Calendar control is inserted into a Director movie as the second sprite on the score, the following script sets the Year property of the Calendar control to a specific year:

```
sprite(2).year = 1995
```

To get the Year property from the same Calendar control and place it into a variable named `CalendarYear`, you can use the following script:

```
CalendarYear = sprite(2).year
```

Some ActiveX control properties are read-only, and trying to set a property for such a control causes an error in Director. For more information, see the documentation for the ActiveX control you are using.

Using ActiveX control methods

An ActiveX control describes its functionality using methods. Methods are simply functions implemented in the control that Director can call to perform some action. For example, an edit or other text-oriented control supports methods that let Director retrieve or modify the current text, perhaps performing such operations as copy and paste.

When you insert an ActiveX control in a Director movie, you can view the methods exposed by the control by clicking the Methods tab of the Control Properties dialog box for the ActiveX control. The dialog box displays each method supported by the ActiveX control and a description of the parameters for each method.

In Director, all methods that an ActiveX control supports are methods for the corresponding sprite. The generic syntax for calling an ActiveX control method is as follows:

```
ReturnValue = sprite(N).MethodName(param1, param2, ...)
```

As an example, if the Microsoft Access Calendar control is inserted into a Director movie as the second sprite on the Score, the following script would increment the year displayed within the Calendar control:

```
sprite(2).NextYear()
```

For the same Calendar control, the following script would decrement the year displayed by the Calendar control:

```
sprite(2).PrevYear()
```

Parameters passed to the ActiveX control are automatically converted from their Director data types to equivalent ActiveX data types. Likewise, the return value is automatically converted from an ActiveX data type to an equivalent Director data type.

Using ActiveX control events

Each ActiveX control typically generates a variety of events. For example, a button ActiveX control may generate a `click` event when the button is pressed, and a calendar ActiveX control may generate a `dateChanged` event when the date within the calendar is changed. Director converts any event generated by the ActiveX control to a sprite event that it can handle. A list of the control's events appears in the Events tab of the ActiveX Control Properties window.

To respond to an event generated by the ActiveX control, you must write an event handler to capture the event. You can place these event handlers in movie scripts, sprite behaviors, scripts assigned to cast members, or frame behaviors. However, you normally place the handler in the behavior attached to the sprite for the ActiveX control.

As an example, if the Microsoft Access Calendar control is inserted into a Director movie as a sprite on the score, the following script would capture the `click` event from the Calendar control:

```
on click
  -- Do something interesting here.
  beep 2
end
```

A sprite behavior is a good location for this handler.

Using Flash components

Macromedia Flash MX 2004 components are bundled movie clips with ActionScript programming interfaces. Director comes with a set of user interface components, including list boxes, radio buttons, check boxes, a scroll pane, and more.

You can also add components to the existing set by creating your own in Flash MX 2004 and dragging and dropping them into the Director Components folder. The Components folder is located in the directory where you installed Director; the exact location within that directory varies based on your operating system.

If you add new components, you must restart Director to use them. For more information about the components that are bundled with Director, other component types, creating your own components, and adding new components, see the Flash Documentation website at www.macromedia.com/go/fl_documentation.

Selecting Flash components

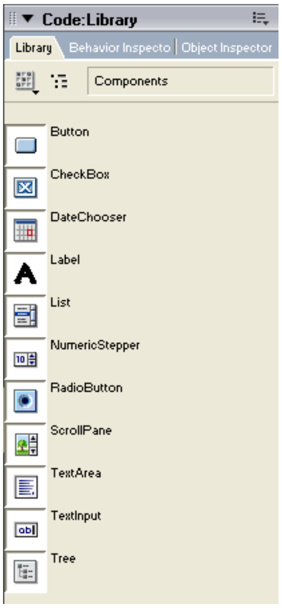
You can select Flash components to use in your Director movie using the Flash component section of the Library palette. You can also access components on the Tools palette. For more information, see "Selecting components using the Tool palette" on page 208.

To select components from the Library palette:

1 Select Window > Library palette.

2 Click the Components icon.

A list of Flash components appears.



3 Drag and drop the component you want to a Cast window slot, the Score, or directly to the Stage.

Director MX 2004 includes the following Flash components:

Button component creates a button that can be customized to include an icon.

CheckBox component lets you insert a graphic component that can represent an on or off state.

DateChooser component lets you insert a graphic calendar of a month. The calendar shows the current date, and allows the user to navigate to any new day, month, or year.

Label component lets you insert a single-line, static text item (not editable).

List component lets you insert a scrolling list that supports single or multiple selections.

NumericStepper component inserts a graphic component that lets a user select a number from an ordered set.

RadioButton component inserts a graphic radio button that represents a single choice within a set of mutually exclusive choices.

ScrollPane component lets you display a linked JPEG or SWF file in a scrollable area.

TextArea component is an editable, multiline text region. A TextArea component can be enabled or disabled in an application. In the disabled state, it doesn't receive mouse or keyboard input.

TextInput component is an editable single-line text region. In the disabled state, it doesn't receive mouse or keyboard input.

Tree component lets you create a graphical representation for organizing and manipulating hierarchical data.

Note: There may be Flash components not described here that have been added to Director after this document was created.

Selecting components using the Tool palette

You can also select some Flash components directly from the Tool palette and drag them to the Stage, Score, or Cast window.

To select a Flash component from the Tool palette:

1 If the Tool palette is not active, select Window > Tool Palette.

The Tool palette appears.

2 Click the arrow on the Tool palette pop-up menu and select Flashcomponent.

3 The Tool palette changes to include a selection of common Flash components.

4 Select the component you want and move the cursor to the Stage.

5 Click on the Stage where you want the component to appear.

You can click the component on the stage again to resize it. You can also set size and position properties using the Sprite tab in the Property inspector. For more information, see "Displaying and editing sprite properties in the Property inspector" on page 59.

Setting Flash component parameters

Each component has unique parameters that you can set to change its appearance and behavior. A parameter is a property or method that appears in the Property inspector. The most commonly used properties and methods appear as authoring parameters; others must be set using scripting only. After playback has begun, you can get and set parameters on a sprite instance of a Flash component in much the same way that you customize Flash content. For more information, see "Using Lingo or JavaScript syntax to set and test Flash variables" on page 189.

Handling Flash component events

All Flash components have events that are broadcast when the user interacts with a component or when something significant happens to the component; for example, when the user presses a key or clicks the mouse. To handle an event, Flash uses ActionScript code that executes when the event is triggered. You can respond to these events in Director MX 2004 by writing an event handler in Lingo or JavaScript. Below, you'll find an example for changing one component. For complete information about individual components and their properties and events, see Using Components in the Flash documentation.

To set the Flash Button component events:

1. In the Stage, Score, or Cast window, select a Button Flash component.
2. Open the Property inspector and select the Flash Component tab.

   A list of events appears that may be generated by the component. Set the event to true if you want to respond to the event in Director.

The following example shows how to add a behavior to an instance of the Flash Button component that prints a message to the Message Window when the button is clicked.

1. Place a Button on the stage and select it.
2. Click the Components tab in the Property inspector.
3. Set the click event to true.
4. In the Behavior inspector, add a new behavior called myClickHandler that contains the following code:

```
on click
  put "Received a click event!"
end
```

Button component

The Button component is a resizable rectangular user interface button. You can add a custom icon to a button. You can also change the behavior of a button from push to toggle. A toggle button stays pressed when clicked and returns to its up state when clicked again.

A button can be enabled or disabled in an application. In the disabled state, a button doesn't receive mouse or keyboard input. An enabled button receives focus if you click it or tab to it.

Button parameters

The following are authoring parameters that you can set for each Button component instance in the Property inspector or in the Component Inspector panel:

label sets the value of the text on the button; the default value is Button.

icon adds a custom icon to the button. The value is the linkage identifier of a movie clip or graphic symbol in the library; there is no default value.

toggle turns the button into a toggle switch. If true, the button remains in the down state when pressed and returns to the up state when pressed again. If false, the button behaves like a normal push button; the default value is false.

selected if the toggle parameter is true, this parameter specifies whether the button is pressed (true) or released (false). The default value is false.

labelPlacement orients the label text on the button in relation to the icon. This parameter can be one of four values: left, right, top, or bottom; the default value is right. For more information, see `Button.labelPlacement`.

You can write scripts to control these and additional options for Button components using its properties, methods, and events. For more information, see "Button class" in the Flash documentation.

Using styles with the Button component

You can set style properties to change the appearance of a button instance. If the name of a style property ends in "Color", it is a color style property and behaves differently than non-color style properties.

A Button component supports the following Halo styles:

Style	Description
`themeColor`	The background of a component. This is the only color style that doesn't inherit its value. Possible values are "haloGreen", "haloBlue", and "haloOrange".
`color`	The text of a component label.
`disabledColor`	The disabled color for text.
`fontFamily`	The font name for text.
`fontSize`	The point size for the font.
`fontStyle`	The font style: either "normal", or "italic".
`fontWeight`	The font weight: either "normal", or "bold".

Property summary for the Button class

Method	Description
`Button.label`	Specifies the text that appears within a button.
`Button.labelPlacement`	Specifies the orientation of the label text in relation to an icon.
`Button.selected`	When the `toggle` property is `true`, specifies whether the button is pressed (`true`) or not (`false`).
`Button.toggle`	Indicates whether the button behaves as a toggle switch.

Event summary for the Button class

Method	Description
`Button.click`	Broadcast when the mouse is pressed over a button instance or when the Spacebar is pressed.

CheckBox component

A check box is a fundamental part of any form or web application. You can use check boxes wherever you need to gather a set of `true` or `false` values that aren't mutually exclusive. For example, a form collecting personal information about a customer could have a list of hobbies for the customer to select; each hobby would have a check box beside it.

CheckBox parameters

The following are authoring parameters that you can set for each CheckBox component instance in the Property inspector or in the Component Inspector panel:

label sets the value of the text on the check box; the default value is defaultValue.

selected sets the initial value of the check box to checked (true) or unchecked (false).

labelPlacement orients the label text on the check box. This parameter can be one of four values: left, right, top, or bottom; the default value is right. For more information, see `CheckBox.labelPlacement`.

You can write scripts to control these and additional options for CheckBox components using its properties, methods, and events. For more information, see "CheckBox class" in the Flash documentation.

Using styles with the CheckBox component

You can set style properties to change the appearance of a CheckBox instance. If the name of a style property ends in "Color", it is a color style property and behaves differently than non-color style properties.

A CheckBox component supports the following Halo styles:

Style	Description
`themeColor`	The background of a component. This is the only color style that doesn't inherit its value. Possible values are "haloGreen", "haloBlue", and "haloOrange".
`color`	The text of a component label.
`disabledColor`	The disabled color for text.
`fontFamily`	The font name for text.
`fontSize`	The point size for the font.
`fontStyle`	The font style: either "normal", or "italic".
`fontWeight`	The font weight: either "normal", or "bold".
`textDecoration`	The text decoration: either "none", or "underline".

Property summary for the CheckBox class

Property	Description
`CheckBox.label`	Specifies the text that appears next to a check box.
`CheckBox.labelPlacement`	Specifies the orientation of the label text in relation to a check box.
`CheckBox.selected`	Specifies whether the check box is selected (`true`) or deselected (`false`).

Event summary for the CheckBox class

Event	Description
`CheckBox.click`	Triggered when the mouse is pressed over a button instance.

DateChooser component

The DateChooser component is a calendar that allows users to select a date. It has buttons that allow users to scroll through months and click on a date to select it. You can set parameters that indicate the month and day names, the first day of the week, and any disabled dates, as well as highlighting the current date.

The DateChooser can be used anywhere you want a user to select a date. For example, you could use a DateChooser component in a hotel reservation system with certain dates selectable and others disabled. You could also use the DateChooser component in an application that displays current events, such as performances or meetings, when a user chooses a date.

DataChooser parameters

The following are authoring parameters that you can set for each DataChooser component instance in the Property inspector.

monthNames sets the month names that are displayed in the heading row of the calendar. The value is an array and the default value is [“January“,“February“,“March“,“April“,“May“,“June“,“July“,“August“,“September“,“October“,“November“,“December“].

dayNames sets the names of the days of the week. The value is an array and the default value is [“S“,“M“,“T“,“W“,“T“,“F“,“S“,].

firstDayOfWeek indicates which day of the week (0-6, 0 being the first element of the `dayNames` array) is displayed in the first column of the DateChooser. This property changes the display order of the day columns.

disabledDays indicates the disabled days of the week. This parameter is an array and can have up to 7 values. The default value is [] (an empty array).

showToday indicates whether or not to highlight today’s date. The default value is `true`.

You can write script to control these and additional options for the DateChooser component using its properties, methods, and events. For more information, see “DateChooser class” in the Flash documentation.

Using styles with the DateChooser component

You can set style properties to change the appearance of a DateChooser instance. If the name of a style property ends in “Color”, it is a color style property and behaves differently than non-color style properties.

A DateChooser component supports the following Halo styles:

Style	Description
`themeColor`	The glow color for the rollover and selected dates. This is the only color style that doesn’t inherit its value. Possible values are "haloGreen", "haloBlue", and "haloOrange".
`color`	The text of a component label.
`disabledColor`	The disabled color for text.
`fontFamily`	The font name for text.
`fontSize`	The point size for the font.
`fontStyle`	The font style: either "normal", or "italic".
`fontWeight`	The font weight: either "normal", or "bold".
`textDecoration`	The text decoration: either "none", or "underline".

Property summary for the DateChooser class

Method	Description
`DateChooser.dayNames`	An array indicating the names of the days of the week.
`DateChooser.disabledDays`	An array indicating the days of the week that are disabled for all applicable dates in the date chooser.
`DateChooser.disabledRanges`	A range of disabled dates or a single disabled date.
`DateChooser.displayedMonth`	A number indicating an element in the `monthNames` array to display in the date chooser.
`DateChooser.displayedYear`	A number indicating the year to display.
`DateChooser.firstDayOfWeek`	A number indicating an element in the `dayNames` array to display in the first column of the date chooser.
`DateChooser.monthNames`	An array of strings indicating the month names.
`DateChooser.selectableRange`	A single selectable date or a range of selectable dates.
`DateChooser.selectedDate`	A Date object indicating the selected date.
`DateChooser.showToday`	A Boolean value indicating whether the current date is highlighted.

Event summary for the DateChooser class

Method	Description
`DateChooser.change`	Broadcast when a date is selected.
`DateChooser.scroll`	Broadcast when the month buttons are pressed.

Label component

A label component is a single line of text. You can specify that a label be formatted with HTML. You can also control alignment and sizing of a label. Label components don't have borders, cannot be focused, and don't broadcast any events.

Use a Label component to create a text label for another component in a form, such as a "Name:" label to the left of a TextInput field that accepts a user's name. If you're building an application using components based on version 2 (v2) of the Macromedia Component Architecture, it's a good idea to use a Label component instead of a plain text field because you can use styles to maintain a consistent look and feel.

Label parameters

The following are authoring parameters that you can set for each Label component instance in the Property inspector or in the Component Inspector panel:

text indicates the text of the label; the default value is Label.

html indicates whether the label is formatted with HTML (`true`) or not (`false`). If the `html` parameter is set to `true`, a Label cannot be formatted with styles. The default value is `false`.

autoSize indicates how the label sizes and aligns to fit the text. The default value is none. The parameter can be any of the following four values:

- none—the label doesn't resize or align to fit the text.
- left—the right and bottom sides of the label resize to fit the text. The left and top sides don't resize.
- center—the bottom side of the label resizes to fit the text. The horizontal center of the label stays anchored at the its original horizontal center position.
- right—the left and bottom sides of the label resize to fit the text. The top and right side don't resize.

Using styles with the Label component

You can set style properties to change the appearance of a label instance. All text in a Label component instance must share the same style. For example, you can't set the `color` style to `"blue"` for one word in a label and to `"red"` for the second word in the same label.

If the name of a style property ends in "Color", it is a color style property and behaves differently than non-color style properties. A Label component supports the following styles:

Style	Description
`color`	The default color for text.
`embedFonts`	The fonts to embed in the document.
`fontFamily`	The font name for text.
`fontSize`	The point size for the font.
`fontStyle`	The font style, either "normal",or "italic".
`fontWeight`	The font weight, either "normal" or "bold".
`textAlign`	The text alignment: either "left", "right", or "center".
`textDecoration`	The text decoration, either "none" or "underline".

Property summary for the Label class

Property	Description
`Label.autoSize`	A string that indicates how a label sizes and aligns to fit the value of its `text` property. There are four possible values: `"none"`, `"left"`, `"center"`, and `"right"`. The default value is `"none"`.
`Label.html`	A Boolean value that indicates whether a label can be formatted with HTML (`true`) or not (`false`).
`Label.text`	The text on the label.

List component

The List component is a scrollable single- or multiple-selection list box. A list can also display graphics, including other components.

The List component uses a zero-based index, where the item with index 0 is the top item displayed. When adding, removing, or replacing list items using the List class methods and properties, you may need to specify the index of the list item.

You can set up a list so that users can make either single or multiple selections. For example, a user visiting an e-commerce website needs to select which item to buy. There are 30 items, and the user scrolls through a list and selects one by clicking it.

You can also design a list that uses custom movie clips as rows so you can display more information to the user. For example, in an e-mail application, each mailbox could be a List component and each row could have icons to indicate priority and status.

List component parameters

The following are authoring parameters that you can set for each List component instance in the Property inspector or in the Component Inspector panel:

data An array of values that populate the data of the list. The default value is [] (an empty array). There is no equivalent runtime property.

labels An array of text values that populate the label values of list. The default value is [] (an empty array). There is no equivalent runtime property.

multipleSelection A Boolean value that indicates whether you can select multiple values (true) or not (false). The default value is false.

rowHeight indicates the height, in pixels, of each row. The default value is 20. Setting a font does not change the height of a row.

You can write scripts to set additional options for List instances using its methods, properties, and events. For more information, see “List class” in the Flash documentation.

Customizing the List component

You can transform a List component horizontally and vertically both while authoring and at runtime.

When a list is resized, the rows of the list shrink horizontally, clipping any text within them. Vertically, the list adds or removes rows as needed. Scroll bars position themselves automatically.

Using styles with the List component

You can set style properties to change the appearance of a List component.

A List component uses the following Halo styles:

Style	Description
`alternatingRowColors`	Specifies colors for rows in an alternating pattern. The value can be an array of two or more colors, for example, 0xFF00FF, 0xCC6699, and 0x996699.
`backgroundColor`	The background color of the list. This style is defined on a class style declaration, ScrollSelectList.
`borderColor`	The black section of a three-dimensional border or the color section of a two-dimensional border.
`borderStyle`	The bounding box style. The possible values are: "none", "solid", "inset" and "outset". This style is defined on a class style declaration, ScrollSelectList.
`defaultIcon`	Name of the default icon to use for list rows. The default value is undefined.
`rollOverColor`	The color of a rolled over row.
`selectionColor`	The color of a selected row.
`selectionEasing`	A reference to an easing equation (function) used for controlling programmatic tweening.
`disabledColor`	The disabled color for text.
`textRollOverColor`	The color of text when the pointer rolls over it.
`textSelectedColor`	The color of text when selected.
`selectionDisabledColor`	The color of a row if it has been selected and disabled.
`selectionDuration`	The length of any transitions when selecting items.
`useRollOver`	Determines whether rolling over a row activates highlighting.

Method summary for the List class

Method	Description
`List.addItem()`	Adds an item to the end of the list.
`List.addItemAt()`	Adds an item to the list at the specified index.
`List.getItemAt()`	Returns the item at the specified index.
`List.removeAll()`	Removes all items from the list.
`List.removeItemAt()`	Removes the item at the specified index.
`List.replaceItemAt()`	Replaces the item at the specified index with another item.
`List.setPropertiesAt()`	Applies the specified properties to the specified item.
`List.sortItems()`	Sorts the items in the list according to the specified compare function.
`List.sortItemsBy()`	Sorts the items in the list according to a specified property.

Property summary for the List class

Property	Description
`List.cellRenderer`	Assigns the class or symbol to use to display each row of the list.
`List.dataProvider`	The source of the list items.
`List.hPosition`	The horizontal position of the list.
`List.hScrollPolicy`	Indicates whether the horizontal scroll bar is displayed (`"on"`) or not (`"off"`).
`List.iconField`	A field within each item to be used to specify icons.
`List.iconFunction`	A function that determines which icon to use.
`List.labelField`	Specifies a field of each item to be used as label text.
`List.labelFunction`	A function that determines which fields of each item to use for the label text.
`List.length`	The length of the list in items. This property is read-only.
`List.maxHPosition`	Specifies the number of pixels the list can scroll to the right, when `List.hScrollPolicy` is set to `"on"`.
`List.multipleSelection`	Indicates whether multiple selection is allowed in the list (`true`) or not (`false`).
`List.rowCount`	The number of rows that are at least partially visible in the list.
`List.rowHeight`	The pixel height of every row in the list.
`List.selectable`	Indicates whether the list is selectable (`true`) or not (`false`).
`List.selectedIndex`	The index of a selection in a single-selection list.
`List.selectedIndices`	An array of the selected items in a multiple-selection list.
`List.selectedItem`	The selected item in a single-selection list. This property is read-only.
`List.selectedItems`	The selected item objects in a multiple-selection list. This property is read-only.
`List.vPosition`	Scrolls the list so the topmost visible item is the number assigned.
`List.vScrollPolicy`	Indicates whether the vertical scroll bar is displayed (`"on"`), not displayed (`"off"`), or displayed when needed (`"auto"`).

NumericStepper component

The NumericStepper component allows a user to step through an ordered set of numbers. The component consists of a number displayed beside small up and down arrow buttons. When a user pushes the buttons, the number is raised or lowered incrementally. If the user clicks either of the arrow buttons, the number increases or decreases, based on the value of the stepSize parameter, until the user releases the mouse or until the maximum or minimum value is reached.

The NumericStepper only handles numeric data. Also, you must resize the stepper while authoring to display more than two numeric places (for example, the numbers 5246 or 1.34).

Using the NumericStepper component

The NumericStepper can be used anywhere you want a user to select a numeric value. For example, you could use a NumericStepper component in a form to allow a user to set their credit card expiration date. In another example, you could use a NumericStepper to allow a user to increase or decrease a font size.

NumericStepper parameters

The following are authoring parameters that you can set for each NumericStepper component instance in the Property inspector.

value sets the value of the current step. The default value is 0.

minimum sets the minimum value of the step. The default value is 0.

maximum sets the maximum value of the step. The default value is 10.

stepSize sets the unit of change for the step. The default value is 1.

You can write code to control these and additional options for NumericStepper components using its properties, methods, and events.

Customizing the NumericStepper component

You can transform a NumericStepper component horizontally and vertically both while authoring and at runtime.

Resizing the NumericStepper component does not change the size of the down and up arrow buttons. If the stepper is resized greater than the default height, the stepper buttons are pinned to the top and the bottom of the component. The stepper buttons always appear to the right of the text box.

Using styles with the NumericStepper component

You can set style properties to change the appearance of a stepper instance. If the name of a style property ends in "Color", it is a color style property and behaves differently than non-color style properties.

A NumericStepper component supports the following Halo styles:

Style	Description
`themeColor`	The background of a component. This is the only color style that doesn't inherit its value. Possible values are "haloGreen", "haloBlue", and "haloOrange".
`color`	The text of a component label.
`disabledColor`	The disabled color for text.
`fontFamily`	The font name for text.
`fontSize`	The point size for the font.
`fontStyle`	The font style; either "normal", or "italic".
`fontWeight`	The font weight; either "normal", or "bold".

Style	Description
`textDecoration`	The text decoration; either "none", or "underline".
`textAlign`	The text alignment; either "left", "right", or "center".

Property summary for the NumericStepper class

Property	Description
`NumericStepper.maximum`	A number indicating the maximum range value.
`NumericStepper.minimum`	A number indicating the minimum range value.
`NumericStepper.nextValue`	A number indicating the next sequential value. This property is read-only.
`NumericStepper.previousValue`	A number indicating the previous sequential value. This property is read-only.
`NumericStepper.stepSize`	A number indicating the unit of change for each step.
`NumericStepper.value`	A number indicating the current value of the stepper.

Event summary for the NumericStepper class

Event	Description
`NumericStepper.change`	Triggered when the value of the step changes.

RadioButton component

A radio button is a fundamental part of any form or web application. You can use radio buttons wherever you want a user to make one choice from a group of options. For example, you would use radio buttons in a form to ask which credit card a customer is using to pay.

RadioButton parameters

The following are authoring parameters that you can set for each RadioButton component instance in the Property inspector or in the Component Inspector panel:

label sets the value of the text on the button; the default value is Radio Button.

data is the value associated with the radio button. There is no default value.

groupName is the group name of the radio button. The default value is radioGroup.

selected sets the initial value of the radio button to selected (true) or unselected (false). A selected radio button displays a dot inside it. Only one radio button within a group can have a selected value of true. If more than one radio button within a group is set to true, the radio button that is instantiated last is selected. The default value is false.

labelPlacement orients the label text on the button. This parameter can be one of four values: left, right, top, or bottom; the default value is right. For more information, see `RadioButton.labelPlacement`.

You can write scripts to set additional options for RadioButton instances using the methods, properties, and events of the RadioButton class. For more information, see “RadioButton class” in the Flash documentation.

Customizing the RadioButton component

You can transform a RadioButton component horizontally and vertically both while authoring and at runtime.

The bounding box of a RadioButton component is invisible and also designates the hit area for the component. If you increase the size of the component, you also increase the size of the hit area.

If the component's bounding box is too small to fit the component label, the label clips to fit.

Using styles with the RadioButton component

You can set style properties to change the appearance of a RadioButton. If the name of a style property ends in "Color", it is a color style property and behaves differently than non-color style properties.

A RadioButton component uses the following Halo styles:l

Style	Description
`themeColor`	The background of a component. This is the only color style that doesn't inherit its value. Possible values are "haloGreen", "haloBlue", and "haloOrange".
`color`	The text of a component label.
`disabledColor`	The disabled color for text.
`fontFamily`	The font name for text.
`fontSize`	The point size for the font.
`fontStyle`	The font style; either "normal", or "italic".
`fontWeight`	The font weight; either "normal", or "bold".

Property summary for the RadioButton class

Property	Description
`RadioButton.data`	The value associated with a radio button instance.
`RadioButton.groupName`	The group name for a radio button group or radio button instance.
`RadioButton.label`	The text that appears next to a radio button.
`RadioButton.labelPlacement`	The orientation of the label text in relation to a radio button.
`RadioButton.selected`	Sets the state of the radio button instance to selected and deselects the previously selected radio button.
`RadioButton.selectedData`	Selects the radio button in a radio button group with the specified data value.
`RadioButton.selection`	A reference to the currently selected radio button in a radio button group.

Event summary for the RadioButton class

Event	Description
`RadioButton.click`	Triggered when the mouse is pressed over a button instance.

ScrollPane component

The Scroll Pane component displays movie clips, JPEG files, and SWF files in a scrollable area. You can enable scroll bars to display images in a limited area. You can display content that is loaded from a local location, or from over the internet. You can set the content for the scroll pane both while authoring and at runtime using scripts.

Using the ScrollPane component

You can use a scroll pane to display any content that is too large for the area into which it is loaded. For example, if you have a large image and only a small space for it in an application, you could load it into a scroll pane.

ScrollPane parameters

The following are authoring parameters that you can set for each ScrollPane component instance in the Property inspector or in the Component Inspector panel:

contentPath indicates the content to load into the scroll pane. This value can be a relative path to a local SWF or JPEG file, or a relative or absolute path to a file on the internet.

hLineScrollSize indicates the number of units a horizontal scroll bar moves each time an arrow button is pressed. The default value is 5.

hPageScrollSize indicates the number of units a horizontal scroll bar moves each time the track is pressed. The default value is 20.

hScrollPolicy displays the horizontal scroll bars. The value can be "on", "off", or "auto". The default value is "auto".

scrollDrag is a Boolean value that allows a user to scroll the content within the scroll pane (true) or not (false). The default value is false.

vLineScrollSize indicates the number of units a vertical scroll bar moves each time an arrow button is pressed. The default value is 5.

vPageScrollSize indicates the number of units a vertical scroll bar moves each time the track is pressed. The default value is 20.

vScrollPolicy displays the vertical scroll bars. The value can be "on", "off", or "auto". The default value is "auto".

You can write code to control these and additional options for ScrollPane components using its properties, methods, and events.

Method summary for the ScrollPane class

Method	Description
`ScrollPane.getBytesLoaded()`	Returns the number of bytes of content loaded.
`ScrollPane.getBytesTotal()`	Returns the total number of content bytes to be loaded.
`ScrollPane.refreshPane()`	Reloads the contents of the scroll pane.

Property summary for the ScrollPane class

Method	Description
`ScrollPane.content`	A reference to the content loaded into the scroll pane.
`ScrollPane.contentPath`	An absolute or relative URL of the SWF or JPEG file to load into the scroll pane
`ScrollPane.hLineScrollSize`	The amount of content to scroll horizontally when an arrow button is pressed.
`ScrollPane.hPageScrollSize`	The amount of content to scroll horizontally when the track is pressed.
`ScrollPane.hPosition`	The horizontal pixel position of the scroll pane.
`ScrollPane.hScrollPolicy`	The status of the horizontal scroll bar. It can be always on (`"on"`), always off (`"off"`), or on when needed (`"auto"`). The default value is `"auto"`.
`ScrollPane.scrollDrag`	Indicates whether there is scrolling when a user presses and drags within the ScrollPane (`true`) or not (`false`). The default value is `false`.
`ScrollPane.vLineScrollSize`	The amount of content to scroll vertically when an arrow button is pressed.
`ScrollPane.vPageScrollSize`	The amount of content to scroll vertically when the track is pressed.
`ScrollPane.vPosition`	The vertical pixel position of the scroll pane.
`ScrollPane.vScrollPolicy`	The status of the vertical scroll bar. It can be always on (`"on"`), always off (`"off"`), or on when needed (`"auto"`). The default value is `"auto"`.

Event summary for the ScrollPane class

Method	Description
`ScrollPane.complete`	Broadcast when the scroll pane content is loaded.
`ScrollPane.progress`	Broadcast while the scroll bar content is loading.
`ScrollPane.scroll`	Broadcast when the scroll bar is pressed.

TextArea component

You can use a TextArea component wherever you need a multiline text field. If you need a single-line text field, use the "TextInput component" on page 225. For example, you could use a TextArea component as a comment field in a form. You could set up a listener that checks if field is empty when a user tabs out of the field. That listener could display an error message indicating that a comment must be entered in the field.

TextArea component parameters

The following are authoring parameters that you can set for each TextArea component instance in the Property inspector or in the Component Inspector panel:

text indicates the contents of the TextArea. You cannot enter carriage returns in the Property inspector or Component Inspector panel. The default value is "" (empty string).

html indicates whether the text is formatted with HTML (true) or not (false). The default value is false.

editable indicates whether the TextArea component is editable (true) or not (false). The default value is true.

wordWrap indicates whether the text wraps (true) or not (false). The default value is true.

You can write code to control these and additional options for TextArea components using its properties, methods, and events.

Using styles with the TextArea component

The TextArea component supports one set of component styles for all text in the field. However, you can also display HTML compatible with Flash Player HTML rendering. To display HTML text, set `TextArea.html` to `true`.

If the name of a style property ends in "Color", it is a color style property and behaves differently than non-color style properties. For more information, see "Using styles to customize component color and text" in the Flash documentation.

A TextArea component supports the following styles:

Style	Description
`color`	The default color for text.
`embedFonts`	The fonts to embed in the document.
`fontFamily`	The font name for text.
`fontSize`	The point size for the font.
`fontStyle`	The font style, either "normal",or "italic".
`fontWeight`	The font weight, either "normal" or "bold".
`textAlign`	The text alignment: either "left", "right", or "center".
`textDecoration`	The text decoration, either "none" or "underline".

Property summary for the TextArea class

Property	Description
`TextArea.editable`	A Boolean value indicating whether the field is editable (`true`) or not (`false`).
`TextArea.hPosition`	Defines the horizontal position of the text within the scroll pane.
`TextArea.hScrollPolicy`	Indicates whether the horizontal scroll bar is always on (`"on"`), never on (`"off"`), or turns on when needed (`"auto"`).S
`TextArea.html`	A flag that indicates whether the text field can be formatted with HTML.
`TextArea.length`	The number of characters in the text field. This property is read-only.
`TextArea.maxChars`	The maximum number of characters that the text field can contain.
`TextArea.maxHPosition`	The maximum value of `TextArea.hPosition`.
`TextArea.maxVPosition`	The maximum value of `TextArea.vPosition`.
`TextArea.password`	A Boolean value indicating whether the field is a password field (`true`) or not (`false`).
`TextArea.restrict`	The set of characters that a user can enter into the text field.
`TextArea.text`	The text contents of a TextArea component.
`TextArea.vPosition`	A number indicating the vertical scrolling position
`TextArea.vScrollPolicy`	Indicates whether the vertical scroll bar is always on (`"on"`), never on (`"off"`), or turns on when needed (`"auto"`).S
`TextArea.wordWrap`	A Boolean value indicating whether the text wraps (`true`) or not (`false`).

Event summary for the TextArea class

Event	Description
`TextArea.change`	Notifies listeners that text has changed.

TextInput component

The TextInput is a single-line component that wraps the native ActionScript TextField object. You can use styles to customize the TextInput component; when an instance is disabled its contents display in a color represented by the “disabledColor” style. A TextInput component can also be formatted with HTML, or as a password field that disguises the text.

You can use a TextInput component wherever you need a single-line text field. If you need a multiline text field, use the “TextArea component” on page 224. For example, you could use a TextInput component as a password field in a form. You could set up a listener that checks if field has enough characters when a user tabs out of the field. That listener could display an error message indicating that the proper number of characters must be entered.

TextInput component parameters

The following are authoring parameters that you can set for each TextInput component instance in the Property inspector or in the Component Inspector panel:

text specified the contents of the TextInput. You cannot enter carriage returns in the Property inspector or Component Inspector panel. The default value is "" (empty string).

editable indicates whether the TextInput component is editable (true) or not (false). The default value is true.

password indicates whether the field is a password field (true) or not (false). The default value is false.

You can write scripts to control these and additional options for TextInput components using its properties, methods, and events. For more information, see “TextInput class” in the Flash documentation.

Customizing the TextInput component

When a TextInput component is resized, the border is resized to the new bounding box. The TextInput component doesn’t use scroll bars, but the insertion point scrolls automatically as the user interacts with the text. The text field is then resized within the remaining area; there are no fixed-size elements in a TextInput component. If the TextInput component is too small to display the text, the text is clipped.

Using styles with the TextInput component

A TextInput component supports the following styles:

Style	Description
`color`	The default color for text.
`fontFamily`	The font name for text.
`fontSize`	The point size for the font.
`fontStyle`	The font style, either "normal",or "italic".
`fontWeight`	The font weight, either "normal" or "bold".
`textAlign`	The text alignment: either "left", "right", or "center".
`textDecoration`	The text decoration, either "none" or "underline".

Property summary for the TextInput class

Property	Description
`TextInput.editable`	A Boolean value indicating whether the field is editable (`true`) or not (`false`).
`TextInput.hPosition`	The horizontal scrolling position of the text field.
`TextInput.length`	The number of characters in a TextInput text field. This property is read only.

Property	Description
`TextInput.maxChars`	The maximum number of characters that a user can enter in a TextInput text field.
`TextInput.maxHPosition`	The maximum possible value for TextField.hPosition. This property is read-only.
`TextInput.password`	A Boolean value that indicates whether or not the input text field is a password field that hides the entered characters.
`TextInput.restrict`	Indicates which characters a user can enter in a text field.
`TextInput.text`	Sets the text content of a TextInput text field.

Event summary for the TextInput class

Event	Description
`TextInput.change`	Triggered when the Input field changes.
`TextInput.enter`	Triggered when the enter key is pressed.

Tree component

The Tree component allows a user to view hierarchical data. The tree appears within a box like the List component, but each item in a tree is called a *node* and can be either a *leaf* or a *branch*. By default, a leaf is represented by a text label beside a file icon and a branch is represented by a text label beside a folder icon with a disclosure triangle that a user can open to expose children. The children of a branch can either be leaves or branches themselves.

The data of a tree component must be provided from an XML data source. For more information, see the next section.

When a Tree instance has focus either from clicking or tabbing, you can use the following keys to control it:

Key	Description
Down arrow	Moves selection down one.
Up arrow	Moves selection up one.
Right arrow	Opens a selected branch node. If a branch is already open, moves to first child node.
Left arrow	Closes a selected branch node. If on a leaf node of a closed branch node, moves to parent node.
Space	Opens or closes a selected branch node.
End	Moves selection to the bottom of the list.
Home	Moves selection to the top of the list.
Page Down	Moves selection down one page.
Page Up	Moves selection up one page.

Key	Description
Control	Allows multiple noncontiguous selections.
Shift	Allows multiple contiguous selections.

The Tree component can be used to represent hierarchical data such as e-mail client folders, file browser panes, or category browsing systems for inventory. Most often, the data for a tree is retrieved from a server in the form of XML, but it can also be well-formed XML that is created while authoring in Director. The best way to create XML for the tree is to create a Lingo object using the XML Parser Xtra or an XML object using the Flash Asset Xtra. After you create a Lingo object that contains an XML data source (or load one from an external source) you assign it to `Alert.cancelLabel`.

Formatting XML for the Tree component

The Tree component is designed to display hierarchical data structures. XML is the data model for the Tree component. It is important to understand the relationship of the XML data source to the Tree component.

Consider the following XML data source sample:

```
<node>
  <node label="Mail">
       <node label="INBOX"/>
       <node label="Personal Folder">
           <node label="Business" isBranch="true" />
           <node label="Demo" isBranch="true" />
           <node label="Personal" isBranch="true" />
           <node label="Saved Mail" isBranch="true" />
           <node label="bar" isBranch="true" />
       </node>
       <node label="Sent" isBranch="true" />
       <node label="Trash"/>
  </node>
</node>
```

Note: The `isBranch` attribute is read-only; you cannot set it directly. To set it, call the `Tree.setIsBranch()` method.

Nodes in the XML data source can have any name. Notice in the sample above that each node is named with the generic name *node*. The tree reads through the XML and builds the display hierarchy based on the nested relationship of the nodes.

Each XML node can be displayed as one of two types in the Tree: branch or leaf. Branch nodes can contain multiple child nodes and appear as a folder icon with a disclosure triangle that allows users to open and close the folder. Leaf nodes appear as a file icon and cannot contain child nodes. Both leaves and branches can be roots; root nodes appear at the top level of the tree and have no parent.

There are many ways to structure XML. The Tree component is not designed to use all types of XML structures, so it's important to use XML that the Tree component can interpret. Do not nest node attributes in a child node; each node should contain all its necessary attributes. Also, the attributes of each node should be consistent to be useful. For example, to describe a mailbox structure with a Tree component, use the same attributes on each node (message, data, time, attachments, and so on). This allows the tree to know what it expects to render, and allows you to loop through the hierarchy to compare data.

When a Tree displays a node it uses the `label` attribute of the node by default as the text label. If any other attributes exist, they become additional properties of the node's attributes within the Tree.

The actual root node is interpreted as the Tree component itself. This means that the `firstChild` (in the sample, `<node label="Mail">`), is rendered as the root node in the Tree view. This means that a tree can have multiple root folders. In this sample, there is only one root folder displayed in the tree: "Mail". However, if you were to add sibling nodes at that level in the XML, multiple root nodes would be displayed in the Tree.

Tree parameters

The following are authoring parameters that you can set for each Tree component instance in the Property inspector.

multipleSelection A Boolean value that indicates whether a user can select multiple items (`true`) or not (`false`). The default value is false.

rowHeight The height of each row in pixels. The default value is 20.

You can write Lingo or JavaScript to control these and additional options for the Tree component using its properties, methods, and events.

You cannot enter data parameters in the Property inspector for the Tree component like you can with other components.

Customizing the Tree component

You can size a Tree component horizontally and vertically both while authoring and at runtime. While authoring, select the component on the Stage and drag the resize handles. At runtime, use the `setSize()` method. When a tree isn't wide enough to display the text of the nodes, the text is clipped.

Method summary for the Tree class

Method	Description
`PopUpManager.createPopUp()`	Adds a node to a tree instance.
`Accordion.createSegment()`	Adds a node at a specific location in a tree instance.
`Accordion.destroyChildAt()`	Specifies whether the folder is a branch (has a folder icon and an expander arrow).
`Accordion.getChildAt()`	Indicates whether a branch is open or closed.
`Tree.getDisplayIndex()`	Returns the display index of a given node.
`Tree.getNodeDisplayedAt()`	Returns the display index of a given node.
`Tree.getTreeNodeAt()`	Returns a node on the root of the tree.
`Tree.removeAll()`	Removes all nodes from a tree instance and refreshes the tree.
`Tree.removeTreeNodeAt()`	Removes a node at a specified position and refreshes the tree.
`Tree.setIsBranch()`	Indicates whether a node is a branch (receives folder icon and expander arrow).

Method	Description
`Tree.setIcon()`	Specifies whether a node is open or closed.
`Tree.setIsOpen()`	Specifies a symbol to be used as an icon for a node.

Property summary for the Tree class

Property	Description
`Alert.cancelLabel`	Specifies an XML data source.
`Alert.noLabel`	Specifies the first node at the top of the display.
`Alert.okLabel`	Specifies the selected node in a tree instance.
`Tree.selectedNode`	Specifies the selected nodes in a tree instance.

Event summary for the Tree class

Event	Description
`Tree.nodeClose`	Broadcast when a node is closed by a user.
`Alert.click`	Broadcast when a node is opened by a user.

CHAPTER 10
Sound and Synchronization

You can give your movie added appeal by including a sound track, a voice-over, ambient noises, or other sounds.

With Macromedia Director MX 2004, you can control when sounds start and stop, how long they last, their quality and volume, and several other effects. Using Macromedia Shockwave Audio, you can compress sounds for easier distribution and stream them from an Internet source. You can also incorporate Windows Media Audio (WMA) in your Director movies.

The media synchronization features in Director let you synchronize events in a movie to precise cue points embedded in sound.

Sound makes significant demands on a computer's processing power, so you might need to manage sounds carefully to make sure they don't adversely affect your movie's performance.

Scripting gives Director more flexibility when playing sound and can help overcome performance concerns. You can use it to play sound in ways not possible with the Score alone. Using Lingo or JavaScript syntax, you can do the following:

- Turn sound on and off in response to movie events.
- Control sound volume.
- Control the pan of a sound relative to the pan of a QuickTime VR movie. (For more information about using video and QuickTime VR in Director, see "Using Digital Video" on page 243.)
- Control the sound in a Windows Media Audio file.
- Preload sound into memory, queue multiple sounds, and define precise loops.
- Synchronize sound and animation precisely.

Importing internal and linked sounds

Director handles sounds as either internal or linked. You can determine whether a sound is internal or linked when you import it. Each type of sound has advantages for different situations.

Director stores all the sound data for an internal sound cast member in a movie or cast file and loads the sound completely into RAM before playing it. After an internal sound is loaded, it plays very quickly. This makes internal sound best for short sounds, such as beeps or clicks, that recur frequently in your movie. For the same reason, making a large sound file an internal sound is not a good choice because the sound might use too much memory.

Director does not store sound data in a linked sound cast member. Instead, it keeps a reference to a sound file's location and imports the sound data each time the sound begins playing. Because the sound is never entirely loaded into RAM, the movie uses memory more efficiently.

Also, Director streams many sounds, which means it begins playing the sound while the rest of the sound continues to load from its source, whether on disk or over the Internet. This can dramatically improve the downloading performance of large sounds. Linked sounds are best for longer sounds such as voice-overs or nonrepeating music.

Director can stream the following sounds:

- QuickTime, Shockwave Audio, and MP3 sounds that are linked from a URL
- QuickTime, Shockwave Audio, MP3, AIFF, and WAV sounds that are linked to a local file

Director imports AIFF and WAV sounds (both compressed and uncompressed), AU, Shockwave Audio, and MP3. For best results, use sounds that have 8- or 16-bit depth and a sampling rate of 44.1, 22.050, or 11.025 kHz.

To import a sound:

1 Select File > Import.

2 Select sound files to import.

3 To determine whether the imported sounds are internal or linked, select a Media option:

 Standard Import makes all the selected sounds internal sound cast members.

 Link to External File makes all the selected sounds linked.

 Include Original Data for Editing lets you edit original sound files in Director.

4 Click Import.

Note: If you're authoring on a Macintosh computer that has an audio input or attached microphone, you can record sounds directly into a movie's cast by selecting Insert > Media Element > Sound. The Sound method opens the Macintosh sound recording dialog box. Director for Windows has no equivalent feature.

Setting sound cast member properties

You can use sound cast member properties to make a sound loop, change its name, change the external sound file to which it's linked (if it's a linked sound), and set its unload priority.

To set sound cast member properties:

1 Select a sound cast member.

2 Click the Sound tab in the Property inspector.

 There are several noneditable options in the Property inspector's Sound tab:

 - The duration of the sound
 - The sample rate, sample size, and channels

3 To make the sound play continuously, click Loop (for more information, see "Looping a sound" on page 234).

4 To play the sound, click the Play button.

5 Click the Member tab in the Property inspector.

The following noneditable settings appear:

- The cast member size in kilobytes
- The cast member creation and edit dates
- The name of the last person who modified the cast member

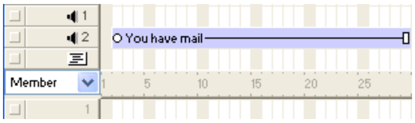
6 To view or edit the cast member name, use the Name text box.

7 To change the external sound file to which the cast member is linked (if it has been imported as a Linked to External File sound), enter a new path and file in the Filename text box. You can also use the Browse button to select a new file.

8 To specify how Director removes the cast member from memory if memory is low, select an option from the Unload pop-up menu. For more information, see "Controlling cast member unloading" on page 47.

Controlling sound in the Score

You control sounds in the Score in much the same way that you control sprites. You place sounds in one of the two sound channels at the top of the Score and extend the sounds through as many frames as required.



Unless you use a behavior or other script to override the Score's sound channels, sounds play only as long as the playhead is in the frames that contain the sound. After a sound begins playing, it plays at its own speed. Director can't control the speed at which a sound plays. If a sound is not set to loop, it stops playing at the end, even if the sprite specifies a longer duration. For more information, see "Looping a sound" on page 234.

Note: You can speed up or slow down a sound by converting it to a sound-only QuickTime movie and using the movieRate sprite property.

In addition to the two sound channels in the Score, Director can use as many as six additional sound channels simultaneously. However, the additional channels are accessible only from Lingo or JavaScript syntax or from behaviors. Available RAM and the computer's speed are the constraints on how many sounds Director can use effectively.

To place a sound in the Score:

1 If the sound channels are not visible, click the Hide/Show Effects Channels button at the upper right side of the Score.

2 Do any of the following:

- Drag a sound cast member from a Cast window to a frame in one of the sound channels.
- Double-click a frame in the sound channel, and then select a sound from the Frame Properties: Sound dialog box. You can also preview any sound cast member in the movie from this dialog box.
- Drag a sound to the Stage to place it into the first available sound channel in the current frame of the Score.

3 Extend the sound through as many frames as necessary.

New sounds are assigned the same number of frames as set for sprites in the Sprite Preferences dialog box. You might need to adjust the number of frames to make the sound play completely or change a tempo setting to make the playhead wait for the sound to finish. For more information, see “Synchronizing media” on page 240.

Note: Sound in the last frame of a movie continues to play (but not loop) until the next movie begins or you exit from the application. This sound can provide a useful transition while Director loads the next movie. You can stop the sound by using the puppetSound method.

Looping a sound

You might find that you want to play a sound repeatedly to create a continuous sound effect, such as the sound of a person walking. A looped sound repeats as long as the playhead is in a frame where the sound is set. See “Importing internal and linked sounds” on page 231.

To make a sound loop:

1 Select a sound cast member.

2 On the Sound tab in the Property inspector, select the Loop option.

You can also loop sounds with Lingo or JavaScript syntax. For more information, see “Playing sounds with Lingo or JavaScript syntax” on page 235.

Using sound in Windows

The following issues are specific to managing sound for Windows:

- In Windows, a sound that is already playing in either sound channel overrides the sound in a QuickTime or AVI video or in Macromedia Flash content. It also prevents the video sound from playing even after the sound in the sound channel stops. After the sound in a digital video starts, however, it overrides a sound in either sound channel.
- To mix QuickTime audio tracks with internal Director sounds, use the `soundDevice` system property to specify QT3Mix or install the Microsoft DirectSound sound driver software version 5.0 or later (available from www.microsoft.com), and use the `soundDevice` property to specify DirectSound. For more information, see the Scripting Reference topics in the Director Help Panel. (Windows NT 4 does not support DirectSound 5.) Check the Director Support Center at www.macromedia.com/go/director_support for the latest developments related to this issue.
- The default number of sounds that Director can mix in Windows is eight. This number can be decreased by modifying the value for MixMaxChannels in the Director.ini file in the Director folder.

Playing sounds with Lingo or JavaScript syntax

Lingo or JavaScript syntax lets you play and control sounds regardless of the settings in the Score. You can use these scripts to play sounds, turn them on and off, and play external sounds that aren't cast members. Using script to play sounds lets you control the exact timing of when sounds start and stop. Lingo or JavaScript syntax also lets you play only part of a sound cast member or play several sounds in succession without interruption.

Sounds played by Director play at the volume that is set in the computer's sound level control. You can use Lingo or JavaScript syntax to modify the computer's sound level to suit the needs of your movie or to modify the volume of the sound channel.

You can also use Lingo or JavaScript syntax to control and stream Shockwave Audio. For more information, see "Playing Shockwave Audio, Windows Media Audio, and MP3 audio with Lingo or JavaScript syntax" on page 239.

Playing sound cast members

After you import a sound as a cast member, you can control many aspects of how the sound plays.

To play sound cast members regardless of the settings in the Score:

- Use the `queue()` and `play()` methods. The `queue()` method loads the sound into the Director RAM buffer so that it can be play immediately when called. The `play()` method starts the sound playing. If you omit the `queue()` method, the sound might not play immediately when called. For more information about these methods, see the Scripting Reference topics in the Director Help Panel.

The following statements load the sound called Siren into RAM and start it playing in sound channel 1:

```
sound(1).queue(member("Siren"))
sound(1).play()
```

To queue more than one sound to play in succession:

- Use the `queue()` method to list sounds in the order you want them to play. If you queue them before they play, Director plays the sounds with no pauses between the sounds. After the sounds are queued, you need only one `play()` method.

These statements queue the sound members Explosion and Siren and play them in succession in sound channel 2:

```
sound(2).queue(member("Explosion"))
sound(2).queue(member("Siren"))
sound(2).play()
```

To control how a queued sound plays:

- Include optional parameters in a property list within the `queue()` method. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

When you use `setPlayList()`, any previously set queue of sounds is replaced by the new playlist.

After queuing sounds, you can still control whether the queue is obeyed. You can select to interrupt loops with the `breakLoop()` method or to pause playback with the `pause()` method. The `playNext()` method lets you skip immediately to the next sound in the queue. For more information about sound methods, see the *Director Scripting Reference*.

Playing external sound files

In addition to playing sounds you have imported as cast members, you can also play external sound files that have not been imported.

To play external sound files that aren't cast members:

- Use the `sound playFile()` method. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

Playing external sound files from disk minimizes the amount of RAM that is used to play sounds. However, because the computer can read only one item from disk at a time, loading cast members or playing more than one sound from disk can cause unacceptable pauses when you use the `sound playFile()` method.

Controlling sound channels

You can use Lingo or JavaScript syntax to make actions in a movie dependent on whether a sound is playing. Lingo or JavaScript syntax lets you determine whether a sound is playing in a particular sound channel and control how a channel plays sound. For more information about the following methods and properties, see the Scripting Reference topics in the Director Help Panel.

- To determine whether a specific channel is playing a sound, use the `isBusy()` method.
- To turn off the current sound in a specific channel, use the `setPlayList()` method with `[ ]` as the new play list. This deletes the entire sound queue and leaves the current sound playing. Use the `stop()` method to stop the currently playing sound.
- To fade a specific channel's sound in and out, use the `fadeTo()` method.
- To control a specific sound channel's volume, specify the `volume` property.
- To control the left-to-right panning of a sound, specify the `pan` property.

About Windows Media Audio

Microsoft Windows Media Audio (WMA) is an audio codec designed by Microsoft for use with streaming content at CD quality. It is designed to resist data loss that can cause signal degradation and can improve download times for audio. It is similar to MP3, with two main advantages: it works better with low bit-rates (8-64 kbts per second) and it can, in general, produce better quality sound at a given bit-rate than MP3. It is generally recommended for music and general sounds, but not for voice. You can use WMA content in Director through the Windows Media feature. You must also have the proper decoders installed. For more information about using WMA and Windows Media in general, see "Using Windows Media files in Director" on page 252.

About Shockwave Audio

Shockwave Audio is a technology that makes sounds smaller and plays them faster from disk or over the Internet.

Shockwave Audio can compress the size of sounds by a ratio of up to 176:1 and is streamable, which means Director doesn't have to load the entire sound into RAM before it begins playing. Director starts to play the beginning of the sound while the rest of the sound is still streaming from its source, which can be from a disk or over the Internet. When used properly, the Shockwave Audio compression and streaming features provide fast playback of high-quality audio, even for users with relatively slow modem connections to the Internet.

Compression quality in Shockwave Audio

Although Shockwave Audio uses advanced compression technology that alters original sounds as little as possible, the more a sound is compressed, the more it is changed.

Set the amount of compression by selecting a bit rate setting in any of the Shockwave Audio Xtra extensions. The bit rate is not related to sampling rates you might have used in other audio programs. Try compressing the same sound at several different bit rates to see how the sound changes.

Select the bit rate that is appropriate for the intended delivery system (56K modem, ISDN, CD-ROM, broadband, hard disk, and so on), the type of movie, and the nature of the sound. Voice-over sound quality, for example, might not need to be as high as that of music. Test the sound on several systems to find the right balance between quality and performance.

The more compressed a sound is, the faster it streams. If you select to use a high quality and low degree of compression, a slow delivery system might not send the data fast enough, which results in gaps during playback. It's also important to consider your target audience: for example, using a lower data rate lets you target a wider audience, but at the expense of audio quality.

Note: Any sound compressed at less than 48 Kbps is converted to monaural.

Compressing internal sounds with Shockwave Audio

Shockwave Audio can compress any internal sounds in a movie. Although internal sounds are not streamed, compressing them with Shockwave Audio dramatically decreases the size of the sound data in a movie, shortens the download time from the Internet, and saves disk space.

You can use Shockwave Audio settings to specify compression settings for internal sound cast members. The selected compression settings apply to all internal sound cast members. You can't specify different settings for different cast members.

You can select compression settings at any time, but compression occurs only when the Director movie is compressed with the Create Projector, Save as Shockwave Movie, or Update Movies methods. When you create a projector, Director compresses sounds only if the Compressed option is turned on in the Projector Options dialog box. Compressing sounds can substantially increase the time required to compress a Director movie. For more information, see "Creating projectors" on page 460.

Note: Shockwave Audio does not compress SWA or MP3 audio sounds.

When you distribute a movie that contains sounds compressed with Shockwave Audio, the SWA Decompression Xtra is already included in the Macromedia Shockwave Player. If you compress sounds in Shockwave format in a projector, you must provide the SWA Decompression Xtra for the projector.

To have Director compress internal sound cast members when you create a projector, save a movie as Shockwave, or update the movie:

1 Select File > Publish Settings.

2 Select the Shockwave tab.

3 Select Shockwave Audio Compression Enabled to turn on compression.

4 Select a setting from the kBits/second pop-up menu.

5 Select Convert Stereo to Mono if you want to convert a stereo file to monaural.

6 Click OK.

Streaming linked Shockwave Audio and MP3 audio files

Director streams sounds that have been compressed with Shockwave Audio as well as MP3 audio files, from either a local disk or a URL. Before you can set up a streaming Shockwave Audio cast member, you must create a Shockwave Audio or MP3 file.

To create external Shockwave Audio files, do one of the following:

- In Windows, select Xtras > Convert WAV to SWA, and select the WAV files to convert.
- On the Macintosh, use the Peak LE 2 software to export Shockwave Audio sounds.

For both methods, the audio settings are similar to those for using Shockwave Audio to compress internal sounds. For more information, see “Compressing internal sounds with Shockwave Audio” on page 237.

Note: Converting WAV to SWA does not compress IMA compressed sounds.

To stream a linked Shockwave Audio or MP3 sound:

1 Select Insert > Media Element > Shockwave Audio.

This process creates a cast member that controls the streaming Shockwave Audio.

2 In the SWA Cast Member Properties dialog box that appears, click Browse and select a Shockwave Audio file on a local disk, or enter a URL in the Link Address text box.

Unless you select a file in the same folder as the movie, the movie always links to the exact location that you specify. Be sure to link to the correct location.

3 Set the remaining cast member properties in the Property inspector, as described in the following list:

- To set the volume of the sound, use the Volume slider in the SWA tab in the Property inspector.
- To select the sound channel for the sound, select a number from the Channel pop-up menu in the SWA tab. To avoid potential conflicts, select Any, which causes the sound to play in the highest numbered available sound channel.
- To specify the size of the stream buffer, use the Preload option in the SWA tab. Director attempts to load enough sound data to play for the specified time in seconds. This prevents gaps in sounds that play over slow or interruption-prone Internet connections.

4 Drag the Shockwave Audio cast member to a sprite channel (*not* one of the sound channels) to create a sprite. Extend the sprite through all frames in which the sound should play, or use the tempo channel to make the movie wait for the end of the sound. For more information, see "Synchronizing media" on page 240.

You can't place streaming audio cast members in the sound channels. The sound streams from the source location when the movie plays.

Playing Shockwave Audio, Windows Media Audio, and MP3 audio with Lingo or JavaScript syntax

Use SWA script to preload and control SWA and MP3 sounds and to determine how much sound has streamed over the Internet.

Script that controls other types of sounds can also control streaming SWA and MP3 sounds by controlling the sound channel in which the sound plays. For more information about the following methods and properties, see the Scripting Reference topics in the Director Help Panel.

- To preload part of a streaming sound file into memory, use the `preLoadBuffer member` method.
- To specify the amount of a streaming cast member to download before playback begins, set the `preLoadTime` cast member property.
- To determine what percentage of a streaming sound file has played, test the `percentPlayed` cast member property.
- To determine the percent of a streaming file that has streamed from an Internet server, test the `percentStreamed` cast member property.
- To specify the sound channel in which a streaming sound plays, set the `soundChannel` property.
- To begin playback of a streaming cast member, use the `play member` method.
- To pause a streaming sound file, use the `pause member` method.
- To stop a streaming sound file, use the `stop member` method.
- To determine the state of a streaming sound file, test the `state` cast member property.
- To determine whether an error occurred when streaming a sound file, use the `getError()` method.
- To obtain a string describing an error that occurred when streaming a sound file, use the `getErrorString()` method.
- To determine the length of a streaming sound file, use the `duration` cast member property.
- To determine the bit rate of a streaming sound cast member, test the `bitRate` cast member property.
- To determine the original bit depth of a streaming sound, test the `bitsPerSample` property.
- To determine the sample rate of the original sound used for a streaming cast member, test the `sampleRate` cast member property.
- To determine the number of channels in a streaming sound, test the `numChannels` streaming cast member property.
- To specify a streaming sound's volume, specify the `volume` streaming cast member property.
- To specify a streaming sound file's URL, set the `URL` cast member property.
- To obtain or set the copyright text in a streaming sound file, test or set the `copyrightInfo` cast member property.

Synchronizing media

To pause the playhead until a specified cue point in a sound or digital video is reached, you can use the Wait for Cue Point option in the Tempo dialog box. You can also use this method to wait for the end of the sound or digital video, even if it has no cue points. Cue points can also be used to trigger events that scripts can interpret. For more information, see "Synchronizing sound with Lingo or JavaScript syntax" on page 241.

Note: The methods discussed in this section for synchronizing media apply to sound and digital video. For more information about using video in Director MX see "Using Digital Video" on page 243.

For example, you can use cue points to make text appear in time with narration. First, use a program such as Peak LE 2 to place cue points in the sound file that correspond to the times when you want the text to appear on Stage. In Director, use the Tempo dialog box to pause the playhead at the frame where the corresponding text appears until the voice-over reaches the proper cue point.

In Windows, use Sound Forge 4.0 or later or Cool Edit 96 or later to define cue points (called markers or regions within these programs). For instructions, see the Readme Windows Sound Loop-Cue.txt file in the Director application folder.

On the Macintosh, use Sound Edit 16 2.07 or later, or Peak LE 2 or later, to define cue points in AIFF and Shockwave Audio sounds and in QuickTime digital videos.

Note: You can insert cue points into QuickTime files only on the Macintosh; however, the cue points can be used on both platforms.

AVI digital video does not support cue points.

To use cue points:

1 Place cue points in a sound file or (on the Macintosh only) in a QuickTime file.

Use an audio-editing program to define cue points in sounds and digital videos.

2 Import the sound or digital video into Director.

Note: Digital video is always linked, whether you select the Standard Import option or the Link to External File option in the Import dialog box.

3 Place the sound or digital video in a channel in the Score, and extend it through all the frames in which you want it to play.

4 Double-click the frame in the tempo channel where you want the playhead to wait for a cue point.

5 In the Tempo dialog box, select Wait for Cue Point.

6 Select the sound or digital video from the Channel pop-up menu.

7 Select the desired cue point from the Cue Point pop-up menu.

Select the End or Next cue point or any named or numbered cue point in the sound or digital video. Director recognizes the end of a sound, regardless of whether you've defined cue points.

When the movie plays, the playhead pauses at the frame until the cue point passes.

Synchronizing sound with Lingo or JavaScript syntax

By writing script that performs an action when a cue point is reached in a sound or QuickTime file, you can synchronize a movie with sound or digital video. For more information about the following methods and properties, see the Scripting Reference topics in the Director Help Panel.

- To set up script that runs when the movie reaches a cue point in a sound or QuickTime file, put the script in an `on cuePassed` handler.
- To determine whether a sound or QuickTime file has passed a specific cue point, use the `isPastCuePoint()` method.
- To find the ordinal number of the last cue point passed in a sound or QuickTime file, use the `mostRecentCuePoint` method.
- To obtain a list of names for the cue points in a specific sound or QuickTime file, test the `cuePointNames` property.
- To obtain a list of times for cue points in a specific sound or QuickTime file, test the `cuePointTimes` property.

Accessibility

With scripts and behaviors, you can provide captioning to help users with hearing impairment experience the audio portions of your movies. For more information, see Chapter 20, “Making Director Movies Accessible,” on page 427.

CHAPTER 11
Using Digital Video

You can give your Macromedia Director MX 2004 movie added appeal by including digital video. Digital video not only offers high-quality real-time image animation and sound, but also supports new types of media such as Windows Media audio and video files and DVD content.

Director supports QuickTime video and Real Media content for Windows and Macintosh. Director also supports Windows Media Video and Audio (WMV and WMA) for Windows only. Audio Video Interleave files (AVI) in Windows are supported through the Windows Media Xtra. The Windows Media Xtra extensions can also support MPEG-1 (including MP3), MPEG-4, WAV, and RIFF.

QuickTime is a multimedia format in its own right. It offers sophisticated sound features and can include graphics in many formats, including basic navigation of QuickTime VR2 files. For a list of supported QuickTime formats, see the Apple Computer website at www.apple.com. To use QuickTime, you must also obtain QuickTime 3 or later (QuickTime 6 or later is recommended) from Apple.

Increasingly, digital media is being provided in the DVD format. The Director DVD editor lets you link to, inspect, manipulate, and access the contents of a DVD. You can link to media on hybrid DVD ROM/Video and regular DVD video discs. However, you cannot export Director files in the DVD format.

Note: DVD support in Director authoring and playback has specific requirements. Please refer to the minimum system requirements at www.macromedia.com/go/sysreqs for more information.

The Director media synchronization features let you synchronize events in a movie to precise cue points embedded in digital video.

Video can make significant demands on a computer's processing power, so you might need to manage video content carefully to make sure it does not adversely affect your movie's performance.

Lingo or JavaScript syntax gives Director more flexibility when playing digital video and can help overcome performance concerns. You can use it to play digital video in ways that are not possible with the Score alone. Using Lingo or JavaScript syntax, you can do the following:

- Precisely synchronize digital video and animation
- Turn digital video on and off on demand and control individual video tracks

- Control QuickTime VR
- Trigger events at key points in time for a video sprite's playback

Note: You can export movies or portions of movies as QuickTime or AVI videos. For more information, see "Exporting digital video and frame-by-frame bitmaps" on page 465.

About digital video formats

When you import Windows Media, DVD content files, AVI, QuickTime, or RealMedia, the cast members you create always remain linked to the original external file, even if you select the Standard Import option. When you distribute a movie, you must always include all digital video files along with the movie.

Windows Media playback is not supported on Macintosh in Director.

Director converts an AVI video to QuickTime when it plays on a Macintosh.

QuickTime must be installed on a computer in order to author or play back a movie that contains QuickTime digital video. RealPlayer 8 or RealOne Player must be installed on a computer to author or play back a movie that contains RealMedia digital video.

For security reasons, Macromedia Shockwave links to media on a local disk only if it is in a folder named dswmedia. To test movies in a browser locally before uploading them to your Internet server, place the movie, linked casts, and linked media in folders within a dswmedia folder, and use relative links to refer to them. In order for them to be accessible from your server, you must use file and folder names that do not have spaces or capital letters and that have recognized file extensions such as .dcr and .gif. For more information, see Chapter 24, "Using Shockwave Player," on page 469.

To import digital video:

1 Select File > Import.

2 Select QuickTime, AVI (Windows only), Windows Media (Windows only), or RealMedia from the Files of Type pop-up menu.

Note: Importing DVD content follows a different process than the other digital video formats. For complete information about importing DVD files, see "Using DVD media content in Director" on page 254.

3 Select the digital video files to import.

Because digital video is always imported as linked, you do not have to select an option in the Media pop-up menu.

4 Click Import.

When you import an AVI file, you are prompted to select QuickTime or AVI as the import format.

If you select QuickTime, Director imports the video as a QuickTime Asset Xtra, which provides additional playback options. For more information, see "Setting QuickTime digital video cast member properties" on page 248.

Using the video windows

Whether a digital video is a cast member or a sprite on the Stage, you can preview it in its corresponding video window. There are different versions of the window for QuickTime, Windows Media, DVD, RealMedia, and AVI digital content.

To open a video window, do one of the following:

1 Click a digital video cast member or sprite and select one of the following menu options:
   - Window > QuickTime
   - Window > DVD
   - Window > RealMedia
   - Window > Windows Media
   - Window > AVI Video

2 Double-click a digital video cast member or sprite and the Video panel appears. The Default panel set contains a tab for each video window.

If you are working with a QuickTime digital video, a video controller bar appears and lets you start, stop, rewind, or forward the movie. Windows Media has play, stop, pause, rewind, and fast forward buttons to perform each task. It also has a time slider that lets you select a time from which the movie should be played. With RealMedia, you can use control buttons to start, stop, and rewind the movie. The DVD viewer window has Root Menu, Title Menu, Pause, Stop, Play, Fast Reverse, and Fast Forward control buttons for previewing and interacting with DVD content.

Playing digital video direct-to-Stage

Director can play digital video using a feature called direct-to-Stage. Direct-to-stage lets video drivers installed on the computer completely control the video playback.

Note: The direct-to-Stage feature cannot be used with DVD or RealMedia digital video because DVD is always direct-to-stage and RealMedia is always non-direct-to-stage.

Direct-to-stage often provides the best performance from a digital video, but it has two disadvantages:

- The digital video always appears in front of all other sprites on the Stage, no matter which channel contains the sprite.
- Ink effects do not work, so it is difficult to conceal the video's bounding rectangle with Background Transparent ink.

When direct-to-Stage is off, Director layers a digital video on the Stage exactly the same as other sprites, and Background Transparent ink works normally. (Matte ink does not work for digital video sprites.)

To set direct-to-Stage options:

1 Select a digital video cast member or sprite.

2 Click the QuickTime or Windows Media tab in the Property inspector.

3 Select or deselect direct-to-Stage (DTS).

4 If the cast member or sprite is a QuickTime video, select one of the following Playback options:

Sync to Soundtrack makes the digital video skip frames (if necessary) to keep up with its soundtrack. The digital video might also take less time to play.

Play Every Frame makes every frame of the digital video appear but does not play the soundtrack because the video cannot play the soundtrack asynchronously while the video portion plays frame by frame. Depending on the data rate of the digital video, the sprite might play more smoothly with this option selected, but this is not a certainty. In addition, playing every frame might cause the digital video to take more time to play.

5 If the sprite or cast member is a QuickTime video, select Controls to display a controller bar below the movie to let the user to start, stop, and step through the movie.

Controlling digital video in the Score

You add a digital video cast member to a score the same way you would add any other sprite. Digital video sprites begin playing when the playhead reaches the frame that contains the video sprite. Use the QuickTime or Windows Media tab in the Property inspector to make QuickTime and Windows Media movies pause or loop. See "Setting QuickTime digital video cast member properties" on page 248. Use the RealMedia tab in the Property inspector to make RealMedia movies pause. For more information, see "The RealMedia tab in the Property inspector" on page 267.

If there's a white bounding rectangle around the video, use the Background Transparent ink to remove it. Inks don't work if direct-to-Stage is turned on (see "Playing digital video direct-to-Stage" on page 245). Matte ink does not work for any type of digital video.

To create a digital video sprite:

1 Do one of the following:

- Drag a digital video cast member to any sprite channel in the Score.
- Place the digital video cast member directly on the Stage.

2 Extend the sprite through as many frames as desired in the Score.

Playing complete digital videos

A digital video, like sound, is a time-based cast member. If you place a video in only a single frame of the Score, the playhead moves to the next frame before Director has time to play more than a brief instant of the video.

To make sure that Director plays an entire digital video, do one of the following:

- Create a tempo setting in the tempo channel using the Wait for Cue Point option in the Frame Properties: Tempo dialog box. For more information about setting tempo, see "Specifying tempo properties" on page 157. This option keeps the playhead from moving to the next frame until a cue point in the video has passed or, if there are no cue points, until the end of the video is reached. For more information, see "Synchronizing video and animation" on page 259.
- Use script or behaviors to make the playhead stay in a frame until the end of the video or until a certain cue point passes. For more information, see "Synchronizing video and animation" on page 259.
- Extend the video through enough frames to give it time to play all the way through.

Playing digital video with Lingo or JavaScript syntax

Lingo or JavaScript syntax can take advantage of the most important and powerful features of digital video. Besides playing digital video linearly, Lingo or JavaScript syntax can pause, stop, and rewind a video. These abilities are useful for jumping to segments within a digital video and for emulating a typical digital video control panel.

Lingo or JavaScript syntax also lets you work with an individual track in a digital video by determining the track's content and position and by turning these tracks on and off.

For information about using Lingo or JavaScript syntax with RealMedia movies, see "Using RealMedia content in Director" on page 259.

Controlling digital video playback with Lingo or JavaScript syntax

The following list provides a general description of how you can control digital video with Lingo or JavaScript syntax. Each media type may have slightly different properties and methods for playback control, but conceptually they work as described below. For more information about particular digital video media types, see the Scripting Reference topics in the Director Help Panel.

- To turn on looping in a digital video cast member, set the digital video's `loop` cast member property to `TRUE`.
- To determine the current time of a digital video sprite, check the sprite's `currentTime` property.
- To pause a digital video sprite, set the sprite's `movieRate` property to 0.
- To start a paused digital video sprite, set the sprite's `movieRate` property to 1.
- To play a digital video sprite in reverse, set the sprite's `movieRate` property to -1.
- To rewind a digital video sprite to the beginning, set the sprite's `movieTime` property to 0.
- To control a digital video sprite's playback rate, set the sprite's `movieRate` property to the desired rate.
- To mix QuickTime audio tracks with internal Director sounds (necessary only in Windows), use the `soundDevice` system property to specify QT3Mix.

Determining digital video content with Lingo or JavaScript syntax

The following list describes how script can determine a digital video's content. For more information, see the Scripting Reference topics in the Director Help Panel.

- To determine the time units a digital video cast member uses, check the video's `timeScale` cast member property.
- To determine whether a digital video is QuickTime or Windows Media, check the digital video's `digitalVideoType` cast member property.
- To determine the number of tracks in a digital video sprite or cast member, check the digital video's `trackCount` sprite or cast member property.
- To determine which type of media a digital video track contains, check the digital video's `trackType` sprite or cast member property.
- To determine the start time of a track in a digital video sprite or cast member, check the digital video's `trackStartTime` sprite or cast member property.

- To determine the stop time of a track in a digital video sprite or cast member, check the digital video's `trackStopTime` sprite or cast member property.
- To determine whether a sprite's track is enabled to play, check the digital video's `trackEnabled` sprite property.
- To obtain the text at the current time from a text track in a digital video sprite, check the digital video's `trackText` sprite property.
- To determine the time of the track before the current time in a digital video, check the digital video's `trackPreviousSampleTime` cast member property and `trackPreviousKeyTime` sprite property.
- To determine the time of the next sample after the current time in a digital video, check the digital video's `trackNextSampleTime` cast member property and `trackNextKeyTime` sprite property.

Turning digital video tracks on and off with Lingo or JavaScript syntax

By turning a digital video's soundtracks on or off, you can play only the animation or control which sounds play.

To control whether individual digital video tracks play:

- Use the `setTrackEnabled()` method. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

Setting QuickTime digital video cast member properties

Use cast member properties to control the media in a QuickTime digital video, specify how it is framed and whether it plays direct-to-Stage, and other important options.

To set QuickTime digital video cast member properties:

1. Select a QuickTime digital video cast member.
2. Click the Member tab in the Property inspector.

   There are several noneditable options on the Member tab in the Property inspector:

   - The cast member size in kilobytes
   - The cast member creation and edit dates
   - The name of the last person who modified the cast member
3. Use the Name text box to view or edit the cast member name.
4. To change the external file to which the cast member is linked, enter a new path and file in the Filename text box. You can also use the Browse button to select a new file.
5. To specify how Director removes the cast member from memory if memory is low, select an option from the Unload pop-up menu. See "Controlling cast member unloading" on page 47.
6. Click the QuickTime tab in the Property inspector to set the remaining properties.

7 To determine how a movie image appears within the sprite bounding rectangle when the movie is rotated, scaled, or offset, set the following Framing options:

Crop displays the movie image at its default size. Any portions that extend beyond the sprite's rectangle are not visible. For more information, see "Cropping digital video" on page 257.

Center is available only if Crop is selected. It determines whether transformations occur with the cast member that is centered within the sprite or with the cast member's upper-left corner aligned with the sprite's upper-left corner.

Scale fits the movie inside the bounding rectangle.

8 To determine how the video plays back, set the following options in the upper portion of the window:

Video displays the video portion of the digital video. If this option is turned off, the video portion does not play. Deselect this option and select Sound if you want to play only the audio portion of a movie.

Audio plays the sound portion of the digital video.

DTS (direct-to-Stage) lets QuickTime drivers installed on the computer completely control the video playback. For more information, see "Playing digital video direct-to-Stage" on page 245.

Controls displays a controller bar at the bottom of the video if direct-to-Stage is selected.

Paused stops the digital video when it first appears on the Stage (while playing the Director movie).

Loop replays the digital video continuously from the beginning to the end.

Streaming begins playing the video while the rest of the video continues to load from its source.

9 If direct-to-Stage is selected, select one of the following options from the Playback pop-up menu to specify how to synchronize the video to its soundtrack:

Sync to Sound makes the digital video skip frames (if necessary) to keep up with its soundtrack. The digital video might also take less time to play.

Play Every Frame (No Sound) makes every frame of the digital video appear but does not play the soundtrack because the video cannot play the soundtrack asynchronously while the video portion plays frame by frame. Depending on the data rate of the digital video, the sprite might play more smoothly with this option selected, but this is not a certainty. In addition, playing every frame might cause the digital video to take more time to play.

10 If Play Every Frame (No Sound) is selected, select the following options from the Rate pop-up menu to set the rate at which a digital video plays:

Normal plays each frame at its normal rate, and no frames are skipped.

Maximum plays the movie as fast as possible while still displaying each frame.

Fixed plays the movie using a specific frame rate. Enter the number of frames per second in the field at the right. Use this option only for digital videos that use the same frame rate for each frame of the movie.

Controlling QuickTime with Lingo or JavaScript syntax

You can use Lingo or JavaScript syntax to control a QuickTime video's appearance and sound volume. For QuickTime VR, you can use Lingo or JavaScript syntax to pan a QuickTime VR digital video and specify what happens when the user clicks or rolls over portions of the video.

You can set the rotation, scale, and translation properties for either a QuickTime cast member or a sprite. For more information, see the Scripting Reference topics in the Director Help Panel.

- To determine whether a cast member or sprite is a QuickTime VR digital video, test the `isVRMovie` property.
- To obtain a floating-point value that identifies which version of QuickTime is installed on the local computer, use the `quickTimeVersion()` method.
- To control a QuickTime sprite's sound volume, set the `volume` sprite property.
- To set the internal loop points for a QuickTime cast member or sprite, set the `loopBounds` sprite property.

Applying masks for QuickTime

Director provides specific script properties for applying masks to QuickTime digital videos. For more information, see the Scripting Reference topics in the Director Help Panel.

- To use a black-and-white cast member as a mask for QuickTime media rendered direct-to-Stage, set the `mask` cast member property.
- To control the way Director interprets a QuickTime video's mask cast member property, set the `invertMask` property.

Responding to user interaction

Lingo or JavaScript syntax lets you control how QuickTime VR responds when the user clicks a QuickTime VR sprite. Use script to specify how Director handles image quality, clicks and rollovers on a QuickTime VR sprite, clicks on hotspots, and interactions with QuickTime VR nodes. For more information, see the Scripting Reference topics in the Director Help Panel.

- To set the codec quality to use when the user drags on a QuickTime VR sprite, set the `motionQuality` sprite property.
- To specify the codec quality to use when a QuickTime VR panorama image is static, set the `staticQuality` sprite property.
- To enable or disable a specific hotspot for a QuickTime VR sprite, use the `enableHotSpot` method.
- To control how Director passes mouse clicks on a QuickTime sprite, set the `mouseLevel` sprite property.
- To find the approximate bounding rectangle for a specific hotspot in a QuickTime VR sprite, use the `getHotSpotRect()` method.
- To specify the name of the handler that runs when the pointer enters a QuickTime VR hotspot that is visible on the Stage, set the `hotSpotEnterCallback` QuickTime VR sprite property.
- To find the ID of the hotspot, if any, at a specific point on the Stage, use the `ptToHotSpotID()` method.
- To specify the name of the handler that runs when the user clicks a hotspot in a QuickTime VR sprite, set the `triggerCallback` sprite property.

- To determine the name of the handler that runs when the pointer leaves a QuickTime VR hotspot that is visible on the Stage, set the `hotSpotExitCallback` property.
- To specify the ID of the current node that a QuickTime VR sprite displays, set the `node` QuickTime VR sprite property.
- To specify the name of the handler that runs after the QuickTime VR sprite switches to a new active node on the Stage, set the `nodeEnterCallback` QuickTime VR sprite property.
- To specify the name of the handler that runs when a QuickTime VR sprite is about to switch to a new active node on the Stage, set the `nodeExitCallback` QuickTime VR sprite property.
- To determine the type of node that is currently on the Stage, test the `nodeType` QuickTime VR sprite property.

Rotating and scaling QuickTime video

Lingo or JavaScript syntax can rotate and scale QuickTime videos, as described in the following list. For more information, see the Scripting Reference topics in the Director Help Panel.

- To control the rotation of a QuickTime sprite, set the `rotation` QuickTime sprite property.
- To control the scaling of a QuickTime sprite, set the `scale` QuickTime sprite property.

Panning QuickTime VR

Use Lingo or JavaScript syntax to pan a QuickTime VR digital video without the user dragging the image, as described in the following list. For more information, see the Scripting Reference topics in the Director Help Panel.

- To set the current pan of the QuickTime VR sprite, set the `pan` QuickTime VR sprite property.
- To nudge a QuickTime VR sprite in a specific direction, use the `nudge` method.

Displaying QuickTime video

Lingo or JavaScript syntax can control how a movie displays QuickTime videos, as described in the following list. For more information, see the Scripting Reference topics in the Director Help Panel.

- To specify the type of warping performed on the panorama of a QuickTime VR sprite, set the `warpMode` QuickTime VR sprite property.
- To specify a QuickTime VR sprite's current field of view, set the `fieldOfView` QuickTime VR sprite property.
- To swing a QuickTime VR sprite to a specific pan, tilt, or field of view, set the `swing` method.

QuickTime VR

You can use a QuickTime VR movie in a Director movie by inserting it as you would any other QuickTime cast member. To get the best performance, turn on direct-to-Stage (see "Playing digital video direct-to-Stage" on page 245).

Using Windows Media files in Director

Director now fully supports Windows Media Video and Audio formats. Windows Media format is a high-quality, secure, and comprehensive digital media format available for streaming and download-and-play applications on Windows systems, set-top boxes, and portable devices. Windows Media format comprises Windows Media Audio and Video codecs, an optional integrated digital rights management (DRM) system, and a file container.

Note: Windows Media is supported on Windows 98, Windows 2000, and Windows XP with DirectX 8.0 and above. Windows Media Xtra extensions are part of the standard Shockwave installer. Shockwave content containing Windows Media content can be viewed in Microsoft Internet Explorer 4.01 or later and in Netscape 6.1 or later. Windows Media Video and Audio is not supported in Director on the Macintosh.

Importing Windows Media

You can import Windows Media content into Director as a cast member. Windows Media cast members can be placed as sprites in movies, like other cast members.

To import Windows Media content as a cast member:

1 Select File > Import.

The Import File dialog box appears.

2 Select Windows Media from the Files of Type pop-up menu.

Both AVI and Windows Media files appear listed.

3 Select the digital video files you want to import.

Because digital video is always imported as linked, you do not have to select an option in the Media pop-up menu.

4 Click Import.

5 A Windows Media cast member appears in the Cast window.

When you import an AVI file (as opposed to a WMV or WMA file), you are prompted to select QuickTime or Windows Media as the import format.

Note: Importing a media file might fail if the appropriate decoders are not present in the system. Installing DirectX 8.0 or later ensures that you have the right decoders.

Inserting Windows Media

You can also insert Windows Media content into your Director movie. The Insert Windows Media Element opens the Windows Media Editor and creates an empty Windows Media cast member. By contrast, importing Windows Media requires the selection of a supported file and does not open the Windows Media Editor.

To insert Windows Media content as a cast member:

1 Select Insert > Media Element > Windows Media.

A Windows Media icon appears as a cast member in the Cast window, and the Windows Media window opens.

2 Select the Windows Media cast member and select Window > Property inspector.

3 Click the Member tab in the Property inspector.

4 In the Filename text box, choose the file to be associated with the cast member. You can type the name of the file, or browse for the file by clicking the "…" button.

5 In the Name text box, type a name for the new Windows Media cast member.

Setting Windows Media properties

You can set Windows Media properties that determine the cast member name, how it is displayed in your movie, whether audio or video in the sprites created from the Windows Media cast member are on or off, and more.

To set Windows Media properties:

1 Select a Windows Media digital video cast member in the Cast window.

2 Click the Member tab in the Property inspector.

3 There are several noneditable options on the Member tab in the Property inspector:

- The cast member size in kilobytes
- The cast member creation and edit dates
- The name of the last person who modified the cast member

4 Use the Name text box to view or edit the cast member name.

5 To change the external file to which the cast member is linked, enter a new path and file in the Filename text box. You can also use the Browse button to select a new file.

6 To specify how Director removes the cast member from memory if memory is low, select an option from the Unload pop-up menu. For more information, see "Controlling cast member unloading" on page 47.

7 Click the Windows Media tab in the Property inspector to set the remaining properties:

Video turns on or off the video image portion of the Windows Media cast member. If this option is turned off, the video portion does not play; the audio portion, however, remains unaffected. Deselect this option and select Audio if you want to play only the audio portion of the Windows Media cast member.

Audio turns on or off the audio portion of the Windows Media member. If this option is turned off, the audio portion does not play; the video portion, however, remains unaffected. Deselect this option and select Video if you want to play only the video portion of the Windows Media cast member.

Loop if checked repeatedly plays the Windows Media cast member from beginning to end.

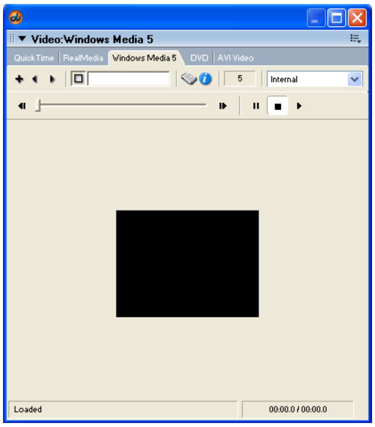
DTS (direct-to-Stage) allows drivers installed on the computer to completely control playback of the video portion of the Windows Media cast member. For more information, see "Playing digital video direct-to-Stage" on page 245.

Paused (paused at start) if checked the Windows Media cast member is paused on the first frame of the video. The audio portion of the cast member is also paused.

Note: A playback rate other than 1.0 might fail if DirectX filters do not support it. Duration and Dimension fields are not editable.

Using the Windows Media video window

The Windows Media window lets you preview Windows Media audio and video files. You can also control Windows Media sprites using the this window. For more information, see "Controlling Windows Media sprites using Lingo or JavaScript syntax" on page 254.



To use the Windows Media window:

- Double-click a Windows Media cast member or sprite.

  The Windows Media window appears. This window provides controls to play, stop, pause, rewind, and fast forward the movie. Using the slider, you can select a position from which to start play.

Controlling Windows Media sprites using Lingo or JavaScript syntax

You can use Lingo or JavaScript syntax to control Windows Media sprites. The script handlers support the following actions: play, stop, pause, rewind, and play from one specific time point to another. The Windows Media sprite displays the attributes, movie duration, elapsed playtime, height, width, and playback state for Lingo or JavaScript syntax scripting. For more information, see the Scripting Reference topics in the Director Help Panel.

Using DVD media content in Director

You can link DVD media content to a Director movie and use the DVD media editor to inspect that linked content. However, you can't actually make changes to DVD content within Director. Rather, you can change the attributes of the DVD media by using either the Property Inspector prior to playback or via Director's scripting capabilities while the movie's playing by polling events and modifying properties and methods that affect the linked DVD content.

When linking to DVD media content, that content can reside either on a DVD disc located in the DVD drive or on a local hard disk in a DVD Volume Folder. In either case, a DVD drive and DVD player and decoder needs to be installed and functional in order for DVD support to be functional in Director.

Linking to DVD media content

You can link to DVD media content by using the Insert menu or the DVD Editor, or you can use scripting to set the folder property value. Setting the folder property will allow Director to play DVDs from the hard drive or from the relative location of a movie/projector.

When you create a DVD cast member on a Windows computer using Director's user interface, Director first looks for DVD media in the form of a video_ts folder in the first DVD disk drive that contains a valid DVD disc. If no disk is found, Director then searches for a video_ts folder at the root of the hard drive. On a Macintosh computer, a video_ts folder will only be detected automatically if it resides on a DVD disc in the DVD drive.

Note: If you are using any of the following methods to link DVD media content to your movie, your system must have the required drivers and decoders necessary to play DVD.

To link DVD media content to a movie by using the Insert menu:

1 Select Insert > Media Element > DVD.

To link DVD media content to a movie by using the DVD Editor:

1 Select Window > DVD.

The DVD Editor appears.

2 Enter a name for the DVD member in the Cast Member Name text box of the DVD Editor. Director creates a new DVD cast member in the first available Cast slot.

Director will attempt to automatically link to available DVD media using the DVD resources that Director relies on to support DVD. For Windows, those resources are DirectX/DirectShow, and third-party elements such as DVD decoders. Some of these decoders, which get installed with DVD players, include WinDVD, PowerDVD, and ATI DVD. For Mactintosh, Director relies on Apple's DVD Framework

To set the property value by using a script:

Use the following examples as a guide for setting the folder property through scripting. These statements set the pathname of the DVD folder property.

Windows example

```
--Lingo syntax
member (2).folder = "C:\myLocalDVDContent\video_ts"
// JavaScript syntax
member (2).folder = "C:\\myLocalDVDContent\\video_ts";
```

Mactinosh example

```
--Lingo syntax
member (2).folder = "/Volumes/Macintosh HD/myLocalDVDContent"
// JavaScript syntax
member (2).folder = "/Volumes/Macintosh HD/myLocalDVDContent";
```

Note: When creating a DVD cast member, if a video_ts folder cannot be found when the DVD cast member is created, an error alert will appear that says, "Unable to locate DVD volume." This alert will appear only once per session. At that point, you can still name the new DVD member and then set its folder property to a folder location that contains a valid video_ts folder.

Note: Two DVDs cannot be played on your computer at the same time. Make sure you are not playing a DVD in Director at the same time as you are trying to play a DVD in a projector, as would be the case if the Preview after publishing option is selected in Publish Settings when publishing a projector.

Issues with Macintosh DVD folder pathnames

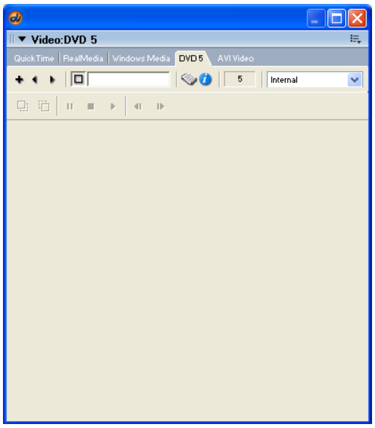
On Macintosh computers, the format of the pathname for the folder property should use a forward slash (/) as the path's delimeter, instead of the standard Macintosh delimiter colon (:). In addition, **/volumes/** should be concatenated at the start of the pathname of the DVD folder. For example, if the DVD folder is located on the root of the boot drive, it would look like the following:

```
member (2).folder = "/Volumes/Macintosh HD/Test_DVD/video_ts"
```

When the `_movie.path` command is used for retrieving the path of the projector or movie on a Macintosh, it will contain a colon (:) instead of the forward slash (/). The use of the colon in the DVD folder's pathname will cause an error. As a workaround, developers can use a script to replace the colon characters in the pathname with forward slashes.

Using the DVD video window

You can access and play DVDs using the DVD window. You cannot, however, edit the files in the real sense of the word; rather, you can preview them in the DVD window.



Setting DVD Cast member properties

The DVD Property inspector lets you set the options for audio, close captioning, volume and whether or not the cast member pauses during playing.

To set DVD cast member properties:

1 Select a DVD cast member on the stage or in the cast window.

2 Select Window > Property Inspector and click on the DVD tab.

3 In Graphical view, set the following properties: Audio, Closed Captions, Paused, and Volume.

4 In List view, set the following properties:

- angle
- audio
- audiotrack
- closedCaptions
- folder
- pausedAtStart
- subPicture
- volume

Cropping digital video

Cropping a digital video means trimming the edges off the top or sides of the movie image. Cropping doesn't permanently remove the portions you crop, but it hides them.

To crop a digital video:

1 Select the cast member in the Cast window.

2 Click the QuickTime tab in the Property inspector.

Note: Cropping is not allowed with RealMedia movies.

3 Select Crop.

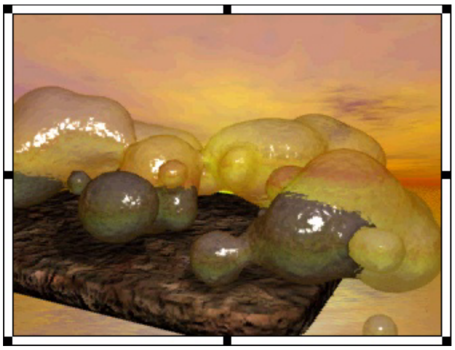
Director retains the movie's original size if you resize the bounding rectangle; however, the edges of the movie are clipped if you make the bounding rectangle too small.

4 Select Center, if you wish.

Director centers the movie when you resize the bounding rectangle. If Center is not selected, the movie maintains its original position when you resize its bounding rectangle. Center is available only if Crop is selected.

5 Select the video in the Score.

6 Go to the Stage and drag any of the handles that appear on the selection rectangle that surrounds the video image.



Director displays only as much of the movie image as can fit in the area that is defined by the selection rectangle.

If you would rather scale the movie than resize it, select Scale instead of Crop on the QuickTime tab in the Property inspector. Director scales the movie if you resize the bounding rectangle.

To use Lingo or JavaScript syntax to move the image of a QuickTime video around within the sprite's bounding rectangle:

- Set the digital video's `translation` QuickTime sprite or cast member property. For more information, see the Scripting Reference topics in the Director Help Panel.

About using digital video on the Internet

In both stand-alone projectors and movies playing in web browsers, Director can handle digital video the same way it handles all other media, or it can stream the digital video using QuickTime 4 or later. You can link the digital video to a URL, and the movie begins to download and play the digital video when its sprite first appears on the Stage.

In order for the digital video cast member to stream, you must set its `streaming` property to `TRUE`. QuickTime 4 or later must be installed to enable streaming.

If a streaming QuickTime file contains cue points you want to use, you must set the text track to be preloaded (use a QuickTime editor such as MoviePlayerPro to do this). If you do not preload the text track, Director disables the cue points so it can stream the file without entirely downloading it first.

You can also import a Real Time Streaming Protocol (RTSP) stream as a QuickTime cast member. The rtsp:// URL must end with the filename extension .mov so that Director knows it should be treated as a QuickTime stream.

When you use streaming digital video in a movie that is distributed on the Internet, remember the following points:

- The video begins to play immediately unless the member's `pausedAtStart` property is set to `TRUE` or the `controller` member property is set to `TRUE`.
- After a digital video begins to download, the download continues until it is finished, even if the sprite no longer appears on the Stage. Use the `percentStreamed` QuickTime sprite property to test how much of the media has been downloaded.The feature works with QuickTime videos only. For more information about this property, see the Scripting Reference topics in the Director Help Panel.

Synchronizing video and animation

To pause the playhead until a specified cue point in digital video is reached, you can use the Wait for Cue Point option in the Tempo dialog box. You can also use this function to wait for the end of the digital video, even if it has no cue points. Cue points can also be used to trigger events that are interpreted by Lingo or JavaScript syntax.

The techniques for synchronizing digital video and animation are the same as those for synchronizing sound and animation. For more information, see "Synchronizing media" on page 240.

Using RealMedia content in Director

The Macromedia Director Xtra for RealSystem Streaming Media (Xtra for RealMedia) adds RealAudio and RealVideo to the media types supported by Director and handles the playback of RealMedia content in the Shockwave Player using an embedded RealPlayer engine. Support for the RealAudio and RealVideo media types allows Director developers to add streaming RealMedia content to Shockwave content and manipulate it using the standard controls available in Director. This content can be viewed by users who have the Shockwave Player and RealPlayer 8 or RealOne Player installed on their system.

RealNetworks RealAudio and RealVideo formats are recognized as the standard for streaming media content on the web today. The ability to add RealMedia content to Shockwave content allows Director developers to take advantage of the millions of RealPlayer programs currently in use, and because the Xtra for RealMedia provides an automatic detection and installation prompt feature for RealPlayer 8 and RealOne Player, there is no risk of locking out any of the millions of users who have already installed the Shockwave Player. In addition, the Xtra for RealMedia is automatically downloaded and installed the first time a user attempts to view Shockwave content containing RealMedia content.

This document contains instructions for using existing RealMedia streams in Director and assumes a basic familiarity with Director, including using inspectors and behaviors.

You cannot create or edit RealMedia content in Director. RealMedia streaming content is created with RealNetworks production tools such as RealProducer Plus and RealProducer Basic. For information about creating content with these tools, see the RealNetworks website at www.realnetworks.com.

System requirements

To create Shockwave movies containing RealMedia content, the following must be installed:

- Director MX, which includes the Xtra for RealMedia.
- RealPlayer 8 or RealOne Player. (RealNetworks products are available for download at www.real.com.)

To view Shockwave content containing RealMedia content, the following software must be installed:

- The Shockwave Player.
- RealPlayer 8 or RealOne Player. If a user without RealPlayer 8 or RealOne Player attempts to play Shockwave content containing a RealMedia cast member, a dialog box asks whether the user wants to go to the RealNetworks website and download RealPlayer 8 or RealOne Player.
- The Xtra for RealMedia (listed in the Movie Xtras dialog box as RealMedia Asset.x32 on Windows) is not part of the standard Shockwave 8.5 installation but is available as an Xtra download package from the Macromedia website (www.macromedia.com). The first time a user attempts to view Shockwave content containing RealMedia content, the Shockwave Player automatically downloads and installs this Xtra. (For more information, see "Publishing Shockwave content with RealMedia" on page 271.)

In addition, viewers of your Shockwave content must have one of the following operating system/browser setups:

- Microsoft Windows 98/2000/XP or later with Microsoft Internet Explorer 5.01 with Service Pack 2 or later or Netscape 7.1 or later.

License restrictions and copyright information

The Macromedia Director license agreement outlines the restrictions and requirements for creating and serving content created using Macromedia Director Xtra for RealSystem Streaming Media (referred to as the Xtra for RealMedia in this document and RealMedia Asset.x32 (Windows) in the Movie Xtras dialog box). Please read the license agreement carefully before creating Shockwave content using RealMedia content.

Macromedia, Director, Lingo, Shockwave, and Xtra are trademarks of Macromedia, Inc. and may be registered in the United States or in other jurisdictions including internationally. Other product names, logos, designs, titles, words, or phrases mentioned within this publication may be trademarks, servicemarks, or tradenames of Macromedia, Inc. or other entities and may be registered in certain jurisdictions including internationally.

RealAudio, RealMedia, RealNetworks, RealPix, RealPlayer, RealOne Player, RealProducer, RealProducer Plus, RealSystem, RealText, and RealVideo are trademarks or registered trademarks of RealNetworks, Inc.

This publication contains links to third-party websites that are not under the control of Macromedia, and Macromedia is not responsible for the content on any linked site. If you access a third-party website mentioned in this guide, then you do so at your own risk. Macromedia provides these links only as a convenience, and the inclusion of the link does not imply that Macromedia endorses or accepts any responsibility for the content on those third-party sites.

RealMedia sample file

If you have RealOne Player or RealPlayer 8 installed on your system and are familiar with Director, viewing and using RealMedia files in Director is incredibly quick and easy. Before you begin reading this document, please view the RealMedia sample file included on the Director MX CD or the videotest.rm file included in the RealPlayer program installation folder.

To view the sample file:

1 Start Director.

2 Select File > Import.

3 Browse to one of these test files:

- If you have the Director MX CD, select a file in the Macromedia\Support\RealMedia folder.
- If you don't have the Director MX CD, select the videotest.rm file in the RealPlayer 8 or RealOne Player installation folder.

4 Click Import.

The file appears in the Cast window.

5 Drag the file to the Stage and select Control > Play.

After you view the sample file, you are ready to start laying out the other elements of your movie.

About RealMedia streams in Director

Director includes the following RealMedia support:

- The RealMedia tab in the Property inspector
- RealMedia behaviors
- Script elements for RealMedia, including methods and properties
- "The RealMedia viewer" on page 270

Director supports RealAudio and RealVideo stream formats, but does not support other formats that RealPlayer supports, such as SMIL (Synchronized Multimedia Integration Language), RealPix, or RealText. Although some of these formats may work with the Xtra for RealMedia, there might be significant problems with playback.

RealMedia cast members

RealMedia cast members are always linked cast members; they reference an external stream via a URL (of type HTTP, RTSP, or PNM) to a location on the Internet or a local file on your hard disk or network file server. RealMedia cast members are always of type `#realMedia`.

Sprites created from RealMedia cast members are treated as regular sprites and can be rotated, skewed, stretched, flipped, colorized, composited with other sprite layers with ink, and manipulated with the new Lingo or JavaScript syntax elements for RealMedia in addition to standard sprite and cast member script.

All RealMedia properties and methods invoked on a RealMedia sprite invoke the corresponding cast member's properties and methods. This is because Director plays back RealMedia files at the cast member level rather than the sprite level. For more information, see "RealMedia stream playback" on page 263.

You can have as many RealMedia streams and cast members in your Director movie as you want, as long as you play them consecutively, not concurrently. This version of Director does not support playing multiple RealMedia cast members at the same time.

RealMedia video

The RealVideo portion of your RealMedia file is rendered offscreen, and there is no direct-to-Stage option. This means that you can layer other sprites on top of your RealMedia sprite. For example, you might do this if you wanted to display text across the RealVideo as it plays. You can use new Lingo or JavaScript syntax elements to jump forward or backward in the stream, or grab the current frame of the RealVideo stream and use it as a texture for a 3D object.

RealVideo is fully integrated into the graphics capabilities of Director, and you add RealMedia cast members containing RealVideo to a movie just as you would any other cast member. The RealVideo content begins playing when the playhead reaches the frame containing the RealMedia content, unless the `pausedAtStart` property of the sprite or cast member is set to `TRUE`.

RealMedia audio

You can have the RealAudio portion of your RealMedia cast member processed in one of two ways: by Director (the default) or by RealPlayer. Your choice depends mainly on whether you want to use Lingo or JavaScript syntax sound elements.

- If you use Director to process RealAudio in your movie, you can use Lingo or JavaScript syntax sound methods and properties to control and manipulate RealAudio, including mixing RealAudio with other Director audio. However, note that all RealAudio is played in a single sound channel. If you inadvertently overlap RealMedia cast members in the score, and the second RealMedia cast member begins to play before the first cast member is finished, the second cast member's sound is played in the same sound channel as the first cast member, even if you assigned a different sound channel to the second cast member. If the RealMedia cast members do not overlap, they are played in the sound channel specified. (If you do not assign a sound channel to a RealMedia cast member, the RealAudio is played in the highest available sound channel.)

  For more information about working with Director audio, see the Scripting Reference topics in the Director Help Panel. For information about using standard script sound elements with RealMedia content, see "Using Lingo or JavaScript syntax sound elements with RealMedia" on page 272.

- If you use RealPlayer native audio to process RealAudio, all script sound elements and the audio property in the Property inspector are ignored. You enable RealPlayer native audio by setting the `realPlayerNativeAudio()` method to `TRUE`. This method should be executed in a `prepareMovie` event handler in a movie script. This is a system-level method that you can only set using Lingo or JavaScript syntax, so you must set it before the first RealMedia cast member is encountered in the Score, which causes Director to load the RealPlayer engine. Once the RealPlayer is loaded, changes to this method are ignored.

  For more information about this method, see the Scripting Reference topics in the Director Help Panel.

RealMedia stream playback

In Director, RealMedia playback occurs at the cast member level, not the sprite level. Therefore, if you have two sprites of a cast member, and you apply a method or property to one of the sprites, the method applies to both sprites. For example, if you select the Display Real Logo check box (or call the corresponding `displayRealLogo` script property in a script) for one of the sprites, the Display Real Logo check box is automatically selected for both of the sprites, and the RealNetworks logo is displayed when either sprite plays. This is true for all methods and properties, not just the `displayRealLogo` property.

If your movie contains more than one RealMedia sprite (not playing at the same time) that reference the same cast member, you may want to create two cast members referencing the same URL, so that the sprites can be controlled independently. Sprites that reference the same cast member are subject to the methods and properties of the member.

Streaming

Streaming is the most efficient and user-friendly method of downloading, viewing, and listening to video and audio content on the Internet. Users can begin viewing content as soon as a small portion (usually a few seconds) of the file has downloaded. As the stream plays, the rest of the stream continues to download in the background.

If you understand how the streaming process works in Director, and for RealMedia cast members in particular, you can minimize the amount of time users must wait before your content begins to play in the browser.

The Shockwave Player first downloads the Score information, scripts, and information about the size and shape of each cast member, and then downloads the media in the cast members as they are played in the movie. When a RealMedia cast member begins to play, the streaming process begins and cycles through the various states. If you are viewing content in the RealMedia viewer, the `mediaStatus` property value that corresponds to the state in the streaming process is displayed in the status bar. For more information about this property, see the Scripting Reference topics in the Director Help Panel.

During the seeking or buffering state (`mediaStatus #buffering`), RealMedia cast members are downloading into a buffer that holds the portion of the stream that is about to play. This initial loading of the buffer is what causes the delay between the call to the `play` method (in the Score or user initiated) and the actual playing of the stream. Once the stream begins to play, the information in the buffer is continually updated with the next portion of the stream to be played, and the stream plays without interruption. For more information, see `percentBuffered` in the Scripting Reference topics in the Director Help Panel.

When using Lingo or JavaScript syntax with RealMedia cast members, you need to know which streaming state the cast member has reached, or script errors can result. For complete information about the order of the states in the streaming process and the impact on RealMedia cast members, see `state (RealMedia)` and `mediaStatus` in the Scripting Reference topics in the Director Help Panel.

Streaming of RealMedia cast members is handled by RealPlayer, not the Shockwave Player. Since the streaming of RealMedia cast members is independent of the streaming functionality of Director, RealMedia cast members are not placed in the Director cache, and NetLingo does not apply to RealMedia cast members.

Authoring tips

Review the following guidelines before you begin creating RealMedia cast members and assembling your movie.

- Whenever possible, refer to sprites rather than cast members in Lingo or JavaScript syntax. Because future versions of Director may enable sprite-level playback of RealMedia cast members, referring to sprites may help ensure that your movies are forward-compatible with Director.
- After you create a RealMedia cast member, play the movie once to obtain and save the `duration`, `height`, and `width` properties of the RealMedia cast member, and then lay out the rest of your movie. These properties are not known until the cast member is played, and the values initially displayed in the Property inspector are placeholders.
- When using RealMedia cast members, it is a good idea to loop the playing of the cast member or sprite in a limited number of frames in the movie. The reason for this is that the Score is frame-based not time-based, which makes it difficult to determine the frame span for the sprite or cast member in the Score. This is true for sprites and cast members of all media types, but especially for streaming RealMedia sprites and cast members that are subject to network congestion and rebuffering.
- RealPlayer is not designed to play concurrent streams, and since RealAudio and RealVideo files in Director content are played by an embedded RealPlayer engine, playing more than one RealMedia cast member at the same time is not supported.
- If the RealMedia cast member in your movie references a local file instead of a remote URL, be sure that the file path you specified in the Property inspector is relative to the final document, or that you update the file path with the new location of the file before publishing your movie.

  All RealMedia content must live on a server authorized to serve RealMedia streaming content. For more information about serving RealMedia content, see the RealProducer Help installed with the RealProducer program. Then visit the developer pages of the RealNetworks website (www.realnetworks.com/devzone).

Creating RealMedia cast members

As with other media types, there are three ways to create a RealMedia cast member: insert the RealMedia content using Insert > Media element, import the remote or local RealMedia file using File > Import, or use the New Cast Member (+) button in the RealMedia viewer. The instructions for importing remote and local files differ slightly.

When you initially create a RealMedia cast member, the values listed for the height, width, rect, and duration properties in the Property inspector are placeholder values. The actual values of these properties remain unknown until the cast member is played and saved for the first time. For more information, see "Obtaining dynamic RealMedia cast member properties" on page 266.

Before following these instructions, make sure the Property inspector is open (Window > Property Inspector).

To create a RealMedia cast member using Insert › Media Element:

1 Select Insert > Media Element > RealMedia.

2 On the Member tab in the Property inspector, enter the name of the RealMedia cast member and enter the URL, or use the "..." button to browse to the location of a local RealMedia file.

3 Use the options on the RealMedia tab in the Property inspector to specify the properties of the cast member.

   For more information, see "The RealMedia tab in the Property inspector" on page 267.

To create a RealMedia cast member from a remote file using File › Import:

1 Select File > Import or press Control+R to open the Import File dialog box.

2 Click the Internet button.

3 In the dialog box that appears, enter the URL where your RealMedia file is located and click OK.

4 Click Import.

   The RealMedia cast member is now listed in the Cast window with a RealMedia icon. The name of the cast member is automatically entered in the Property inspector's Name text box, and the URL of the file is automatically entered in the Name text box on the Property inspector Member tab.

5 Specify the properties of the cast member using the Property inspector RealMedia tab. For more information, see "The RealMedia tab in the Property inspector" on page 267.

To create a RealMedia cast member from a local file using File › Import:

1 Select File > Import or press Control+R to open the Import File dialog box.

2 Navigate to the RealMedia file you want to import.

3 Click Import.

   The RealMedia cast member is now listed in the Cast window with a RealMedia icon. The name of the cast member is automatically entered in the Property inspector's Name text box, and the URL of the file is automatically entered in the Name text box on the Property inspector Member tab.

4 Specify the properties of the cast member using the Property inspector RealMedia tab. For more information, see "The RealMedia tab in the Property inspector" on page 267.

To create a RealMedia cast member using the New Cast Member (+) button in the RealMedia viewer:

1 Select Window > RealMedia to open the RealMedia viewer.

2 Click the New Cast Member (+) button on the RealMedia viewer to create a new cast member.

3 On the Member tab in the Property inspector, enter a name for the RealMedia cast member, and enter the URL or use the "..." button to browse to the location of a local RealMedia file.

4 Use the options on the Property inspector RealMedia tab to specify the properties of the cast member. For more information, see "The RealMedia tab in the Property inspector" on page 267.

Obtaining dynamic RealMedia cast member properties

When a RealMedia cast member is initially created, the values that are listed in the Property inspector for the dynamic properties—`height`, `width`, `rect`, and `duration`—are placeholder values. After you play the cast member on the Stage or in the RealMedia viewer, the actual values for the properties are saved and appear in the Property inspector. When you save the movie, these values are saved with the cast member.

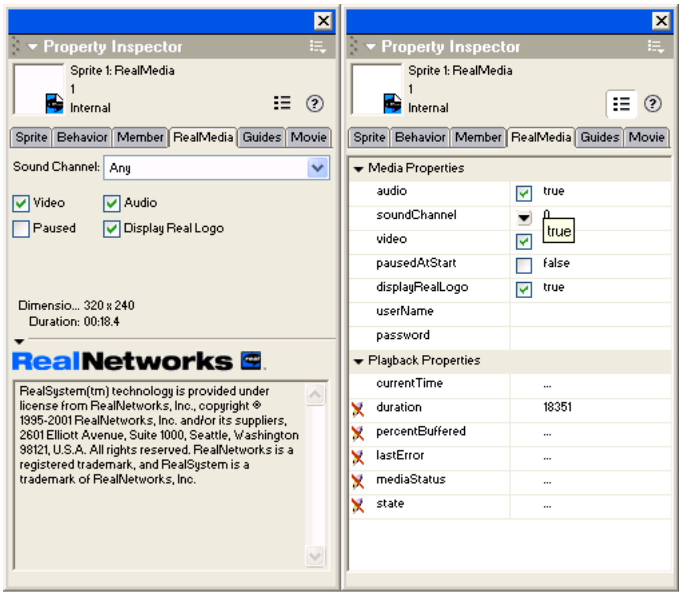
It's a good idea to obtain and save the actual values of the dynamic RealMedia properties before you begin laying out your movie. Although you cannot set or change any of these properties, you can adjust the height and width of RealMedia sprites on the Stage to fit your movie's proportions. It is important to remember that the actual playback time of a stream can vary, depending on the level of network congestion and stream rebuffering and that the `duration` value of a cast member referencing a live feed is always 0.

To play RealMedia cast members:

- To play the RealMedia cast member with other elements of your Director movie, drag the RealMedia cast member to the Stage or Score and select Control > Play. The RealMedia cast member starts to play after the playhead reaches the frame that contains the cast member, unless the `pausedAtStart` property for the cast member is set to `TRUE`.
- To play the cast member without the other elements of your movie, double-click the cast member in the Cast window and click the Play button in the RealMedia viewer.

The RealMedia tab in the Property inspector

The RealMedia tab in the Property inspector displays the properties of RealMedia cast members. To set or change the editable properties, you can use the Property inspector or the script properties for RealMedia. Even if you are not planning to use the script properties, it is a good idea to read the script entries for the properties that appear in the Property inspector because they contain valuable information. For more information, see "Using Lingo or JavaScript syntax sound elements with RealMedia" on page 272.



Sound Channel: Any

✓ Video ✓ Audio

☐ Paused ✓ Display Real Logo

Dimensio... 320 x 240
Duration: 00:18.4

RealSystem(tm) technology is provided under license from RealNetworks, Inc., copyright © 1995-2001 RealNetworks, Inc. and/or its suppliers, 2601 Elliott Avenue, Suite 1000, Seattle, Washington 98121, U.S.A. All rights reserved. RealNetworks is a registered trademark, and RealSystem is a trademark of RealNetworks, Inc.

Media Properties	
audio	✓ true
soundChannel	0
video	✓ true
pausedAtStart	☐ false
displayRealLogo	✓ true
userName	
password	
Playback Properties	
currentTime	...
duration	18351
percentBuffered	...
lastError	...
mediaStatus	...
state	...

Two views of the RealMedia tab in the Property inspector

You can work with the following media properties:

- `audio (RealMedia)` specifies whether the audio portion of the RealMedia stream plays (`TRUE`) or not (`FALSE`). The default setting is `TRUE`. This property has no effect if `realPlayerNativeAudio()` is enabled.
- `soundChannel (RealMedia)` specifies the Director sound channel where the RealAudio plays. The default setting is Any (0), which means the audio plays in the highest available channel. This property has no effect if you enable `realPlayerNativeAudio()`.
- `video (RealMedia)` specifies whether the video portion of the RealMedia stream appears (`TRUE`) or not (`FALSE`). The default setting is `TRUE`.
- `pausedAtStart (RealMedia)` specifies whether the RealMedia stream begins to play automatically when the playhead enters the frame span of the RealMedia cast member or sprite (`FALSE`) or not (`TRUE`). The default setting is `FALSE`.
- `displayRealLogo` specifies whether the RealNetworks logo appears. When this property is set to `TRUE`, the RealNetworks logo appears at the beginning of the stream and when the video is stopped or rewound.

- `userName (RealMedia)` lets you specify a user name if the cast member references a protected URL. For security purposes, after a user name has been entered, it cannot be retrieved. If this property has been set, the value that appears in the Property inspector is ********.
- `password` allows you to specify a password if the cast member references a protected URL. For security purposes, after a password has been entered, it cannot be retrieved. If this property has been set, the value that appears in the Property inspector is ********.

  For more information about these properties, see the Scripting Reference topics in the Director Help Panel.

You can work with the following playback properties:

- `currentTime (RealMedia)` displays the current time in the RealMedia stream in milliseconds. Setting this property in the Property inspector is the same as using the `seek` method in script.
- `duration (RealMedia)` displays the length of the RealMedia stream in milliseconds. This property is not known until the movie has been played once and saved. The duration of a live stream is always 0. This property cannot be set.
- `percentBuffered` displays the percentage of the playback buffer that has been filled with the RealMedia stream. This property cannot be set.
- `lastError` displays the last error returned by RealPlayer. This property cannot be set.
- `mediaStatus` displays the current status of the RealMedia stream. For a list of possible values, see the Scripting Reference topics in the Director Help Panel. This property cannot be set.
- `state` displays which state the cast member is currently in the streaming process. For a list of possible values, see the Scripting Reference topics in the Director Help Panel. This property cannot be set.

  For more information about all of these properties, see the Scripting Reference topics in the Director Help Panel.

About RealMedia behaviors

The RealMedia behaviors are designed to let you easily add playback controls for RealMedia streams in your movie using custom graphics.

The following RealMedia behaviors are listed in the Media > RealMedia section of the Library palette:

RealMedia Target identifies a RealMedia sprite as the target for RealMedia behaviors that are attached to the graphic, text, or field sprites for the playback controls. You must attach this behavior to a RealMedia sprite on the Stage before you can use any of the other RealMedia behaviors. This behavior does not control the RealMedia sprite by itself, but works with the other RealMedia behaviors to control the sprite.

RealMedia Control Button lets a graphic sprite function as a control button for the RealMedia sprite with the RealMedia Target behavior attached. Possible control button behaviors are Pause, Play, Stop, Small Forward, Small Backward, Large Forward, Large Backward, Toggle Audio, Audio On, Audio Off, Toggle Video, Video On, and Video Off.

RealMedia Slider Bar lets a graphic sprite define the horizontal limits of travel for the RealMedia Slider Knob behavior, which must be used with this behavior. The RealMedia Slider Bar behavior requires that a RealMedia sprite (with the RealMedia Target behavior attached) be on the Stage.

RealMedia Slider Knob lets a graphic sprite function as a slider to control and monitor the playback location (current time) of the RealMedia sprite with the RealMedia Target behavior attached. When the user drags a sprite that has this behavior attached, a `seek` action is performed on the stream.

RealMedia Buffering Indicator lets a text area or field provide a graphical display of the stream-buffering progress of the RealMedia sprite with the RealMedia Target behavior attached. As stream buffering progresses, the width of the sprite increases from 0 to 100%.

RealMedia Stream Information lets a text area or field display text information for the RealMedia sprite with the RealMedia Target behavior attached. The text information can include one of the following: the percent buffered, media status, current time, or file location or URL of the RealMedia file. You select the information to display using the pop-up menu in the Parameters dialog box for this behavior.

Using RealMedia behaviors

You attach RealMedia behaviors in the same way as you do other Director behaviors: drop the behavior onto the sprite and use the dialog box to assign a group and other parameters.

The RealMedia Target behavior is the central RealMedia behavior and must be dropped onto the RealMedia sprite before you can use any other behavior.

The RealMedia Slider Knob and RealMedia Slider Bar behaviors must be used together; if they can't locate one another, a one-time error message appears.

To attach RealMedia behaviors:

1. Create a RealMedia sprite on the Stage.
2. Open the Library palette (Window > Library Palette), and select Media > RealMedia from the pop-up menu to display the RealMedia behaviors.
3. Drag the RealMedia Target behavior to the RealMedia sprite on the Stage.
4. Enter the number of milliseconds for a long and short seek operation, and assign the behavior to a group or accept the defaults.

   The number of milliseconds you specify for a long or short seek is the number that is used by the various forward and back options of the RealMedia Control Button behavior. Be aware that short seeks are ineffective because the amount of time it takes the stream to rebuffer is generally longer than the number of milliseconds that are skipped in the stream.
5. Create graphic sprites to act as slider controls and Play, Stop, and Pause buttons for the RealMedia sprite and place them on the Stage. You can also create a graphic sprite to display the buffering progress of the RealMedia stream.

   These sprites should be basic graphic sprites, not functional buttons; the RealMedia behaviors add the button functionality.
6. Drag the RealMedia Control Button, RealMedia Slider Bar, RealMedia Slider, and RealMedia Buffering Indicator behaviors to the sprites that you created on the Stage, and select the appropriate action and group using the pop-up menu in the Parameters dialog box.

   The group to which you assign the behavior must be the same group you created for the RealMedia Target behavior.

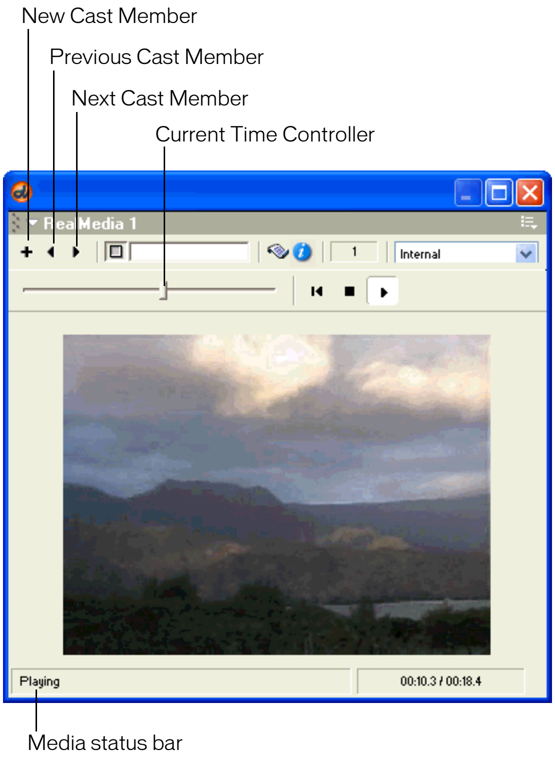
7 Create a field on the Stage to display playback information about the RealMedia sprite similar to the information that appears in the status bar of the RealMedia viewer.

8 Drag the RealMedia Stream Information behavior to the field; then select the type of information you want to display and the group that the behavior belongs to in the Parameters dialog box.

You can create as many of these features as you like. You do not have to use control buttons in movies with RealMedia cast members if you want to control them from the Score or by using Lingo or JavaScript syntax.

The RealMedia viewer

The new RealMedia viewer is a simple media viewer that lets you play RealMedia cast members in isolation from other elements of your movie. You cannot edit RealMedia cast members in the RealMedia viewer.



The RealMedia viewer

The viewer has the following controls:

New Cast Member (+) lets you create a new RealMedia cast member. You need to open the Property inspector to specify the name and filename for the cast member.

Next Cast Member (right arrow) lets you view the next RealMedia cast member (in the current cast) in the viewer.

Previous Cast Member (left arrow) lets you view the previous RealMedia cast member (in the current cast) in the viewer.

Play initiates the streaming process for the current RealMedia cast member. For more information about the streaming process, see the `state (RealMedia)` property in the Scripting Reference topics in the Director Help Panel.

Rewind stops playback, empties the stream buffer, and resets the stream to the beginning. This is equivalent to the `stop (RealMedia)` method in script. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

Stop stops the playback but does not reset the stream to the beginning or empty the stream buffer. If the user clicks Play after clicking Stop, play resumes where it left off, without rebuffering (unless it is a live stream, in which case it rebuffers to join the live stream in progress). This is equivalent to the `pause (RealMedia)` method in script. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

Current Time Controller lets the user jump ("seek") to any position in the RealMedia stream. The slider is disabled if the duration of the stream is not yet known (for example, the first time a stream plays), if `mediaStatus` is `#closed`, or if the cast member references a live feed. If the user drags the slider while the stream is playing, the stream buffers and automatically starts playing from the new position, but if the slider is dragged while the stream is paused or stopped, the user must click the Play button to restart the stream. This is equivalent to the `seek` method in script. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

The media status bar displays the current value of the `mediaStatus` property on the left, and the current time and duration of the stream on the right, in the format MM:SS.S or HH:MM:SS.S. If the stream is playing, the status appears as Playing. For more information about the `mediaStatus` property, see the Scripting Reference topics in the Director Help Panel.

To view a RealMedia cast member in the RealMedia viewer:

1 Select Window > RealMedia to open the RealMedia viewer.

2 Do one of the following:

- Click the Play button to play the RealMedia cast member that is currently on the Stage or selected in the Cast window.
- Use the Next (right arrow) and Previous (left arrow) buttons to select the RealMedia cast member you want to view, and click the Play button.
- Double-click a RealMedia cast member in the Cast window; a RealMedia viewer opens automatically.

Publishing Shockwave content with RealMedia

The Xtra for RealMedia is not part of the standard Shockwave 8.5, but can be downloaded from the Macromedia website. The first time a user attempts to play Shockwave content that contains a RealMedia cast member, the Shockwave Player automatically downloads and installs the Xtra for RealMedia if you selected RealMedia Asset.x32 in the Movie Xtras dialog box.

To add the Xtra for RealMedia to your movie:

1 Select Modify > Movie > Xtras to display the Movie Xtras dialog box.

2 Select RealMedia Asset.x32 in the list.

If RealMedia Asset.x32 does not appear in the list, click the Add button, and select it in the Add Xtras dialog box.

3 Select the Include in Projector and Download if Needed options.

4 Click OK.

Using Lingo or JavaScript syntax sound elements with RealMedia

All the Lingo or JavaScript syntax elements in this section are documented in the main *Director Scripting Reference* and are discussed here only as they pertain to working with RealMedia content. For complete information, see the *Director Scripting Reference.*

Supported sound elements

The following Lingo or JavaScript syntax elements operate on a sound channel and are fully supported for sound channels playing the audio portion of a RealMedia cast member:

- `elapsedTime`
- `fadeIn()`
- `fadeOut()`
- `fadeTo()`
- `pan (Sound Channel)`
- `soundBusy()`

Although you can use the following Lingo or JavaScript syntax elements with a RealMedia cast member, they cause problems when used on a sound channel. For example, you can use `member("Real").stop()` but should not use `sound(`*`whichChannel`*`).stop()` if the audio portion of a RealMedia cast member uses *`whichChannel`*.

- `member` (sound property)
- `pause()`
- `play()`
- `stop()`

You can use the following property on a sound channel playing the audio portion of a RealMedia stream but not directly on a RealMedia cast member. For example, you can use `sound(`*`whichChannel`*`).volume = 200`, but not `member("Real").volume = 200`.

- volume

You can set the system variable `soundEnabled` to `FALSE` to turn off RealAudio, but if you reset it to `TRUE`, you must also call the `play` method to resume playback.

Unsupported sound elements

The following Lingo or JavaScript syntax elements are not supported for RealMedia cast members or for sound channels playing the audio portion of a RealMedia stream:

- `breakLoop()`
- `channelCount`
- `endTime`
- `getPlayList()`
- `loopCount`
- `loopEndTime`
- `loopsRemaining`
- `loopStartTime`
- `play()`
- `playFile()`
- `playNext()`
- `queue()`
- `rewind()`
- `sampleCount`
- `setPlayList()`
- `status` (Use the `state (RealMedia)` or `mediaStatus` RealMedia cast member properties instead. For more information about these properties, see the Scripting Reference topics in the Director Help Panel.)

CHAPTER 12
Behaviors

A behavior in Macromedia Director MX 2004 is prewritten Lingo or JavaScript syntax that you use to provide interactivity and add interesting effects to your movie. You drag a behavior from the Library palette and drop it on a sprite or frame to attach it.

If the behavior includes parameters, a dialog box appears that lets you define those parameters. For example, most navigation behaviors let you specify a frame to jump to. You can attach the same behavior to as many sprites or frames as necessary and use different parameters for each instance of the behavior.

Most behaviors respond to simple events such as a mouse click on a sprite or the entry of the playhead into a frame. When the event occurs, the behavior performs an action, such as jumping to a different frame or playing a sound.

Director comes packaged with customizable, reusable behaviors for many basic functions; you and other developers can also create and share your own behaviors by writing Lingo or JavaScript syntax. To modify behaviors, you use the Behavior inspector or Property inspector.

For more information about using included behaviors, see Using Director Behaviors on the Director Support Center at www.macromedia.com/go/director_behaviors_en.

Attaching behaviors

You use the Library palette to display behaviors included in Director.

Director allows you to attach the same behavior to several sprites or several frames at the same time. You can attach as many behaviors as you want to a sprite, but you can attach only one behavior to a frame. If you attach a behavior to a frame that already has a behavior, the new behavior replaces the old one. Behaviors attached to frames are best suited to actions that affect the entire movie. For example, you might attach `Loop Until Media in Frame is Available` to make the movie wait while the media for a particular frame downloads.

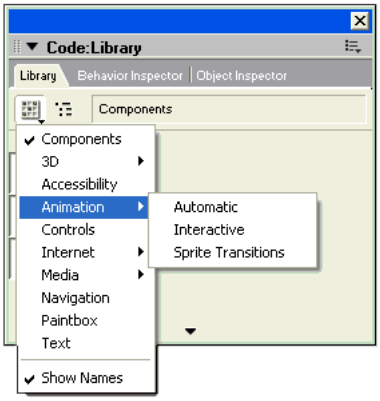
When you attach a behavior, and the Parameters dialog box appears, the parameters you specify apply to the behavior only as it is attached to the current sprite or frame. These settings do not affect the way the behavior works when attached elsewhere. Use the Behavior inspector to change parameters for behaviors attached to sprites or frames.

Once you attach a behavior to a sprite or frame, Director copies the behavior from the Behavior library to the currently selected cast in the movie. This means you do not have to include the Behavior library when you distribute the movie.

To attach a behavior to a single sprite or frame using the Library palette:

1 Select Window > Library Palette.

2 Select a library from the Library pop-up menu in the upper left corner of the palette.

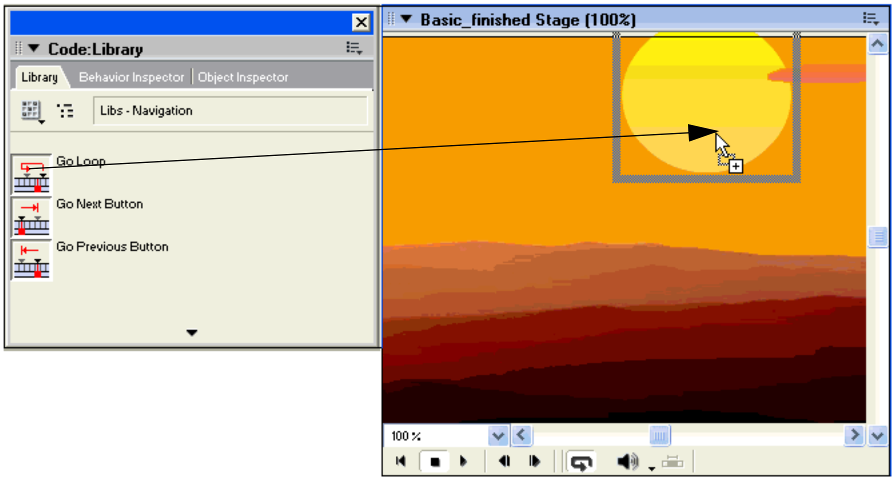


3 To view a brief description of included behaviors, move the pointer over a behavior icon.

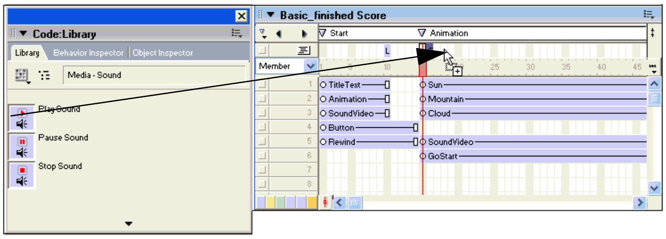
If the behavior includes a longer description, you can view it in the Behavior inspector or in Director Help. For more information, see "Getting information about behaviors" on page 278. The behaviors included with Director come with descriptions. Behaviors from other sources may not.

Select Show Names from the Library pop-up menu to turn the display of behavior names on or off.

4 To attach a behavior to a single sprite, drag a behavior from the Library palette to a sprite on the Stage or in the Score.



5 To attach a behavior to a frame in the behavior channel, drag a behavior from the Library palette to a frame in the behavior channel.



6 Enter parameters for the behavior in the Parameters dialog box.

Note: If you attach a behavior from a Director library of behaviors, the behavior is copied to the currently active cast.

To attach the same behavior to several sprites at once using the Library palette:

- Select the sprites on the Stage or in the Score and drag a behavior to any one of them.

To attach behaviors that have already been copied to a cast:

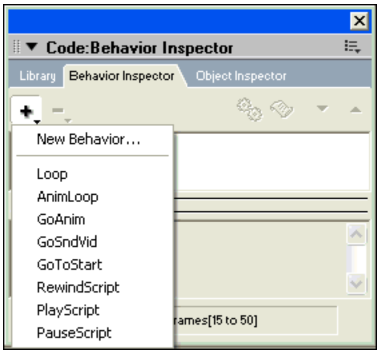
1 Select Window > Behavior Inspector to open the Behavior inspector.

2 Do one of the following:

- Select a sprite or several sprites.
- Select a frame or several frames.

3 Select a behavior from the Behaviors pop-up menu.

Director attaches the behavior you select to the sprite(s) or frame(s).



Note: Some behaviors work only when applied to either a sprite or a frame; for more information, read the behavior descriptions.

To change parameters for a behavior attached to a sprite or frame:

1 Select the sprite or frame to which the behavior is attached.

2 In the Behavior tab of the Property inspector, use the pop-up menus or text boxes to change any parameters.

The Behavior tab has the same fields for the behavior as those in the Parameters dialog box.

Changing the order of attached behaviors

Director executes behaviors in the order they were attached to a sprite, and they are listed in this order in the Property inspector and Behavior inspector. It is sometimes necessary to change the sequence of behaviors so that actions occur in the proper order.

To change the order of the behaviors attached to a sprite:

1 Select the sprite in the Score or on the Stage.

2 Open the Behavior inspector or click the Behavior tab in the Property inspector.

3 Select a behavior from the list.

4 Click the arrows in the toolbar to move the selected behavior up or down on the list.

Getting information about behaviors

Behaviors included with Director have pop-up descriptions that appear when you hold the pointer over a behavior in the Library palette. Some behaviors, however, have longer descriptions and instructions, which you can view in the Behavior inspector. A scrolling pane in the Behavior inspector displays the complete description provided by the behavior's author. The Behavior inspector only displays information about a behavior attached to a sprite or frame.

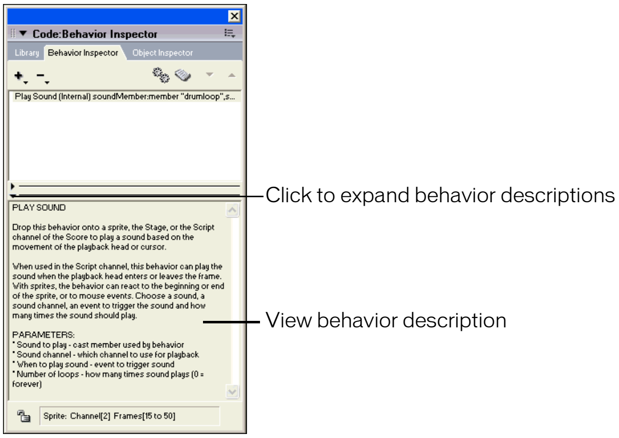
To view a behavior description:

1 Open the Behavior inspector.

2 Select a sprite or frame to which a behavior has been attached.

3 Click the arrow that expands the Behavior inspector's description pane.

You can leave the description pane expanded and select different behaviors to see their descriptions.



Play Sound (Internal) soundMember:member "drumloop",s...

PLAY SOUND

Drop this behavior onto a sprite, the Stage, or the Script channel of the Score to play a sound based on the movement of the playback head or cursor.

When used in the Script channel, this behavior can play the sound when the playback head enters or leaves the frame. With sprites, the behavior can react to the beginning or end of the sprite, or to mouse events. Choose a sound, a sound channel, an event to trigger the sound and how many times the sound should play.

PARAMETERS:
* Sound to play - cast member used by behavior
* Sound channel - which channel to use for playback
* When to play sound - event to trigger sound
* Number of loops - how many times sound plays (0 = forever)

Sprite: Channel[2] Frames[15 to 50]

All of the behaviors included in the Director library have descriptions. Behaviors from other developers may not.

Creating and modifying behaviors

Without any scripting or programming experience, you can use the Behavior inspector to create and modify behaviors to perform basic actions. To create behaviors with more complex structures, you must understand scripting in Lingo or JavaScript syntax.

Using the Behavior inspector is a good way to learn Lingo or JavaScript syntax. You can examine the scripts created by the Behavior inspector to see how basic functions are assembled. To view the associated script, select any behavior and click the Script button.

Most behaviors detect an event and then perform one or more actions in response. The Behavior inspector lists the most common events and actions used in behaviors.

For experienced programmers, the Behavior inspector also provides a shortcut for writing simple scripts.

Note: To always edit behaviors in the Script window instead of the Behavior inspector, select Edit > Preferences > Editors. In the Editors Preferences dialog box, select Behaviors from the list and then click Edit. In the Select Editor box, select Script Window. (If you are using a Macintosh OS X operating system, select the Director menu, instead of the Edit menu, to access Preferences.)

To create or modify a behavior:

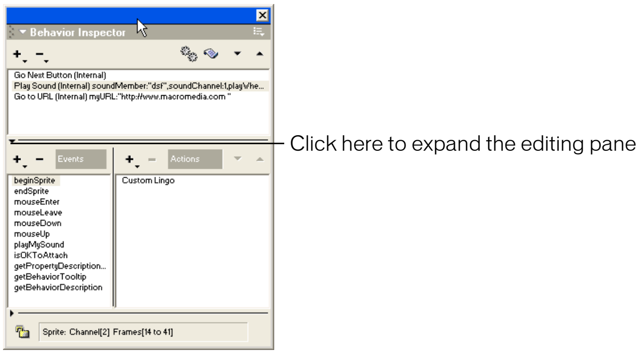
1 Do one of the following:

- To create a new behavior, click the Behaviors pop-up menu, select New Behavior, and enter a name for the new behavior.

  The behavior appears in the currently selected Cast window in the first empty position. Select an empty cast position first if you want the behavior to appear in a different place.

- To modify a behavior, select it in the Behavior inspector.

2 Click the arrow in the lower left of the Behavior inspector to expand the editing pane.



Go Next Button (Internal)
Play Sound (Internal) soundMember:"dsf",soundChannel:1,playWhe...
Go to URL (Internal) myURL:"http://www.macromedia.com "

beginSprite
endSprite
mouseEnter
mouseLeave
mouseDown
mouseUp
playMySound
isOKToAttach
getPropertyDescription...
getBehaviorTooltip
getBehaviorDescription

Custom Lingo

Sprite: Channel[2] Frames[14 to 41]

The editing pane shows the events and actions in the current behavior. If you are creating a new behavior, no events or actions appear.

- To add a new event or action group to the behavior, select an event from the Events pop-up menu and then select actions for the event from the Actions pop-up menu.

You can choose as many actions as you need for a single event.

- To change an existing event or action group, select an event from the list and then add or remove actions in the Actions list.
- To delete an event or action group, select the event and press Delete.
- To change the sequence of actions in an event or action group, select an event from the Events list, select an action from the Actions list, and then click the up and down arrows above the Actions list to change the order of actions.
- To lock the current selection so nothing changes in the Behavior inspector when new sprites are selected, click the Lock Selection button in the lower left of the expanded Behavior inspector.

If you are familiar with Lingo or JavaScript syntax, you can also edit a behavior's script directly.

Events and actions in the Behavior inspector

The actions and events included with Director are basic building blocks you can use to create simple or complex behaviors.

The Behavior inspector makes the following events available:

BeginSprite contains the statements that run when the playback head moves to a frame that contains a sprite that was not previously encountered.

End Sprite contains the statements that run when the playback head leaves a sprite and goes to a frame in which the sprite does not exist.

MouseUp indicates that the mouse button was released.

MouseDown indicates that the mouse button was clicked.

RightMouseUp indicates that the right mouse button was released. (On the Macintosh, Director treats a Control-click the same as a right mouse click on a Windows system.)

RightMouseDown indicates that the right mouse button was clicked.

MouseEnter indicates that the pointer entered a sprite's region.

MouseLeave indicates that the pointer left a sprite's region.

MouseWithin indicates that the pointer is within the sprite's region.

KeyUp indicates that a key was released in a text or field sprite.

KeyDown indicates that a key was pressed in a text or field sprite.

PrepareFrame indicates that the playhead has left the previous frame but has not yet entered the next frame.

EnterFrame indicates that the playhead has entered the current frame.

ExitFrame indicates that the playhead has exited the current frame.

NewEvent indicates that a specified message was received from a script or behavior. You must specify a name for this event.

The Behavior inspector makes the following actions available:

Go to Frame moves the playhead to the specified frame.

Go to Movie opens and plays the specified movie.

Go to Marker moves the playhead to the specified marker.

Go to Net Page goes to the specified URL.

Wait on Current Frame waits at the current frame until another behavior or script advances to the next frame.

Wait until Click waits at the current frame until the mouse button is clicked.

Wait until Key Press waits at the current frame until a key is pressed.

Wait for Time Duration waits at the current frame for the specified time.

Play Cast Member plays the specified sound cast member.

Play External File plays the specified external sound file.

Beep plays the current system beep.

Set Volume sets the system volume level to the specified setting.

Change Tempo changes the movie's tempo to the specified setting.

Perform Transition performs the specified transition.

Change Palette changes to the specified palette.

Change Location moves the current sprite to the specified coordinates.

Change Cast Member switches the sprite's cast member to the specified cast member.

Change Ink switches to the specified ink.

Change Cursor changes the pointer to a shape you select from the pop-up menu.

Restore Cursor restores the current system pointer.

New Action executes any method or sends a message to a handler. You specify the new handler's name.

Writing behaviors with Lingo or JavaScript syntax

If you are familiar with Lingo or JavaScript syntax, you can author your own behaviors. A behavior is a Lingo or JavaScript syntax script with these additional features:

- Each instance of the behavior has independent values for properties. The script uses a `property` statement to declare properties that can have independent values in each instance of the behavior. For more information, see the Scripting Reference topics in the Director Help Panel.
- The same set of handlers can be shared by multiple sprites or frames.

  The handlers in a behavior are basically the same as other handlers. Include as many handlers as appropriate to implement the behavior.

  A behavior is usually attached to multiple sprites or frames. As a result, the sprites and frames share the same handlers. Director tracks which instance of the behavior is which by assigning each instance a reference number. The variable `me` contains the reference for the object that the instance of the behavior is attached to.

  In many cases, it is most efficient to create behaviors dedicated to specific tasks and then attach a set of behaviors that perform together the variety of actions you want.

- The behavior can have parameters that users edit from the Parameters dialog box. The optional `on getPropertyDescriptionList` handler sets up the Parameters dialog box. For more information, see the Scripting Reference topics in the Director Help Panel.
- A description of the behavior can be added to the Behavior inspector. The optional `on getBehaviorDescription` handler displays a description of the behavior in the Behavior inspector. For more information, see the Scripting Reference topics in the Director Help Panel.
- A brief description appears as a tooltip for the behavior in the Library palette if the optional `on getBehaviorToolTip` handler that creates the tooltip has been written. For more information, see the Scripting Reference topics in the Director Help Panel.

Setting up a Parameters dialog box

It is impossible to predict exactly what a user will want behaviors to do. You can make behaviors more flexible by letting the user customize the behavior's parameters.

For example, this handler moves the sprite 5 pixels to the right each time the playhead enters a new frame:

```
if ( sprite(me.spriteNum).locH > window("stage").rect.right ) then
  sprite(me.spriteNum).locH = window("stage").rect.left
else
  sprite(me.spriteNum).locH = sprite(me.spriteNum).locH + 5
end if
```

However, users could adjust the speed of each sprite if they could specify how far individual sprites move to the right in each frame.

To allow users to set different values for a property in different instances of the behavior, the behavior's script needs the following:

- A `property` statement (Lingo) or a `var` statement (JavaScript syntax) that allows each instance to maintain a separate value for the property
- An `on getPropertyDescriptionList` handler that sets up the property or variable.

Setting behavior properties with script

Behaviors usually have properties for which each instance of the behavior maintains its own values. (An instance is the unique instance of the behavior assigned to sprites or frames.) These properties are shared among handlers in a behavior's script the same way that properties are shared among handlers in an object.

To declare properties in each instance of the behavior:

- Put the `property` or `var` statement at the beginning of the behavior's script.

A `property` or `var` statement starts with the word `property` or `var` followed by the names of the individual properties or variables. For example, the statement `property movement` declares that `movement` is a property of the behavior.

Customizing a behavior's property

If a behavior's script includes an `on getPropertyDescriptionList` handler, Director lets users set the property's initial values from the Parameters dialog box. The behavior's Parameters dialog box opens in three circumstances:

- After the user drags a behavior to a sprite or frame
- When the user double-clicks the behavior in the Behavior inspector dialog box
- When the user clicks the Parameters button in the Behavior inspector

The `on getPropertyDescriptionList` handler generates a property list that specifies these attributes of the property or variable:

- The default initial value
- The type of data the property or variable contains, such as Boolean, integer, string, cast member, or a specific type of cast member
- A comment in the Parameters dialog box to describe what the user is setting

The definition of a behavior's property or variable must include the property's or variable's name, default value, and data type and the descriptive string that appears in the Parameters dialog box. The definition can also include an optional specification for the range of values allowed for the property or variable.

The name of the property or variable comes first in the definition. The remainder of the definition is a property list that assigns a value to each of the property's or variable's attributes.

For example, to define the property `movement` as an integer that can be set to a value from 1 to 10 and whose default value is 5, use a phrase similar to this:

```
#movement: [#default: 5, #format:#integer,
#comment: "Set motion to the right:", #range: [#min:1, #max:10]]
```

- `#movement` is the property's name. A symbol (#) operator must precede the name in the property definition. A colon separates the name's definition and the list of parameters.
- `#default` specifies the property's default value. This example sets 5 as the default.
- `#format` specifies the property's type. This example sets the type as an integer. Some other possible types are Boolean, string, cast member, event, and sound. For more information, see the Scripting Reference topics in the Director Help Panel.
- `#comment` specifies a string that appears next to the parameter in the Parameters dialog box. This example makes "Set motion to the right" the comment that appears in the Parameters dialog box.

- #range specifies a range of possible values that the user can assign to the property. Specify the possible values as a list.

  To specify a range between a minimum and maximum number, use the form [#min:minimum, #max:maximum]. The example sets the range from 1 to 10. When the range is between a maximum or minimum number, the Parameters dialog box provides a slider that sets the value.

  To specify no range, omit the #range parameter. If the property's definition does not include #range, a text entry field appears for the user to enter a value in the Parameters dialog box.

  To specify a set of possible choices, use a linear list. For example, the list [#mouseUp, #mouseDown, #keyUp, #keyDown] makes these four events possible choices for a parameter. When you specify values in a linear list, the choices appear in a pop-up menu in the Parameters dialog box. (For this example list, you need to specify #format: #symbol for the list to display correctly.)

As another example, this statement defines the property whichSound:

```
description.addProp(#whichSound,  [#default: "", #format:#sound, #comment: \
"Which cast member"]
```

The value #sound assigned to #format provides a pop-up menu in the Parameters dialog box that includes every sound cast member available in the movie.

If the behavior includes a method that plays a sound, this property can be used to specify a sound cast member to play. For example, if the user selects Growl from the pop-up menu in the Parameters dialog box, the statement puppetSound whichSound would play the sound cast member Growl.

Creating an on getPropertyDescriptionList handler

To build a list of properties for a behavior, add each property to the list that the on getPropertyDescriptionList handler returns. Then use the return method to return the list.

For example, this handler creates a property list named Description that contains the definitions for movement and whichSound:

```
--Lingo syntax

on getPropertyDescriptionList

  description = [:]

  description[#movement] = \
                [#default: 5, \
                 #format:#integer, \
                 #comment: "Set motion to the right:", \
                  #range: [#min:1, #max:10] \
                ]

  description[#noise] = \
                [#default:"", \
                 #format: #sound, \
                 #comment:"Sound cast member name" \
               ]

return description

end
```

```
// JavaScript syntax

function getPropertyDescriptionList() {
  var description = new Array();

  description[#movement] = [:]
  description[#movement][#default] = 5
  description[#movement][#format] = #integer
  description[#movement][#comment] = "Set motion to the right:"
  description[#movement][#range] = [#min:1, #max:10]

  description[#noise] = [:]
  description[#noise][#default] = ""
  description[#noise][#format] = #sound
  description[#noise][#comment] = "Sound cast member name"

  return description
}
```

Including a description for the Behavior inspector

An `on getBehaviorDescription` handler in a behavior's script provides a description that appears in the bottom pane of the Behavior inspector when the behavior is selected. For example, this handler displays the phrase "This changes sprite color and position" in the Behavior inspector:

```
on getBehaviorDescription
  return "This changes sprite position"
end
```

Example of a complete behavior

If the handlers described here were in one behavior, the script would look like this (the `puppetSound` method was added to the `on mouseUp` handler in this example):

```
--Lingo syntax

property movement
property noise

on enterFrame me
  if ( sprite(me.spriteNum).locH > window("stage").rect.right ) then
    sprite(me.spriteNum).locH = window("stage").rect.left
  else
    sprite(me.spriteNum).locH = sprite(me.spriteNum).locH + 5
  end if
end

on mouseUp me
  sprite(me.spriteNum).foreColor = random(255)
  puppetSound noise
end

on getBehaviorDescription(me)
  return "This changes sprite position"
end

on getPropertyDescriptionList(me)
```

```
  description = [:]

  description[#movement] = \
                 [#default: 5, \
                  #format:#integer, \
                  #comment: "Set motion to the right:", \
                   #range: [#min:1, #max:10] \
                 ]

  description[#noise] = \
                 [#default:"", \
                  #format: #sound, \
                  #comment:"Sound cast member name" \
                ]

  return description
end

// JavaScript syntax

function enterFrame() {
  if (sprite(spriteNum).locH  > _movie.stageRight) {
    sprite(spriteNum).locH = _movie.stageLeft
  } else {
    sprite(spriteNum).locH += movement
  }
}

function mouseUp() {
  sprite(spriteNum).foreColor = Math.floor(Math.random())*255
  sound(noise)
}

function getBehaviorDescription() {
  return "This changes sprite color and position"
}

function getPropertyDescriptionList {

  description = new Array();

  description["Movement"] = new Array();
  description["Movement"]["default"] = 5;
  description["Movement"]["format"] = "integer";
  description["Movement"]["comnt"] = "Set motion to the right";
  description["Movement"]["range"] = new Array();
  description["Movement"]["range"]["min"] = 1;
  description["Movement"]["range"]["max"] = 10;

  description["noise"] = new Array();
  description["noise"]["default"] = "";
  description["noise"]["format"] = "sound";
  description["noise"]["comment"] = "Sound cast member name";

  return description;
}
```

When this behavior is attached to a sprite, each time the playhead enters a frame, the sprite moves to the right by the amount the user specifies. When the user clicks a sprite, its color changes and a specified sound plays.

Sending messages to behaviors attached to sprites

Script can run handlers in behaviors attached to specific sprites by sending messages to the behaviors attached to one sprite, all sprites, or several specific sprites.

Sending messages to a sprite

The `sendSprite` method sends a message to a specified sprite. If none of the sprite's behaviors has a handler that corresponds to the message, the message passes to the cast member script, the frame script, and then the movie script. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

For example, this handler sends the custom message `bumpCounter` and the argument 2 to sprite 1 when the user clicks the mouse:

```
--Lingo syntax

on mouseDown me
  sendSprite (1, #bumpCounter, 2)
end

// JavaScript syntax

function mouseDown() {
  _movie.sendSprite(1, symbol("bumpCounter"), 2);
}
```

Note: The symbol (#) operator must precede the message in the `sendSprite` method.

Sending messages to all sprites

The `sendAllSprites` method sends a message to every sprite in the frame. If no behavior has a handler that corresponds to the message, the message passes to the cast member script, the frame script, and then the movie script. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

For example, this handler sends the custom message `bumpCounter` and the argument 2 to all sprites in the frame when the user clicks the mouse button:

```
--Lingo syntax

on mouseDown me
  sendAllSprites (#bumpCounter, 2)
end

// JavaScript syntax

function mouseDown() {
  _movie.sendAllSprites(symbol("bumpCounter"), 2);
}
```

Note: The symbol (#) operator must precede the message in the `sendAllSprites` method.

Sending messages to specific behaviors only

The `call` method sends an event to specific behaviors. Unlike the `sendSprite` method, the `call` method does not pass the message to frame scripts, scripts of the cast member, or movie scripts.

Before sending a message to a specific behavior, check the `scriptInstanceList` sprite property to find a behavior script reference to use with the `call` method.

The `scriptInstanceList` property provides a list of references for the behaviors attached to a sprite while a movie is playing.

For example, this handler displays the list of references for all behaviors attached to the same sprite as this behavior's handler:

```
--Lingo syntax

on showScriptRefs me
  put sprite(me.spriteNum).scriptInstanceList
end

// JavaScript syntax

function showScriptRefs() {
  trace(sprite(spriteNum).scriptInstanceList);
}
```

This handler sends the message `bumpCounter` to the first script reference attached to sprite 1 (the `getAt` method identifies the first script reference in the `scriptInstanceList`):

```
--Lingo syntax

on mouseDown me
  xref = getAt(sprite(1).scriptInstanceList, 1)
  call (#bumpCounter, xref, 2)
end

// JavaScript syntax

function mouseDown() {
  xref = sprite(1).scriptInstanceList.getAt(1);
  bumpCounter (xref, 2);
}
```

Note: The symbol (#) operator must precede the message in the `call` method.

To remove all instances of a sprite while the movie is playing:

- Set the sprite's `scriptInstanceList` property to an empty list([]). For more information about this property, see the Scripting Reference topics in the Director Help Panel.

Using inheritance in behaviors

Behaviors can have ancestor scripts in the same way that parent scripts do. (Ancestor scripts are additional scripts whose handlers and properties a parent script can call on and use.)

- The ancestor's handlers and properties are available to the behavior.
- If a behavior has the same handler or property as an ancestor script, then the script uses the property or handler in the behavior instead of the one in the ancestor.

For more information about the concept of ancestors and inheritance, see the Scripting Reference topics in the Director Help Panel.

To make a script an ancestor, do one of the following:

- Declare that `ancestor` is a property in the `property` statement at the beginning of the behavior's Score script.

  For example, the statement `property ancestor` declares that `ancestor` is a property.
- Include a statement that specifies which script is the ancestor. Put the statement in an `on beginSprite` handler in the behavior.

  For example, this handler makes the script Common Behavior an ancestor of the behavior when Director first enters the sprite:

  ```
  --Lingo syntax

  on beginSprite
    sprite(me.spriteNum).ancestor = script("Common Behavior").new()
  end
  ```

  This handler will let the behavior also use the handler in the script Common Behavior.

CHAPTER 13
Navigation and User Interaction

Adding interactivity lets you involve your audience in your Macromedia Director MX 2004 movies. Using the keyboard, the mouse, or both, your audience can download content from the Internet, jump to different parts of movies, enter information, move objects, click buttons, and perform many other interactive operations.

Unless made to do otherwise, a movie plays through every frame in the Score from start to finish. Behaviors and Lingo or JavaScript syntax script can make the movie jump to a different frame, movie, or URL when a specified event occurs. With script, you can include simple navigation instructions as part of more complex handlers; you can also place navigation code in movie scripts and scripts that are attached to cast members such as buttons.

There are several other interactive features that you can add to your movie:

- Draggable sprites give your audience the ability to move sprites anywhere on the Stage. You can also create boundaries beyond which sprites cannot move.
- Editable fields are fields in which your audience can enter or edit information.
- Rollovers make certain sprites change in appearance when the mouse pointer passes over them, even if the user has not clicked the mouse. Using rollovers is an excellent way to give your audience feedback based on their actions.
- The mouse pointer (that is, the cursor) can be changed based on criteria you select. Using script, you can provide animated cursors or specify one of the standard cursors or a bitmap cast member as a cursor image. For more information, see the Scripting Reference topics in the Director Help Panel.
- Push buttons, radio buttons, and check boxes provide an easy to way to quickly create user interfaces for forms or applications.

Creating basic navigation controls with behaviors

Director provides behaviors that let you create basic navigation controls without knowing Lingo or JavaScript syntax. You can use behaviors to move the playhead to a frame number or marker. You can also stop the playhead at any frame and wait for the user to act.

The following examples explain the basic use of the `Hold on Current Frame` and `Go Next Button` behaviors. You can also create custom navigation behaviors or get them from third-party developers. For information about using behaviors, see Chapter 12, “Behaviors,” on page 275.

To use basic navigation behaviors:

1 Create a movie that contains a sprite in frame 1 and at least one marker in a later frame.

2 Select Window > Library Palette, and select the Navigation library.

3 Drag `Hold on Current Frame` to frame 1 in the script channel.

Typically, you use this behavior in a frame that requires user interaction such as selecting a menu command.

4 Play the movie.

The playhead remains in frame 1 where you attached the behavior. The movie is still playing, but the playhead remains on the single frame. Use the `Go Next Button` behavior to send the playhead to a new frame and continue playing, as described in the following steps.

5 Stop the movie.

6 Drag the `Go Next Button` behavior from the Library palette to the sprite in frame 1.

7 Rewind and play the movie again.

The playhead is again stopped in the first frame by the `Hold on Current Frame` behavior.

8 Click the sprite to which you attached the `Go Next Button` behavior.

The playhead jumps to the frame that contains the next marker and continues playing.

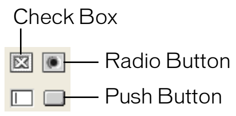
Adding push buttons, radio buttons, and check boxes

Director MX provides several built-in user interface elements for quickly adding interactivity to your movies. These elements include push buttons, radio buttons, and check boxes.

To add a push button, radio button, or check box:

1 Open the Tool palette by selecting Window > Tool Palette, if it's not already open.

2 Select the Push Button, Radio Button, or Check Box tool in the Tool palette.



3 Click and drag on the Stage to create the selected button type.

4 Type a label in the text area next to the button or check box.

Setting properties for push buttons, radio buttons, and check boxes

When you create a push button, radio button, or check box on the Stage, a Button cast member is added to the Cast. You can use button cast member properties to change the name and button type of button cast members.

To view or change button cast member properties:

1 Select a button cast member (a push button, radio button, or check box) and click the Member tab of the Property inspector using the Graphical view.

2 To view or edit the cast member name, use the Name text box.

3 To specify how Director removes the cast member from memory if memory is low, select an option from the Unload pop-up menu. For more information, see "Controlling cast member unloading" on page 47.

4 To change the type of button, click the Button tab and select Push Button, Check Box, or Radio Button from the Type pop-up menu.

Jumping to locations with Lingo or JavaScript syntax

The Lingo or JavaScript syntax navigation features can make a movie jump to other frames, to other movies, or to Internet movies and web pages. You can also use script to make a movie appear to pause by looping in one frame or a group of frames.

For more information about specifying the locations of frames, markers, and movies in Lingo and JavaScript syntax, see the Scripting Reference topics in the Director Help Panel.

Jumping to a different frame

Lingo or JavaScript syntax let you jump to a different frame in the current movie or in another movie.

- To jump to a specific frame in the current movie, use the `go` method, and pass it a frame name or number as a parameter.

  For example, the statement `go("Begin Over")` jumps to the frame labeled Begin Over.
- To jump to the beginning of a different movie, or to a specific frame in a different movie, use the `go` method, and pass it the name of the movie to jump to, and the frame name or number within that movie to jump to.

  For example, the statement `go("Rosebud", "Citizen_Kane")` jumps to a frame labeled Rosebud in the movie Citizen_Kane.dir.

For more information, see the Scripting Reference topics in the Director Help Panel.

Jumping to a URL

Lingo or JavaScript syntax lets you jump to a URL that represents an Internet movie or a web page.

- To jump to an Internet movie, use the `gotoNetMovie` method.

  For example, the statement `gotoNetMovie "http://www.yourserver.com/movies/movie1.dcr"` retrieves and plays the movie named movie1.dcr. For more information about this method, see the Scripting Reference topics in the Director Help Panel.
- To jump to a web page, use the `gotoNetPage` method.

  For example, the statement `gotoNetPage "http://www.yourserver.com/movies/intro.html"` displays the web page named intro.html in a browser window. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

Looping in a group of frames

Looping within frames lets you create animation that recycles or makes a movie appear to pause. Such looping is useful for allowing a network operation to complete before the movie proceeds. Looping a movie by jumping from the current frame back to the first frame in the sequence can create a recycling animation effect.

- To loop within one segment of the Score, use the `go loop` statement to return to the first marker to the left of the frame that contains the `go loop` statement. If there is no previous marker, the playhead jumps to frame 1.
- To pause a movie in one frame but keep it playing so the movie can react to events, use the statement `go to the frame` to loop in the current frame.
- To resume playing a movie that is looping in one frame, use the `go to the frame + 1` statement.

Jumping away and returning to the original location

You might want a movie to jump to a different frame or a separate movie and then return to the original frame. For example, at a website that describes the weather, you could jump to a movie segment that explains a weather term and then return to the original location.

To jump away and return to the original location:

- Use the `play` and `play done` methods.

  The `play` method branches a movie to another frame, another movie, or a specified frame in another movie. The `play done` method remembers the original frame and returns to it without requiring that you specify where to return.

Use the `play` and `play done` methods in the following situations:

- When the movie you want to play does not have instructions about where to return.
- When you want to play several movies sequentially from a single script. When one movie finishes, the movie returns to the script that issued the `play` method.
- When you want to put one sequence inside another sequence and easily return to where you were in the outer sequence.
- When you want to jump to one loop from several different locations.

For more information about these methods, see the Scripting Reference topics in the Director Help Panel.

Detecting mouse clicks with Lingo or JavaScript syntax

Users can click the mouse button in several ways, each of which can be detected by script. The following are ways that you can use Lingo or JavaScript syntax to detect what the user does with the mouse. For more information, see the Scripting Reference topics in the Director Help Panel.

- To determine the last place the mouse was clicked, use the `clickLoc()` method.
- To determine the last active sprite (a sprite with a script attached) that the user clicked, use the `clickOn` method.
- To determine whether the last two clicks were a double-click, use the `doubleClick` method.
- To determine the time since the mouse was last clicked, use the `lastClick()` method.
- To determine whether the mouse button is pressed, check the `mouseDown` property.
- To determine whether the mouse button is released, check the `mouseUp` property.
- To determine whether the user presses the right mouse button (Windows) or Control-click (Macintosh), check the `rightMouseDown` property.
- To determine whether the user releases the right mouse button (Windows) or Control-click (Macintosh), check the `rightMouseUp` property.

For example, this handler checks whether the user double-clicked the mouse button and, if so, runs the `openWindow` handler:

```
on mouseDown
  if the doubleClick = TRUE then openWindow
end
```

Making sprites editable and draggable

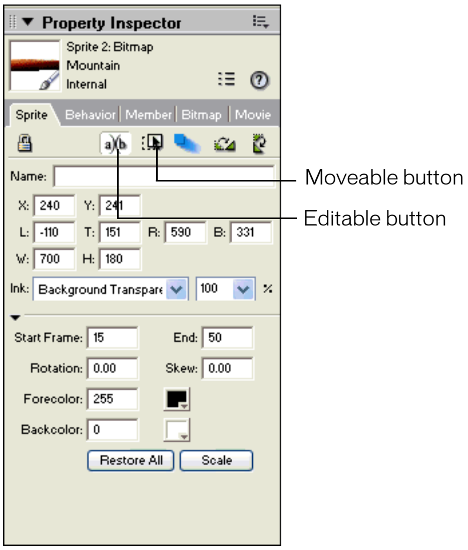
Using the Property inspector, you can make a sprite editable, draggable, or both while your movie is running. For more information, see "Displaying and editing sprite properties in the Property inspector" on page 59.

To make a sprite draggable on the Stage:

- Click the Moveable button in the Property inspector.

To make a text sprite editable:

- Click the Editable button in the Property inspector.



Sprite 2: Bitmap
Mountain
Internal

Name:
X: 240 Y: 241
L: -110 T: 151 R: 590 B: 331
W: 700 H: 180
Ink: Background Transpare 100 %
Start Frame: 15 End: 50
Rotation: 0.00 Skew: 0.00
Forecolor: 255
Backcolor: 0

Making sprites editable or moveable with Lingo or JavaScript syntax

Lingo or JavaScript syntax can make sprites editable or moveable regardless of the settings in the Score. You can also use script to constrain a moveable sprite to a certain region. For example, you can create a draggable slider with an indicator that moves across a gauge. For more information, see the Scripting Reference topics in the Director Help Panel.

- To make a text sprite editable with script, set the text sprite's `editable` property to `TRUE`. For best results, set this property in a script that is attached to the sprite or the frame where the sprite is located.
- To make a sprite moveable with script, set the `moveableSprite` sprite property to `TRUE`. For best results, set this property in a script that is attached to the sprite or the frame where the sprite is located.
- To restrict the registration point of a moveable sprite so it stays within the bounding rectangle of a second sprite, use the `constraint` sprite property.
- To constrain a sprite along a horizontal or vertical path, use the `constrainH()` or `constrainV()` method.

Checking which text is under the pointer with Lingo or JavaScript syntax

Lingo or JavaScript syntax can detect which text component in a text or field cast member is currently under the mouse pointer. For more information, see the Scripting Reference topics in the Director Help Panel.

Use script that applies to text and field cast members as follows:

- To detect which character in a text or field cast member is under the pointer, use the `pointToChar()` method.
- To detect which item in a text or field cast member is under the pointer, use the `pointToItem()` method.
- To detect which word in a text or field cast member is under the pointer, use the `pointToWord()` method.
- To detect which paragraph in a text or field cast member is under the pointer, use the `pointToParagraph()` method.

Use script that applies only to text cast members as follows: To detect whether a specific point is in a hypertext link within a text cast member and is under the pointer, use the `pointInHyperlink()` method. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

Use script that applies only to field cast members as follows:

- To detect which line in a field is under the pointer, use the `mouseLine` property. For more information about this property, see the Scripting Reference topics in the Director Help Panel.
- To detect which word in a field is under the pointer, use the `mouseWord` property. For more information about this property, see the Scripting Reference topics in the Director Help Panel.

Responding to rollovers with Lingo or JavaScript syntax

You often want some action to occur when the user rolls the mouse pointer over a sprite or a particular place on the Stage. You can use Lingo or JavaScript syntax to specify how the movie responds to such rollovers.

Director provides several event handlers that run when the pointer rolls over a sprite. Messages for each of these events are sent to the sprite script, the script of the cast member, the frame script, and then the movie script. For more information, see the Scripting Reference topics in the Director Help Panel.

- To set up script that runs when the mouse pointer enters a sprite's bounding rectangle, place the script in an `on mouseEnter` event handler.
- To set up script that runs when the mouse pointer leaves a sprite's bounding rectangle, place the script in an `on mouseLeave` event handler.
- To set up script that runs when the user clicks a sprite, rolls the pointer off the sprite, and then releases the mouse button, place the script in an `on mouseUpOutside` event handler.

- To set up script that runs when the mouse pointer is within a sprite's bounding rectangle when the playhead enters the frame that contains the sprite, place the script in an `on mouseWithin` event handler.

  The `mouseWithin` event can occur repeatedly as long as the mouse pointer remains inside the sprite.
- To determine whether the cursor is over a specific sprite, use the `rollOver()` method.

Finding mouse pointer locations with Lingo or JavaScript syntax

Determining where the mouse pointer is on the Stage is a common need in Director.

To determine the mouse pointer's horizontal and vertical positions:

- Use the `mouseH` and `mouseV` properties. For more information about these properties, see the Scripting Reference topics in the Director Help Panel.

The `mouseV` property returns the distance, in pixels, between the mouse pointer and the upper-left corner of the Stage. The `mouseH` property returns the distance, in pixels, between the mouse pointer and the upper left corner of the Stage.

The statements `put the mouseH` and `put the mouseV` display the mouse pointer's location in the Message window.

For example, this handler directs the Message window to display the distance (in pixels) between the pointer and the upper left corner of the Stage:

```
on exitFrame
  put(_mouse.mouseH)
  put(_mouse.mouseV)
  _movie.go(_movie.frame)
end
```

Checking keys with Lingo or JavaScript syntax

Lingo or JavaScript syntax can detect the last key that the user pressed. For more information, see the Scripting Reference topics in the Director Help Panel.

- To obtain the ANSI value of the last key that was pressed, use the `key` property.
- To obtain the keyboard's numerical (or ASCII) value for the last key pressed, use the `keyCode` property.

  A common place for using `key` and `keyCode` is in an `on keyDown` handler, which instructs the script to check the value of `key` only when a key is actually pressed. For example, the following handler in a frame script sends the playhead to the next marker whenever the user presses Enter (Windows) or Return (Macintosh):

  ```
  on keyDown
    if the key = RETURN then go to marker (1)
  end
  ```

Equivalent cross-platform keys

Because of inherent differences between Windows and Macintosh keyboards, keys on Windows and Macintosh computers don't always correspond directly.

This discrepancy can create confusion because script often uses the same term to refer to corresponding keys on Windows and Macintosh computers, even though the key's name differs on the two platforms.

The following table lists script elements that refer to specific keys and the keys they represent on each platform.

Lingo term	Windows key	Macintosh key
RETURN	Enter	Return
commandDown	Control	Command
optionDown	Alt	Option
controlDown	Control	Control
ENTER	Enter key on the numeric keypad (during authoring, pressing Enter starts playing the movie)	Enter key on the numeric keypad (during authoring, pressing Enter starts playing the movie)
BACKSPACE	Backspace	Delete

Identifying keys on different keyboards

Characters can vary on different keyboards. You can avoid possible confusion by identifying a character by its ASCII value. For more information, see the Scripting Reference topics in the Director Help Panel.

- To obtain a character's ASCII value, use the `charToNum()` method.

  For example, the following statement finds the ASCII value for the letter A and displays it in the Message window:

  ```
  put charToNum("A")
  -- 65
  ```

- To find out which character corresponds to an ASCII value, use the `numToChar()` method.

  For example, the following statement finds the character that corresponds to the ASCII value 65. The result is the letter A:

  ```
  put numToChar(65)
  -- A
  ```

About animated color cursors

Director supports animated cursors. You can use any 8-bit bitmap source in your Director cast as an image in the cursor animation, automatically scale images, and generate masks for 16 x16 pixel and 32 x 32 pixel cursors. (Macintosh computers don't support 32 x 32 pixel cursors.)

An animated cursor consists of a series of bitmap cast members. Each bitmap cast member is a frame of the cursor. You can control the rate at which Director plays the frames of an animated cursor. Using the Cursor Properties Editor, you designate one or more bitmap cast members as frames of a single cursor cast member.

Xtra extensions that support animated cursors

The Director installation program places two animated color cursor files in the Media Support folder within the Director application's Xtras folder. The specific files depend on the platform you are using.

Windows	PowerPC	Purpose
Cursor Options.x32	Cursor Options	This file supports the creation of cursors while you author movies in Director. Do not distribute this file with projectors; it is not licensed for redistribution.
Cursor Asset.x32	Cursor Asset	Distribute this file with any movies or projectors that you create using the animated color cursors.

Requirements for animated color cursors

All cast members used for an animated color cursor must meet certain criteria:

- They must be bitmap cast members.
- They must have a color depth of 8 bits (256 colors).
- They must use only the first eight or the last eight colors that are in the standard System - Win palette. These colors provide the most predictable results when playing back across platforms. Other colors might not appear correctly.

The cast members need not be in sequence in the cast, and they need not be in the same cast.

A cursor's maximum size depends on the computer:

- With Microsoft Windows 98, Windows 2000, and Windows XP, you can create cursors of either 16 x 16 pixels or 32 x 32 pixels (almost always 32 x 32 pixels, but some video cards might support only 16 x 16 pixels).
- With Macintosh OS X, you can create cursors of 16 x 16 pixels.

When you create cursors in the Cursor Properties Editor, Director dims any size option that is not available on your computer.

The 16 x16 and 32 x 32 pixel sizes are the maximum sizes at which Director can display a cursor on the screen. The actual cast members you specify for the cursor can be larger than the maximum, and Director scales the cast members to the appropriate size, maintaining the aspect ratio as it scales them. If you specify a cast member smaller than the maximum size, Director displays the cast member at its original size, without scaling. For example, if you select a maximum size of 16 x 16 pixels and specify a cursor that is 12 x 14 pixels, Director displays the cursor at 12 x 14 pixels.

Creating an animated color cursor cast member

Before creating an animated color cursor cast member, make sure that the cast members you want to use in the cursor are stored in a cast that is linked to the movie. For more information, see "Managing external casts" on page 47.

To create an animated color cursor cast member:

1 Select Insert > Media Element > Cursor.

Director opens the Cursor Properties Editor, which you use to set up the cursor.

2 From the Cast pop-up menu, select the cast that contains the cast member you want to add as a frame in your cursor.

The cast members used for a single cursor can be stored in different casts.

3 Use the < and > buttons to find the cast member you want.

As you click the buttons, the preview shows a thumbnail of the selected cast member. If you do not see the cast member you want, the cast member probably isn't a bitmap or has a color depth greater than 8 bits (256 colors). The Cursor Properties Editor shows only bitmaps that can be used in an animated color cursor.

You can also enter a cast member number in the Member box and press Tab; Director selects the cast member that has that number or the cast member with the number closest to it.

4 Select the cast member you want, and click Add.

You see the cast member in the Cursor Frames preview area. The Frame X of Y text box shows where the cast member falls within an animated series of cursor frames.

5 Repeat steps 2 through 4 until you have added all the cast members for the cursor.

In the Cursor Frames area, you can use the < and > buttons to review the order of the cursor frames. Click the Remove button to delete the currently selected frame from the cursor (this deletes the cast member only from the cursor animation, not from the cast).

6 In the Interval text box, specify the number of milliseconds that elapse between each frame of the cursor animation.

This interval affects all frames of the cursor and cannot vary for different frames. The cursor frame rate is independent of the frame rate that is set for the movie by using the tempo channel or the `puppetTempo` method.

Note: By inserting the same bitmap in multiple frames of the cursor, you can create the illusion of variable-rate cursor animation.

7 In the Hotspot Position text boxes, specify the location of the mouse pointer's active point.

Director uses this point to track the mouse pointer's position on the screen. For example, Director uses this point's location when it returns values for the `mouseH` and `mouseV` properties. The hotspot also determines the point where a rollover occurs.

The first text box specifies the horizontal (*x*) location, and the second text box specifies the vertical (*y*) location. The upper left pixel is location 0,0. In a 16 x 16 pixel cursor, the lower right pixel is location 15,15. You can't enter a point that is beyond the bounds of the cursor.

8 Click one of the Size options to specify the maximum size of the cursor.

If a Size option is dimmed, your computer does not allow you to create cursors of that size.

9 Select the Automask option if you want the white pixels of the cursor frames to be transparent.

Note: The Automask option makes all white pixels transparent. If you want some white pixels to be opaque, you can't use white for those pixels, but you can achieve the same effect by using the lightest shade of gray available in the system palette.

10 Click OK to close the Cursor Properties Editor.

After you create a cursor cast member, use script to switch to the cursor in a movie, as shown in the next section.

Using an animated color cursor in a movie

After you add an animated color cursor to the cast, use script to switch to the animated cursor as you would any other cursor. You can set up an animated cursor as the movie's cursor or a sprite's cursor.

To switch to an animated color cursor, use the following method:

```
cursor (member whichCursorCastMember)
```

For `whichCursorCastMember`, substitute a cast member name (surrounded by quotation marks) or a cast member number. For example, the following sprite script changes the cursor to the cast member named myCursor when the cursor is over the sprite:

```
on mouseEnter
  cursor (member "myCursor")
end
```

To reset the cursor to the regular arrow cursor, specify a cursor type of -1 and do not use parentheses. The following sample sprite script resets the cursor:

```
on mouseLeave
  cursor -1
end
```

Note: Do not place an animated color cursor cast member on the Stage.

For more information, see the Scripting Reference topics in the Director Help Panel.

CHAPTER 14
3D Basics

Macromedia Director MX 2004 lets you bring robust, high-performance 3D graphics to the web. With Director, you can develop a wide spectrum of 3D productions, ranging from simple text handling to interactive product demonstrations to complete immersive game environments. Using Macromedia Shockwave Player, users can view your work on the web with Netscape Navigator, Microsoft Internet Explorer, or other Shockwave-supported browsers.

Director lets you detect the capabilities of the user's system and adjust playback demands accordingly. A powerful computer with 3D hardware acceleration brings the best results, but users can successfully use Director movies with 3D on most Macintosh or Windows hardware platforms. The faster the computer's graphics processing, the better the results. The ability to adjust for client-side processing power makes Director ideal for web delivery.

What you need to know

You can perform many basic 3D operations by using the built-in 3D behaviors in Director.For more information, see "Using 3D behaviors" on page 327. Most complex 3D operations, however, are performed by using either Lingo or JavaScript syntax, the built-in scripting languages of Director. The 3D documentation assumes that you understand either Lingo or JavaScript syntax. If you have not yet learned Lingo or JavaScript syntax, see the Scripting Reference topics in the Director Help Panel, which list all of the Lingo and JavaScript syntax methods and properties that are available in Director. The Scripting Reference topics in the Director Help Panel describe each expression, illustrate its syntax, and provide examples.

Because 3D is primarily controlled by behaviors and scripts, the 3D methods and properties are described here in detail. You'll find them grouped by category in Chapter 15, "The 3D Cast Member, 3D Text, and 3D Behaviors," on page 315, Chapter 16, "Working with Models and Model Resources," on page 335, and Chapter 17, "Controlling the 3D World," on page 393.

About 3D in Director MX 2004

The following are the main components that are common between Director MX 2004 and earlier versions of Director:

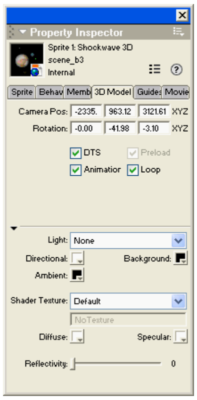
- The Stage is the authoring area in which the Director movie is assembled.
- The Score displays the arrangement of channels that organize, display, and control the movie over time.

  Because 3D is primarily script-controlled, it involves much less direct manipulation in the Score than other Director features.
- The Cast window is where all cast members, including the 3D cast members, are stored.

  Cast members are the media in your movies, such as sounds, text, graphics, and 3D scenes.
- Sprites are instances of cast members that appear on the Stage with individual properties and attributes.

  A sprite of a 3D cast member displays a particular camera's view into the 3D world. The 3D cast member contains models, which are individual objects inside the 3D cast member. For more information about models, see "The 3D world" on page 313. Also see "About the 3D cast member" on page 315 and Chapter 16, "Working with Models and Model Resources," on page 335.
- The Property inspector is a tabbed panel that lets you view and control properties of multiple objects in your movie.

  The Property inspector has been modified to include a 3D Model tab. See "Using the Property inspector for 3D" on page 308.



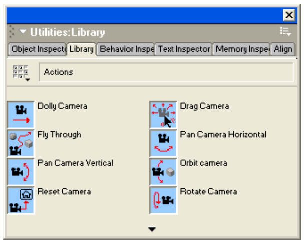
Sprite 1: Shockwave 3D
scene_b3
Internal

Camera Pos: -2335. 963.12 3121.61 XYZ
Rotation: -0.00 -41.98 -3.10 XYZ

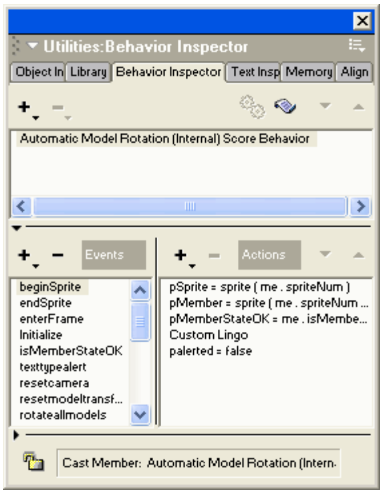
☑ DTS ☑ Preload
☑ Animation ☑ Loop

Light: None
Directional:
Background:
Ambient:
Shader Texture: Default
NoTexture
Diffuse:
Specular:
Reflectivity: 0

- The Behavior library lets you select the behaviors you want to use.



- The Behavior inspector lets you create and modify behaviors.



Automatic Model Rotation (Internal) Score Behavior

beginSprite
endSprite
enterFrame
Initialize
isMemberStateOK
texttypealert
resetcamera
resetmodeltransf...
rotateallmodels

pSprite = sprite (me . spriteNum)
pMember = sprite (me . spriteNum ...
pMemberStateOK = me . isMembe...
Custom Lingo
palerted = false

Cast Member: Automatic Model Rotation (Intern.

  For an introduction to the Behavior inspector, see "3D behaviors" on page 312. For a full discussion, see "Using 3D behaviors" on page 327.

- Director provides easy but powerful 3D text handling.

  For more information, see "Creating 3D text" on page 324.

- Lingo and JavaScript syntax are the Director scripting languages. They can be used to create movies that are more complex and interactive.

  For detailed Lingo and JavaScript syntax information, see "About the 3D cast member" on page 315, Chapter 16, "Working with Models and Model Resources," on page 335, "About lights and cameras" on page 382, and Chapter 17, "Controlling the 3D World," on page 393.

  3D methods and properties are covered according to function in each of these sources. They are also presented in dictionary form, with syntax guidelines, definitions, and examples, in the Scripting Reference topics in the Director Help Panel.

The 3D Xtra

The 3D Xtra lets you include 3D models in a Director movie. You can import 3D models or worlds created with a 3D modeling program and use Director to deliver them on the web. You can also combine the abilities of Director and your 3D modeling software by building a 3D world in your modeling program and adding to it or modifying it in Director.

To use 3D images and text created in third-party rendering software, you must convert the file to the W3D (Web 3D) format, which Director supports. Typically, each rendering application requires its own specific file converter to create W3D files. For more information about creating W3D files, see the documentation for your 3D modeling software.

Using the Shockwave 3D window

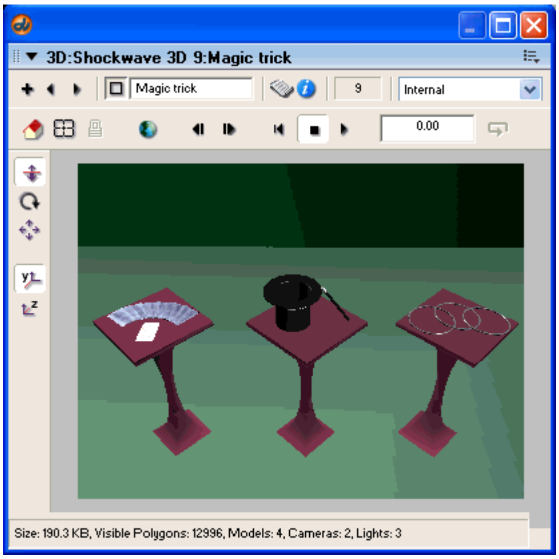
The Shockwave 3D window provides an easy way for you to inspect a 3D cast member. Some properties of 3D cast members can also be edited in this window.

To use the Shockwave 3D window:

1 Select a 3D cast member in the cast.

2 Click the Shockwave 3D Window button on the Director toolbar.

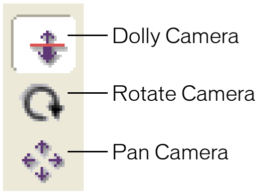
The Shockwave 3D window appears, displaying the 3D cast member currently selected in the cast.



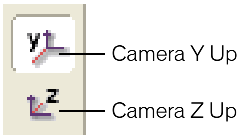
Size: 190.3 KB, Visible Polygons: 12996, Models: 4, Cameras: 2, Lights: 3

3 Use the following controls:

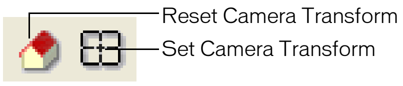
- The camera buttons along the side—Dolly, Rotate, and Pan—let you change your viewing angle by zooming in and out, moving around the world origin, and moving in a straight line horizontally or vertically, respectively. Hold the Shift key while using these tools to make the camera move faster.



- The two buttons below the camera buttons let you control whether the *y*-axis or the *z*-axis is the up axis when using the Camera Rotate tool.



- The playback buttons let you either play the cast member's animation at normal speed or step through the animation, forward or backward, controlling the movement with mouse clicks.
- The Loop button lets you play animations within the 3D cast member repeatedly.
- The Set Camera Transform and Reset Camera Transform buttons let you set and undo the changes you make to camera angles for the member's default camera. Set Camera Transform remembers the current camera position. Reset Camera Transform restores the camera to the previously remembered position.



- The Root Lock button fixes an animation in place, so that it doesn't change its position on the Stage while playing.

- The field at the top of the Shockwave 3D window shows the name of the cast member on display. The square button to the left of the text box lets you drag that cast member to the Stage or the Score.
- The New Cast Member, Previous Cast Member, and Next Cast Member buttons at the upper left of the Shockwave 3D window let you add or display 3D cast members.
- The Reset World button restores the 3D scene to its original state, with all models, cameras, and so on assuming their original positions.

Using the Property inspector for 3D

With the Property inspector, you can modify the 3D cast member without using script. The 3D Model tab of the Property inspector offers a simple way to view and control numerous aspects of the 3D world.

To view the 3D Model tab:

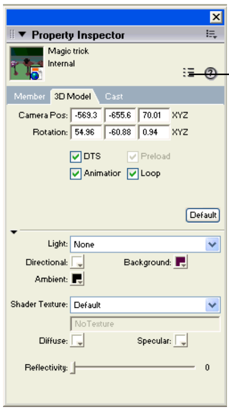
1 Open the Cast window if it isn't already open.

2 Click the 3D cast member you want to select.

3 Click the Property inspector button in the toolbar. The Property inspector opens.

4 Click the 3D Model tab in the Property inspector.

The Property inspector should appear in Graphical view. If the Property inspector is in List view, click the List View Mode icon to toggle the view to Graphical.



Camera Pos: -569.3 -655.6 70.01 XYZ
Rotation: 54.96 -60.88 0.94 XYZ
✓ DTS ✓ Preload
✓ Animatior ✓ Loop
Light: None
Directional:
Background:
Ambient:
Shader Texture: Default
NoTexture
Diffuse:
Specular:
Reflectivity: 0

The Property inspector's 3D Model tab provides several options:

- The text boxes at the top of the tab show the initial position and orientation of the default camera. The default (0, 0, 0) represents a vantage point looking up the *z*-axis through the middle of the screen. The values you enter in these text boxes replace the displayed values and move the camera.
- The Direct to Stage (DTS) option controls whether rendering occurs directly on the Stage (the default) or in the Director offscreen buffer. The offscreen graphics buffer is where Director calculates which sprites are partly hidden behind other sprites. When Direct to Stage is on, Director bypasses its offscreen buffer and saves time, increasing playback speed. When Direct to Stage is on, however, you can't use the inker modifier on 3D models or place other sprites on top of the 3D sprite.
- The Preload option controls how media that's being downloaded to the user's computer is displayed. The media can be held back from display until it has been completely streamed into memory, or it can be displayed progressively on the Stage as data becomes available.
- The Play Animation option controls whether any existing animation, either bones or keyframe, is played or ignored.
- The Loop option controls whether the animation loops continuously or plays once and stops.

- The Director Light area lets you select one of ten lighting positions to apply to a single directional light. You can also adjust the color for the ambient light. (Directional light comes from a particular, recognizable direction; ambient light is diffuse light that illuminates the entire scene.) Finally, you can adjust the background color of the scene.
- The Shader Texture area lets you work with shaders and textures. A shader determines the method used to render the surface of a model; a texture is an image that is applied to the shader and drawn on the surface of the model. All new models use the default shader until you apply a different shader. Using the Property inspector, you can assign a texture to the default shader. You can also control the default shader's specular (highlight) color, its diffuse (overall) color, and its reflectivity. For more information, see "The 3D world" on page 313 and Chapter 16, "Working with Models and Model Resources," on page 335.

Using rendering methods

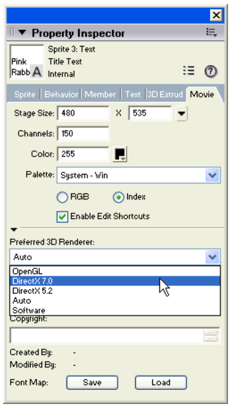
The rendering method refers to the specific way Director displays 3D images on the Stage. The methods available depend on the type of hardware you have. The rendering methods include the following:

- `#auto`: Director selects the best method based on the client computer's specific hardware and drivers.
- `#openGL`: OpenGL drivers for a 3D hardware accelerator are used. OpenGL is available for the Macintosh and Windows platforms.
- `#directX7_0`: DirectX7_0 drivers for a 3D hardware accelerator are used. This option is available for Windows only.
- `#directX5_2`: DirectX5_2 drivers for a 3D hardware accelerator are used. This option is available for Windows only.
- `#software`: The Director built-in software renderer is used. This option is available on the Macintosh and Windows platforms.

The rendering method can have a dramatic effect on performance. If your hardware permits you to select different methods, you can do so using the following procedure.

To choose a rendering method:

1 Select the Stage.
2 Open the Property inspector.
3 Click the Movie tab.
4 Select a rendering method from the pop-up menu.



Stage Size: 480 X 535
Channels: 150
Color: 255
Palette: System - Win
( ) RGB (●) Index
[✓] Enable Edit Shortcuts
Preferred 3D Renderer: Auto
OpenGL
DirectX 7.0
DirectX 5.2
Auto
Software
Copyright:
Created By: -
Modified By: -
Font Map:

If you don't select a rendering method, Director defaults to `#auto`.

Below the pop-up menu, the name of the active 3D renderer property appears. The value of this property indicates which rendering method is currently being used. This is especially useful when you want to know which renderer is active while you have `#auto` selected.

Using 3D Anti-aliasing

Director gives you the ability to use anti-aliasing with 3D cast members in your movies. Anti-aliasing improves the appearance of graphics by smoothing the lines between shapes or areas of different color so that the lines do not appear jagged. When you use anti-aliasing with a 3D sprite, the edges of each model in the sprite appear smoother against each other and against the background. Anti-aliasing of 3D sprites is particularly well-suited for merchandise demos and other e-commerce applications because its image quality is high and it can be turned on and off, as needed, in real time.

Effects of anti-aliasing

An anti-aliased 3D sprite uses more processor power and memory than one that is not anti-aliased, resulting in lower frame rates. Because of this, it is recommended that you turn off anti-aliasing for 3D sprites while any part of the sprite is being moved or animated and turn it back on when the animation is complete. Movies that are designed to animate quickly, such as games, might work better with anti-aliasing turned off. During authoring, movies that use anti-aliasing continue to draw heavily on the processor, even after the movie is stopped. You might want to turn off anti-aliasing each time you stop your movie to ensure that the performance of Director is not affected.

Determining whether anti-aliasing is supported

Not all 3D renderers can perform the additional calculations that anti-aliasing requires. If you have a 3D sprite that you want to anti-alias, check first that the 3D renderer supports anti-aliasing. The renderers that currently support anti-aliasing include the Director software renderer, and DirectX 5.2 and DirectX 7.0.

If the 3D sprite is in channel 1 of the Score, you would test the `antiAliasingSupported` property of sprite 1, as shown in the following example:

```
if sprite(1).antiAliasingSupported = TRUE then
```

Turning on anti-aliasing

If the `antiAliasingSupported` property is `TRUE`, you can turn on anti-aliasing for the 3D sprite by setting the sprite's `antiAliasingEnabled` property to `TRUE`.

```
sprite(1).antiAliasingEnabled = TRUE
```

For example, if you have a 3D sprite in channel 5 and you want to turn on anti-aliasing for the sprite when it first appears on the Stage, you would write a `beginSprite` script and attach it to the sprite. Your script should contain code as shown in the following example:

```
-- Lingo syntax
on beginSprite
  -- check whether anti-aliasing is supported by the current 3D renderer
  if sprite(5).antiAliasingSupported = TRUE then
    -- if it is, turn on anti-aliasing for the sprite
    sprite(5).antiAliasingEnabled = TRUE
  end if
end beginSprite
// JavaScript syntax
function beginSprite() {
  // check whether anti-aliasing is supported by the current 3D renderer
  if (sprite(5).antiAliasingSupported) {
    // if it is, turn on anti-aliasing for the sprite
    sprite(5).antiAliasingEnabled = true;
  }
}
```

Turning off anti-aliasing

If you plan to animate any part of a 3D sprite, you might want to turn anti-aliasing off temporarily to improve the animation performance. To do this, set the `antiAliasingEnabled` property for the sprite to `FALSE`. You can then set it back to `TRUE` when the animation is complete.

It is a good idea to turn anti-aliasing on and off on separate handlers. For example, you might want to animate a model, camera, or light while the mouse button is held down and stop the animation when the mouse button is released. In that case you would turn off anti-aliasing in a `mouseDown` handler and turn it back on in a `mouseUp` handler, as shown in the following example:

```
-- Lingo syntax
on mouseDown
  -- user interaction/animation is about to start so turn
  -- anti-aliasing OFF
  sprite(1).antiAliasingEnabled = FALSE

  -- start animation
```

```
end

on mouseUp
  -- stop animation

  -- the interaction/animation has ended so turn
  -- anti-aliasing ON
  sprite(1).antiAliasingEnabled = TRUE
end
// JavaScript syntax
function mouseDown() {
  // user interaction/animation is about to start so turn
  // anti-aliasing OFF
  sprite(1).antiAliasingEnabled = false;

  //start animation
}

function mouseUp() {
  // stop animation

  // the interaction/animation has ended so turn
  // anti-aliasing ON
  sprite(1).antiAliasingEnabled = true;
}
```

3D behaviors

The Director Behavior library includes 3D-specific behaviors. 3D behaviors are divided into four types:

- Local behaviors are actions that accept triggers only from the sprite to which they're attached.
- Public behaviors are actions that accept triggers from any sprite.
- Triggers are behaviors that send signals to a local or public behavior to cause the behavior to execute.

  For example, attaching the Create Box action and Mouse Left trigger behaviors to a sprite will cause a box to be created in the 3D world each time the sprite is clicked with the left mouse button.
- Independent behaviors are behaviors that perform their actions without a trigger.

  The Toon behavior, for example, changes a model's rendering style to the toon style.

3D text

You can easily create 3D text in Director by performing the following steps.

To create 3D text:

1 Create a normal (2D) text cast member.

2 Convert the text to 3D by selecting 3D Mode from the Display pop-up menu on the Text tab of the Property inspector.

3 Set properties of the 3D text using the 3D Text tab to manipulate the specific properties of the 3D text.

You can also manipulate the text cast member with script or a behavior. For more information, see "Creating 3D text" on page 324.

The 3D world

This section provides a brief overview of the contents of 3D cast members. For more detailed information, see "About the 3D cast member" on page 315.

Each 3D cast member contains a complete 3D world. It can contain models (the objects that viewers see within the world) that are illuminated by lights and viewed by cameras. A sprite of a 3D cast member represents a specific camera's view into the world. Imagine that the 3D cast member is a room filled with furniture with cameras pointing in from several windows. A given sprite using that cast member will display the view from one of those cameras, but the room itself—the 3D cast member—remains the same regardless of which view is used.

The key difference between 3D cast members and other cast members is that the models within the 3D world are not independent entities—they're not sprites. They are integral parts of the 3D cast member.

Your movies can use 2D and 3D cast members simultaneously. For example, a product demonstration movie might consist of a 3D cast member that represents the product and one or more 2D controls that allow users a virtual tryout of the product.

Models and model resources

Models are the objects that users see within the 3D world. Model resources are elements of 3D geometry that can be used to draw 3D models. A model is a visible object that makes use of a model resource and occupies a specific position and orientation within the 3D world. The model also defines the appearance of the model resource, such as what textures and shaders are used. For more information, see Chapter 16, "Working with Models and Model Resources," on page 335.

The relationship between a model and a model resource is similar to that between a sprite and a cast member. Model resource data can be reused because multiple models can use the same model resource in the same way as cast member data can be reused by multiple sprites. Unlike sprites, however, models don't appear in and can't be controlled from the Score.

For example, a 3D cast member might contain two model resources. One could be the geometry for a car body, and the other could be the geometry for a car wheel. In order for a complete car to appear visibly in the 3D scene, the model resource for the car body would be used once, and the model resource for the wheel would be used four times—once for each wheel.

Each 3D cast member contains a group object called *world*, which may contain a tree-like parent-child hierarchy of nodes, such as models, groups, lights, and cameras. Each node may have one parent and any number of children. Nodes that have world as an ancestor are rendered. A cast member may also contain nodes that do not have world as an ancestor, such as nodes with a `parent` property set to VOID; such nodes are not rendered.

The primary benefit of these parent-child relationships is that they make it easier to move complex models around in the 3D world and to have the component parts of those models move together in the proper way. In the example of the car previously described, if the wheels of the car are defined as children of the car model, then moving the car will cause the wheels to be moved with the car in the expected manner. If no parent-child relationship is defined between the car and the wheels, moving only the car will cause the wheels to be left behind in their original position in the world.

Lights and cameras

Lights are used to light the 3D world. Without lights, the objects within the world can't be seen.

Although lights control the appearance of the 3D world, cameras control how a sprite views the 3D world. A 3D sprite displays a particular camera's view into the world. For more information, see "About lights and cameras" on page 382.

Groups

A group is a node that has no geometry, used for grouping models, lights, and cameras together so they can be treated as a single unit. Groups let you rotate or translate their contents as a unit. A group has a name, a transform, and a parent, and it can have one or more children. It has no other properties. The highest level group is the group called *world*, which is essentially synonymous with the 3D cast member. For more information, see "Groups" on page 323.

Shaders and textures

A model's surface color is determined by its shader or shaders. You can draw images on the surface of a model by applying one or more textures to each shader. For more information, see "Shaders" on page 352 and "Textures" on page 360.

Modifiers

Modifiers let you control many aspects of how models are rendered and how they behave. When you attach a modifier to a model, you can then set the properties for that modifier with script. Depending on the type of modifier you use, setting its properties can give you fine control over the model's appearance and behavior. Modifiers are described in "Working with Models and Model Resources" on page 335 and "About lights and cameras" on page 382.

Animation

Director supports complex model animations through the following means:

- The collision modifier lets models detect and respond appropriately to collisions.
- The Bones player modifier lets models that have a skeletal structure defined in them play back animations of those skeletons. These animations are created in separate 3D modeling tools.
- The Keyframe player modifier lets models play back time-based animation sequences. These sequences can also be created in separate 3D modeling tools.

3D animations are called motions and can be initiated by 3D behaviors or script. For more information, see "Animation modifiers" on page 372 and "Motions" on page 381.

CHAPTER 15
The 3D Cast Member, 3D Text, and 3D Behaviors

Several Macromedia Director MX 2004 features let you create a 3D movie:

- A 3D cast member contains a complex internal structure that includes model resources, models, lights, and cameras. Each of these objects has its own array of properties.
- Director lets you convert 2D text to 3D and then work with the 3D text as you would with any other 3D cast member. You can apply behaviors to the 3D text, manipulate it with Lingo or JavaScript syntax, and view and edit it in the Macromedia Shockwave 3D window. You can also add extruded 3D text to a 3D cast member.
- Director comes with a library of behaviors that let you build and control a 3D environment without any knowledge of Lingo or JavaScript syntax. Although scripting is still required for complex projects, you can build simple 3D movies with behaviors alone.

About the 3D cast member

Each Director 3D cast member contains an entire 3D world. Each world is a collection of one or more models and other objects. These objects include the following:

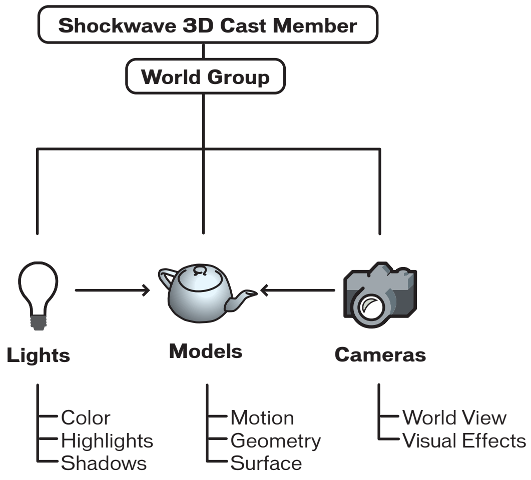
- Model resources are elements of 3D geometry used to render models. The same model resource can be used by several models in the 3D world.
- Models are visible objects in the 3D cast member that use a model resource geometry. For more information about models, see Chapter 16, "Working with Models and Model Resources," on page 335.
- Shaders are methods of displaying the surface of a model. Shaders control how the surface of the model reflects light, and whether the surface looks like metal, plaster, or other materials.
- Textures are simple 2D images that are drawn onto the surface of a 3D model. A model's surface appearance is the result of the combination of its shader and any textures that have been applied to it.
- Motions are predefined animation sequences that involve the movement of models or model components. Individual motions can be set to play by themselves or with other motions. For example, a running motion can be combined with a jumping motion to simulate a person jumping over a puddle.
- Lights are sources of illumination within the 3D world. Lights can be directional, such as a spotlight, or they can be diffuse.

- Cameras are views into the 3D cast member. Each sprite that uses a given cast member can display the view from a different camera. For more information about sprites, see Chapter 14, “3D Basics,” on page 303.
- Groups are clusters of models, lights, and/or cameras that are associated with one another. This makes moving the associated items much easier; rather than moving each item separately, you can write Lingo or JavaScript syntax that moves the group with a single command.

Each model, light, camera, and group within a 3D cast member is referred to as a *node*. Nodes can be arranged in parent-child hierarchies. When a parent moves, its children move with it. For example, a car wheel can be a child of a car body. These parent-child relationships are established in your third-party 3D modeling software or with script.

The following figures show the relationships between cameras, lights, and models within the 3D cast member as well as the relation of a model to a model resource and of a model to shaders, textures, and motion.

Note: A member can also contain lights, models, groups, and cameras that have no parent and are therefore not rendered.



Although the elements of a 3D scene can be modified and manipulated with 3D behaviors, complex work requires scripting. For more information about behaviors, see “3D behaviors” on page 312. The following section describes the methods and properties that can manipulate each type of node in a 3D cast member. For another view of this material, with examples of the use of individual methods and properties, see the Scripting Reference topics in the Director Help Panel.

Model resources

Model resources are pieces of 3D geometry that can be used to display 3D models. Model resources are visible only when they are used by a model. Model resources are reusable, and multiple models can share the same model resource.

The following cast member methods and properties perform basic model resource operations:

Method	Function	Returns
`modelResource.count`	Returns the number of model resource objects included in the cast member.	Integer.
`modelResource(`*`name`*`)`	Returns the model resource named *`name`*.	Returns the model resource object named *`name`*, if it exists. Returns `void` if the object does not exist.
`modelResource[`*`index`*`]`	Returns the model resource at the designated position in the index. The index number can change if model resources are added or deleted.	Returns the model resource object at that index number if it exists. Returns `void` if the object does not exist at that index number.
`newMesh(`*`name, numFaces, numVertices, numNormals numColors, numTexture Coordinates`*`)`	Creates a new mesh model resource. *`numFaces`* is the user-specified number of triangles. *`numVertices`* is the user-specified number of vertices. A vertex can be used by more than one face. *`numNormals`* is the user-specified number of normals. Enter 0 or omit this step to use the `generateNormals()` method. *`numColors`* is the user-specified number of colors. You can specify a color for each point of a triangle. *`numTextureCoordinates`* is the number of user-specified texture coordinates. Enter 0 or omit this step to get the default coordinates.	Returns a new mesh model resource with a unique name. If the name isn't unique, returns a script error.
`newModel Resource(`*`name, type`*`)`	Creates a new model resource and adds it to the model resource object list. The *`type`* can be `#plane`, `#box`, `#sphere`, `#cylinder`, `#extrusion`, or `#particle`.	Returns a new model resource object with a unique name. If the name isn't unique, returns a script error.
`newModel Resource(`*`name, type, facing`*`)`	Creates a new model resource with the specified facing and adds it to the model resource object list. The *`type`* can be `#plane`, `#box`, `#sphere`, or `#cylinder`. The *`facing`* can be `#front`, `#back`, or `#both`.	Returns a new model resource object with a unique name. If the name isn't unique, returns a script error.
`deleteModel Resource(`*`name`*`)`	Deletes the model resource named *`name`*. Script references to this model resource persist but can do nothing.	`TRUE (1)` if the model resource named *`name`* existed and was successfully deleted. `FALSE (0)` if the model resource named *`name`* doesn't exist.
`deletemodel Resource(`*`index`*`)`	Deletes the model resource with the given index number. Script references to this model resource persist but can do nothing.	`TRUE (1)` if the model resource with this index number exists. `FALSE (0)` if the model resource with this index number doesn't exist.

Models

The models in a cast member are the actual visible objects that are seen in the 3D cast member. Models move according to the motions you assign to them in your 3D modeling software, and according to animation scripts that you write. Their movement results from repositioning and reorienting their geometries in 3D space.

The following cast member methods and properties can be used to perform basic model operations:

Method	Function	Returns
`model.count`	Returns the number of model objects included in the cast member.	Integer.
`model(`*`name`*`)`	Returns the model named *`name`*.	Returns the model object named *`name`* if it exists. Returns `void` if the object does not exist.
`model[`*`index`*`]`	Returns the model at the designated index position. The index number can change if models lower in the list are deleted.	Returns the model object at that index number if it exists. Returns `void` if the object does not exist at that index number.
`newModel(`*`name, modelResource`*`)`	Creates a new model named *`name`* and adds it to the world. Fails if a model by that name already exists. The *`modelResource`* argument is optional and can be set at a later time. If supplied, this second argument must be an existing model resource object.	Returns a new model with a unique name. If the name isn't unique, returns an error.
`deleteModel (`*`name`*`)`	Deletes the model named *`name`*. Script references to this model persist but can do nothing. Children of the model aren't deleted but are *re-parented* to the world group.	`TRUE (1)` if the model named *`name`* exists. `FALSE (0)` if the model named *`name`* doesn't exist.
`deleteModel (`*`index`*`)`	Deletes the model with the given index number. Script references to this model persist but can do nothing.	`TRUE (1)` if the model with this index number exists. `FALSE (0)` if the model with this index number doesn't exist.
`model(`*`name1`*`)`.clone`(`*`name2`*`)`	Creates a copy of the model named *`name1`* and assigns it the name *`name2`*. The new model uses the same model resource and shader as the original model; changes to the model resource and shader of the original model appear in the new model as well.	Returns the model named *name2*.

Method	Function	Returns
`model(`*`name1`*`).` clone`Deep(`*`name2`*`)`	Creates a copy of the model named *`name1`* and assigns it the name *`name2`*. The new model uses a copy of the original model's model resource and shader; changes to the original model's model resource and shader have no effect on the new model.	Returns the model named *name2*.
cloneModelFromCast Member`(`*`name1`*, *`name2`*, member(*`name3`*)`)`	Copies the model named *name2* (from the member named *name3*) into the current member. The new copy is named *name1*. This is similar to the `cloneDeep()` method, but it copies a model from one cast member to another.	Returns the model named *name1*.

Shaders

A shader defines the basic appearance of a model's surface. You apply textures to shaders. The standard shader is photorealistic. The following list describes some of the other available shaders:

- `#painter`, which looks like a painted surface
- `#engraver`, which looks like an engraved surface
- `#newsprint`, which looks like a newspaper photograph

The following cast member methods and properties can be used to perform basic shader operations:

Method	Function	Returns
`shader.count`	Returns the number of shader objects included in the cast member.	Integer.
`shader(`*`name`*`)`	Returns the shader named *`name`*.	Returns the shader object named *`name`* if it exists. Returns `void` if the object does not exist.
`shader[`*`index`*`]`	Returns the shader at the designated position in the index. The index number can change if shaders are added or deleted.	Returns the shader object at that index number if it exists. Returns `void` if the object does not exist at that index number.
`newShader (`*`name,type`*`)`	Creates a new shader and adds it to the shader object list. The *`type`* can be `#standard`, `#painter`, `#engraver`, or `#newsprint`.	Returns a new shader object with a unique name. If the name isn't unique, returns a script error.

Method	Function	Returns
`deleteShader (name)`	Deletes the shader named *name*. Script references to this shader persist but can do nothing.	`TRUE (1)` if the shader named *name* exists. `FALSE (0)` if the shader named *name* doesn't exist.
`deleteShader (index)`	Deletes the shader with the given index number. Script references to this shader persist but can do nothing.	`TRUE (1)` if the shader with this index number exists. `FALSE (0)` if the shader with this index number doesn't exist.

Textures

Textures are 2D images that are drawn on the surface of a 3D model. Textures can be assigned to models in your 3D modeling software, or you can use any bitmap cast member in your movie. You can also use any image object created using Lingo or JavaScript.

The following cast member methods and properties can be used to perform basic texture operations:

Method	Function	Returns
`texture.count`	Returns the number of textures in the texture object list of the cast member.	Integer.
`texture(name)`	Returns the texture object named *name*.	Returns the texture object named *name* if it exists. Returns `void` if the object does not exist.
`texture[index]`	Returns the texture at the designated position in the index. The index number can change if textures are added or deleted.	Returns the texture object at that index number if it exists. Returns `void` if the object does not exist at that index number.

`newTexture (name, type, source)`	Creates a new texture named *name*.
	The *type* can have the following values:
	`#fromCastmember`
	`#fromImageObject`
	If *type* is `#from Castmember`, *source* is a cast member reference. For example, `member("concrete") or member[2,3]`
	If *type* is `#from ImageObject`, *source* is a script image object.
		Returns a new texture object with a unique name. If the name isn't unique, returns a script error.

`deleteTexture (name)`	Deletes the texture named *name*. Script references to this texture persist but can do nothing.	`TRUE (1)` if the texture named *name* exists. `FALSE (0)` if the texture named *name* doesn't exist.
`deleteTexture (index)`	Deletes the texture with the given index number. Script references to this texture persist but can do nothing.	`TRUE (1)` if the texture with this index number exists. `FALSE (0)` if the texture with this index number doesn't exist.

Motions

A motion is an animation of a model. Motions can be shared by multiple models. A 3D cast member contains a palette of motions that are available to any model in the world.

The following cast member methods and properties can be used to perform basic motion operations:

Method	Function	Returns
`motion.count`	Returns the number of motion objects included in the cast member.	Integer.
`motion(`*`name`*`)`	Returns the motion named *`name`*.	Returns the motion object named *`name`* if it exists. Returns `void` if the object does not exist.
`motion[`*`index`*`]`	Returns the motion at the designated position in the palette of available motions.	Returns the motion object at that index number if it exists. Returns `void` if the object does not exist at that index number.
`newMotion (`*`name`*`)`	Creates a new motion object.	Returns a new motion object with a unique name. If the name isn't unique, returns a script error.
`deleteMotion (`*`name`*`)`	Deletes the motion named *`name`*. Script references to this motion persist but return `void`.	`TRUE (1)` if the motion named *`name`* exists. `FALSE (0)` if the motion named *`name`* doesn't exist.
`deleteMotion (`*`index`*`)`	Deletes the motion at the given index. Script references to this motion persist but return `void`.	`TRUE (1)` if the motion with this index number exists. `FALSE (0)` if the motion with this index number doesn't exist.
cloneMotionFrom CastMember`(`*`name1`*, *`name2`*, member(*`name3`*)`)`	Copies a motion object from one cast member to another. Specifically, copies the motion object named *name2* (from the member named *name3*) into the current member. The new copy is named *name1*.	Returns a reference to the newly created motion object.

Lights

Models in the 3D world are illuminated by lights. Each light has a color, direction, intensity, and other characteristics. By default, each 3D cast member contains one white light, which lets Director users see the models in the cast member without having to explicitly add a light. This light has a default position of upper center in the world. You can modify or replace this light with one or more new lights. To turn off the default light, set its `color` property to `color(0,0,0)`.

The following methods and properties can be used to perform basic light operations:

Method	Function	Returns
`light.count`	Returns the number of light objects included in the cast member.	Integer.
`light(`*`name`*`)`	Returns the light named *`name`*.	Returns the light object named *`name`* if it exists. Returns `void` if the object does not exist.
`light[`*`index`*`]`	Returns the light at the designated position in the index. The index number can change if lights are added or deleted.	Returns the light object at that index number if it exists. Returns `void` if the object does not exist at that index number.
`newLight(`*`name, type`*`)`	Creates a new light and adds it to the light object list. The *`type`* can be `#ambient`, `#directional`, `#point`, or `#spot`.	Returns a new light object with a unique name. If the name isn't unique, returns a script error.
`deleteLight (`*`name`*`)`	Deletes the light named *`name`*. Script references to this light persist but can do nothing.	`TRUE (1)` if the light named *`name`* exists. `FALSE (0)` if the light named *`name`* doesn't exist.
`deleteLight (`*`index`*`)`	Deletes the light with the given index number. Script references to this light persist but can do nothing.	`TRUE (1)` if the light with this index number exists. `FALSE (0)` if the light with this index number doesn't exist.

Cameras

Cameras provide different views of the 3D world. A 3D cast member can have many cameras. Each sprite that uses the cast member can display a different camera view of the 3D world.

The following cast member methods and properties can be used to perform basic camera operations:

Method	Function	Returns
`camera.count`	Returns the number of camera objects included in the cast member.	Integer.
`camera(`*`name`*`)`	Returns the camera named *`name`*.	Returns the camera object named *`name`* if it exists. Returns `void` if the object does not exist.
`camera[`*`index`*`]`	Returns the camera at the designated position in the index. The index number can change if cameras are added or deleted.	Returns the camera object at that index number if it exists. Returns `void` if the object does not exist at that index number.
`newCamera (`*`name`*`)`	Creates a new camera and adds it to the camera object list.	Returns a new camera object with a unique name. If the name isn't unique, returns a script error.

Method	Function	Returns
`deleteCamera` (*`name`*)	Deletes the camera named *`name`*. Script references to this camera persist but can do nothing.	`TRUE (1)` if the camera named *`name`* exists. `FALSE (0)` if the camera named *`name`* doesn't exist.
`deleteCamera` `(`*`index`*`)`	Deletes the camera with the given index number. Script references to this camera persist but can do nothing.	`TRUE (1)` if the camera with this index number exists. `FALSE (0)` if the camera with this index number doesn't exist.

Groups

Groups are collections of models and other objects that are formally associated with one another. These associations can be created in your 3D modeling software or with script. Each 3D cast member has a default group called world, which is the cast member.

Groups simplify the rotation and translation of models by letting all members of a group move together with a single command. A group has a name, a transform, and a parent, and can also have children. It has no other properties.

The following cast member methods and properties can be used to perform basic group operations:

Method	Function	Returns
`group.count`	Returns the number of group objects included in the cast member.	Integer.
`group(`*`name`*`)`	Returns the group named *`name`*.	Returns the group object named *`name`* if it exists. Returns `void` if the object does not exist.
`group[`*`index`*`]`	Returns the group at the designated position in the index. The index number can change if groups are added or deleted.	Returns the group object at that index number if it exists. Returns `void` if the object does not exist at that index number.
`newGroup(`*`name`*`)`	Creates a new group and adds it to the group object list.	Returns a new group object with a unique name. If the name isn't unique, returns a script error.
`deleteGroup` `(`*`name`*`)`	Deletes the group named *`name`*. Script references to this group persist but can do nothing.	`TRUE (1)` if the group named *`name`* exists. `FALSE (0)` if the group named *`name`* doesn't exist.
`deleteGroup` `[`*`index`*`]`	Deletes the group with the given index number. Script references to this group persist but can do nothing.	`TRUE (1)` if the group with this index number exists. `FALSE (0)` if the group with this index number doesn't exist.

Creating 3D text

To create 3D text, you first create 2D text, and then you convert the text to 3D.

Note: Alternatively, you can use the `extrude3d()` method of a 3D cast member to create an extruded-text model resource in the cast member.

To create 3D text:

1 Select Window > Text to open the text editor.

2 Select the desired font, size, style, and alignment.

Most standard fonts work well with 3D text. You might need to experiment to get specific results.

3 Enter the text. (You can edit the text after you've entered it.)

4 Drag the text cast member onto the Stage.

You can either drag the cast member from the Cast window or drag the Drag Cast Member button next to the Name text box in the Text window.

5 Click the Property Inspector button in the Director toolbar.

6 Click the Text tab in the Property inspector.

7 Select #mode3d Mode from the Display pop-up menu.

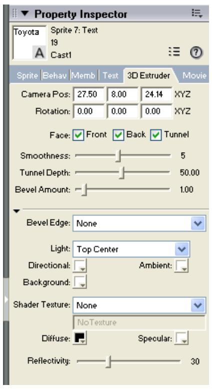
The text on the screen changes to 3D. You can now work with it, as discussed in the next section.

Modifying 3D text

After your 2D text has been changed to 3D, you can modify it.

To modify the 3D text:

1 Click the Property inspector's 3D Extruder tab.



Camera Pos: 27.50 8.00 24.14 XYZ
Rotation: 0.00 0.00 0.00 XYZ
Face: ✓ Front ✓ Back ✓ Tunnel
Smoothness: 5
Tunnel Depth: 50.00
Bevel Amount: 1.00
Bevel Edge: None
Light: Top Center
Directional:
Ambient:
Background:
Shader Texture: None
NoTexture
Diffuse:
Specular:
Reflectivity: 30

2 Set the camera position and rotation.

As with the standard 3D Property inspector tab, you control camera position and rotation with the values that you enter in the text boxes at the top of the pane. The default camera position represents a vantage point looking up through the middle of the scene.

Note: You might prefer to define these settings using the Shockwave 3D window instead of the Property inspector.

3 Select from among the Front Face, Back Face, and Tunnel check boxes.

   These options control which sides of the text appear.

4 Set the smoothness.

   This determines the number of polygons that are used to construct the text. The more polygons that are used, the smoother the text appears.

5 Set the tunnel depth.

   This is the length of the tunnel from the front face to the back face.

6 Select a beveled edge type.

   Beveling makes the edges of the 3D letters appear rounded or angled. Select Round for rounded edges, and Miter for angled edges.

7 Select a bevel amount.

   This determines the size of the bevel.

8 Set up the lighting.

   You can select a color and position for the text's default directional light. A directional light is a point source of light and comes from a specific, recognizable direction. You can also select a color for the ambient and background lights in the 3D world that the text occupies. Ambient light is diffuse light that illuminates the entire world; background light appears to come from behind the camera.

9 Apply a shader and a texture.

   Shaders and shader properties determine the appearance of the surface of the 3D text model. Textures are 2D images drawn on the surface of the text. Using the Property inspector, you can assign a texture to the text's shader. You can also control the color of the shader's specular highlights and its diffuse or overall color and reflectivity.

   As with any model, you can apply a texture that uses a bitmap cast member. You can import a bitmap cast member or create a new one in the Paint window. Be sure to give your bitmap cast member a name if doesn't already have one. To assign this bitmap as the texture, specify it in the Property inspector: Select Member from the Shader Texture menu, and enter the name of the member you want to use in the text box to the right of the menu.

Script and 3D text

Director contains methods and properties in Lingo and JavaScript syntax for working with 3D text. Most 3D methods and properties work with 3D text exactly as with any other object. For the methods and properties that don't work with 3D text, see the following section. For information about new 3D text properties that have been added, see "Script and 3D text". The new properties are also described in the Scripting Reference topics in the Director Help Panel.

Exceptions

The following methods and properties work differently when used with 3D text.

Type of Script	Element
Member property	`antiAlias`
Member property	`antiAliasThreshold`
Member property	`picture`
Member property	`preRender`
Member property	`scrollTop`
Member property	`useHypertextStyles`
Member property	`autoTab`
Member property	`boxType #scroll`
Member command	`scrollByPage`
Member command	`scrollByLine`
Member function	`charPosToLoc`
Member function	`linePosToLocV`
Member function	`locToCharPos`
Member function	`locVToLinePos`
Sprite property	`editable`
Sprite function	`pointInHyperLink`
Sprite function	`pointToChar`
Sprite function	`pointToItem`
Sprite function	`pointToLine`
Sprite function	`pointToParagraph`
Sprite function	`pointToWord`
Hypertext Lingo	`hyperlinkClicked`
Hypertext Lingo	`hyperlink`
Hypertext Lingo	`hyperlinks`
Hypertext Lingo	`hyperlinkRange`
Hypertext Lingo	`hyperlinkState`

Lingo and JavaScript syntax script for 3D text

In addition to working with most existing methods and properties, 3D text also adds some properties of its own. These properties give you a more precise way to define the characteristics of the text than is possible using the Property inspector.

These properties can be set while the text is in 2D mode. They have no visible effect until the text appears in 3D mode.

When you access the properties listed in the following table for an extruded 3D text model that you created by using the `extrude3D` method, you must refer to the model resource of the text. The Lingo syntax for this is shown in the following example:

```
member(whichMember).model[modelIndex].resource.3DTextProperty
```

For example, to set the `bevelDepth` property of the first model in cast member 1 to a value of 25, use the following syntax:

```
member(1).model[1].resource.bevelDepth = 25
```

Property	Access	Description	Range or Default
`bevelDepth`	Get and set	Degree of beveling on front or back faces.	Floating-point value from 1.0 to 100.0 Default is 1.0

`bevelType`	Get and set	Type of bevel.	`#none`
			`#miter`
			`#round`
			Default is `#miter`

`displayFace`	Get and set	Faces of shape to display.	`#front`
			`#tunnel`
			`#back`
			Default is to show all three faces

`displayMode`	Get and set	Specifies how the text appears.	`#modeNormal`
			`#Mode3D`
			Default is `#modeNormal`, which is 2D text

`member(1).extrude3d(member(2))`	Not applicable	Creates a new model resource in member 2 by extruding the text in member 1. Member 1 must be a text cast member.	Specify an existing 3D cast member
`smoothness`	Get and set	Number of subdivisions for curved outlines.	Integer from 1 to 100 Default is 5
`tunnelDepth`	Get and set	Extrusion depth.	Floating-point value from 1.0 to 100.0

Using 3D behaviors

Director includes a library of behaviors that lets you build and control a 3D environment without any knowledge of Lingo or JavaScript syntax. Although scripting is still required for complex projects, you can build simple 3D movies with behaviors alone.

Behavior types

Director provides two types of 3D behaviors: trigger and action. Action behaviors are divided into three types: local, public, and independent. These behavior types and subtypes are defined in the following table.

Type	Function
Trigger behavior	A behavior that sends an event, such as a mouse click, to an action behavior
Local action behavior	A behavior that is attached to a particular sprite and that can accept triggers only from that sprite
Public action behavior	A behavior that can be triggered by any sprite
Independent action behavior	A behavior that needs no trigger

If you're familiar with behaviors from earlier versions of Director, you'll recognize that the trigger/action distinction is new. In previous versions, the trigger instruction had to be included as a handler, such as `on mouseDown`, inside the behavior. The trigger behavior type makes it easier to reuse action behaviors in different ways with different triggers. These behaviors can be used with any 3D cast member.

Using the 3D Behavior Library

All 3D behaviors are listed in the Behavior Library. The Behavior Library is divided into two sublibraries: actions and triggers.

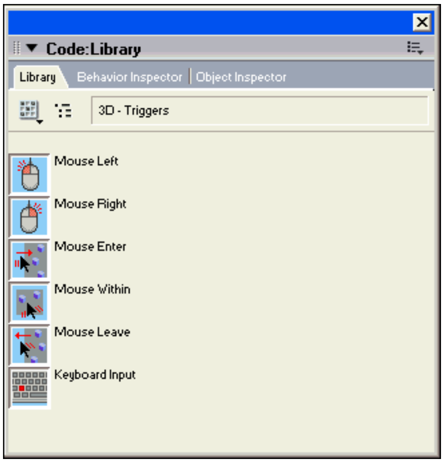
To view 3D trigger behaviors:

1 Click the Library Palette button on the Director toolbar.

2 Click the Library List button and select 3D.

3 Select Triggers from the 3D submenu.

The trigger behaviors appear, as shown in the following figure.



The following table describes the available triggers.

Name	Description
Mouse Left	Triggers action when the user presses, holds down, or releases the left mouse button (Windows) or the mouse button (Macintosh).
Mouse Right	Triggers action when user presses, holds down, or releases the right mouse button. To use on the Macintosh, you must set the emulateMultibuttonMouse property to true; then Control-click is interpreted as a right-click. To use this with Shockwave, you must disable the Shockwave context menu and pass right-clicks through to the player.
Mouse Within	Triggers an action when the pointer is inside a sprite.
Mouse Enter	Triggers an action when the pointer enters a sprite.
Mouse Leave	Triggers an action when the pointer leaves a sprite.
Keyboard Input	Lets the author specify a given key as a trigger.

You can add modifier keys to any trigger, so that a given trigger can launch two actions, depending on whether the modifier key is pressed. You can, for example, use Mouse Left and Mouse Left+Shift as separate triggers.

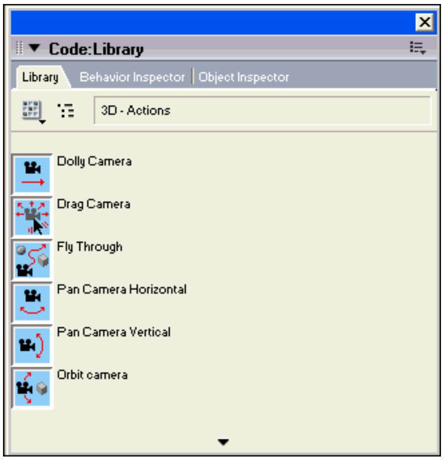
To view 3D action behaviors:

1 Click the Library Palette button on the Director toolbar.

2 Click the Library List button and select 3D.

3 Select Actions from the 3D submenu.

The action behaviors appear, as shown in the following figure.



Local actions

When you attach a local action to a sprite, that action responds only to a trigger that is attached to that same sprite. The following table describes the available local actions.

Name	Effect	Description
Create Box	Primitive	Adds a box to the 3D world each time the trigger action occurs. The author can set the dimensions and texture.
Create Particle System	Primitive	Creates a particle system whenever the trigger is activated. The user can set the number of particles; the life span of each particle; the starting and finishing color of particles; and the angle, speed, and distribution of particles on emission. The user can also set gravity and wind effects along any axis.
Create Sphere	Primitive	Adds a sphere to the 3D world each time the trigger action occurs. The author can set the diameter and texture.
Drag Camera	Camera	Provides full camera control, including panning (changing the direction in which the camera is pointing), dollying (moving), and rotating, through a single behavior. Use separate mouse triggers for panning, zooming, and rotating.
Drag Model	Model	Lets users move a model in any direction by dragging it with the mouse.
Drag Model to Rotate	Model	Lets you specify an axis or pair of axes around which you can rotate a model by dragging it with the mouse.
Fly Through	Camera	Simulates flying through the 3D world with a camera. Accepts separate triggers for forward and reverse travel and for stopping.
Click Model Go to Marker	Model	Moves the playhead to a marker in the Score when a model is clicked.
Orbit Camera	Camera	Circles the camera around a model.
Play Animation	Model	Plays a preexisting animation when the model is clicked. This behavior cannot be used with 3D text.

Public actions

As with local actions, you can add public actions to a movie by attaching them to any 3D sprite. Unlike local actions, public actions are triggered whether the trigger is attached to the same sprite as the action or to any other sprite. Public actions use the same triggers as local actions. The following table describes available public actions.

Name	Effect	Description
Dolly Camera	Camera	Dollies the camera into or out of the 3D scene by a specified amount each time the trigger action occurs. Dollying in and dollying out require separate triggers.
Generic Do	Custom	Lets you use the standard triggers to launch custom handlers or execute specific script methods. Requires knowledge of scripting in Director.

Name	Effect	Description
Pan Camera Horizontal	Camera	Pans along the horizontal axis by a specified number of degrees each time the trigger action occurs. Panning left and panning right require separate triggers.
Pan Camera Vertical	Camera	Pans along the vertical axis (up and down) by a specified number of degrees each time the trigger action occurs. Panning up and panning down require separate triggers.
Reset Camera	Camera	Resets the camera to its initial location and orientation when the trigger action occurs.
Rotate Camera	Camera	Rotates the camera around the *z*-axis by a specified number of degrees each time its trigger is activated. This makes the 3D scene appear to rotate and turn upside down.
Toggle Redraw	Drawing	Toggles the redraw mode on or off. Turning redraw off produces visible trails as a model moves through space. Turning redraw on causes the trails to disappear.

Independent actions

Independent actions don't require triggers. The following table describes the available independent actions.

Name	Effect	Description
Automatic Model Rotation	Motion	Automatically rotates a model around a given axis and continues rotating it while the movie plays. To rotate the model around multiple axes, attach multiple instances of the behavior to the sprite and select the desired axis for each one.
Level of Detail	Model	Enables the level of detail (LOD) modifier for the model. Dynamically lowers the number of polygons used to render the model as its distance from the camera increases. Reduces demands on the CPU.
Model Rollover Cursor	Model	Changes the mouse pointer to the pointer of your choice when the mouse rolls over the given model.
Show Axis	Debugging	Establishes red, green, and blue lines along the *x*-, *y*-, and *z*-axes, respectively, letting you see them in the 3D scene.
Subdivision Surfaces	Model	Enables the subdivision surfaces (SDS) modifier for the given model, which synthesizes additional detail to smooth out curves as the model's distance from the camera decreases.
Toon	Model	Enables the toon modifier, which renders the model in a cartoon style, with a reduced number of colors and distinct boundaries. The user can set the toon style precisely, selecting the number of colors, line color, brightness and darkness of highlights and shadows, and anti-aliasing.

Applying 3D behaviors

You apply 3D behaviors in the same way as standard behaviors in Director. You can attach as many behaviors to a sprite as needed, but each behavior that requires a trigger must have a unique trigger to activate it.

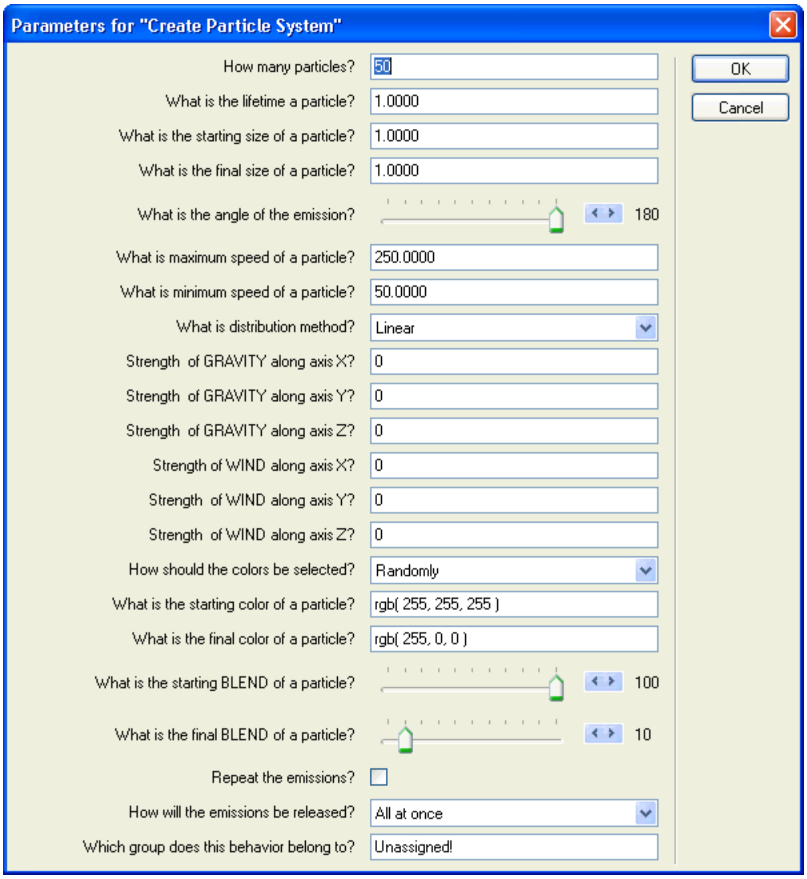
To apply a 3D behavior:

1 Open the Library palette.

2 Open the 3D library.

3 Attach an action behavior to the sprite either on the Stage or in the Score.

The Parameters dialog box appears. You can use this dialog box to control the behavior. Depending on the behavior you select, the dialog box might have many options or only a few.



Parameters for "Create Particle System"

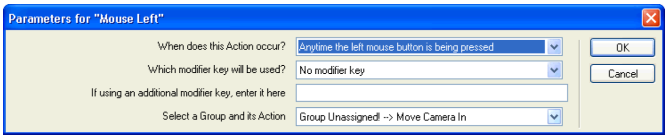
How many particles?	50
What is the lifetime a particle?	1.0000
What is the starting size of a particle?	1.0000
What is the final size of a particle?	1.0000
What is the angle of the emission?	180
What is maximum speed of a particle?	250.0000
What is minimum speed of a particle?	50.0000
What is distribution method?	Linear
Strength of GRAVITY along axis X?	0
Strength of GRAVITY along axis Y?	0
Strength of GRAVITY along axis Z?	0
Strength of WIND along axis X?	0
Strength of WIND along axis Y?	0
Strength of WIND along axis Z?	0
How should the colors be selected?	Randomly
What is the starting color of a particle?	rgb(255, 255, 255)
What is the final color of a particle?	rgb(255, 0, 0)
What is the starting BLEND of a particle?	100
What is the final BLEND of a particle?	10
Repeat the emissions?	☐
How will the emissions be released?	All at once
Which group does this behavior belong to?	Unassigned!

4 Specify options in the Parameters dialog box.

5 Click OK.

6 For local behaviors, attach a trigger behavior to the same sprite. For public behaviors, attach a trigger behavior to a specific sprite.

The Parameters dialog box appears. You can use this dialog box to control when the trigger should work; what modifier keys, if any, are associated with the trigger; and to which sprite group the trigger is assigned. For more information about groups, see "About behavior groups" on page 333.


Parameters for "Mouse Left"

When does this Action occur? Anytime the left mouse button is being pressed
Which modifier key will be used? No modifier key
If using an additional modifier key, enter it here
Select a Group and its Action Group Unassigned! --> Move Camera In

7 Specify options in the Parameters dialog box.

8 Click OK.

About behavior groups

The Parameters dialog boxes of the local and public action behaviors give you the option to assign the behavior to a group of behaviors. Groups let a single trigger initiate actions across multiple sprites. To establish a group, select a name for the group and enter that name in the Parameters dialog box of each behavior that you attach to the sprites in the group.

Note: A behavior group is not the same as a group node inside a 3D cast member.

Because the triggers are sent to the group name rather than to a specific sprite number, there are no reference changes to update when a sprite moves from one Score channel to another.

CHAPTER 16
Working with Models and Model Resources

This chapter covers the Lingo or JavaScript syntax methods and properties used to work with models and model resources, as well as lights and cameras to enhance Macromedia Director MX 2004 three-dimensional (3D) movies. For more information about the methods and properties listed here, see the Scripting Reference topics in the Director Help Panel, where you can find syntax, definitions, and examples. Because much of a model's behavior depends on modifiers (which are attached to the model), this chapter also discusses modifiers. A model's surface appearance, controlled by shaders and textures, is also discussed here.

Lights illuminate the 3D world and the models in it. Without lights, the world exists, and actions can take place, but users see nothing. You can add lights to your 3D world in your 3D modeling application and with the Property inspector or with Lingo or JavaScript syntax.

Cameras act as windows into a 3D world. Each camera that exists in a 3D cast member offers a different view into it, and each sprite that uses a 3D cast member uses one of these cameras. A camera's position can be moved with the Property inspector or the Macromedia Shockwave 3D window. You can also use the Director 3D behaviors or Lingo or JavaScript syntax to add camera and manipulate camera positions.

About models and model resources

Models are the objects you see in the 3D world. You can create models within Director. Spheres, boxes, planes, cylinders, and particle systems can be created either with Lingo or JavaScript syntax or with Director behaviors. These simple shapes are called *primitives*. They are the basic shapes from which more complicated models are built. (Particle systems are different from the other primitives: instead of being shapes, they create cascades of moving particles.) You can also create mesh primitives, which allow you to define any kind of complex shape you wish.

For the most part, however, you should create complex models outside of Director, using a 3D modeling application, and then imported into Director in the W3D file format.

Accessing properties and methods of a model or any other node type, such as a light or camera, requires that the node be on the Stage or explicitly loaded with the `preLoad()` method.

The following sections describe models, model resources, and primitives in more detail, along with the Lingo or JavaScript syntax used to work with them.

Model resources

Model resources are 3D geometries defined in 3D modeling software or created in Director MX 2004 by scripting in Lingo or JavaScript syntax. A model is an object that makes use of a model resource's geometry and occupies a particular position and orientation within the 3D world. Model resources are viewable only when in use by a model. Several models may use the same model resource.

Common model resource properties

The following properties are shared by all model resources:

Property Name	Access	Description	Range or Default
`name`	Get	Unique string naming model resource.	If imported, the name of the model. If created by scripting, the assigned name in the constructor method.

`type`	Get	Type of geometry.	`#plane`
			`#box`
			`#sphere`
			`#mesh`
			`#cylinder`
			`#particle`
			`#fromfile`

`bone.count`	Get	Total number of bones in bones hierarchy.	Nonnegative integer.
`modelResource.getBoneId("`*`name`*`")`	Get	Returns a unique ID for the bone named *`name`* in this model's bone hierarchy. Returns `FALSE (0)` if no bone by that name can be found.	None.

The `bone.count` and `getBoneID()` properties are only available for model resources that have a type of `#fromFile`. They are not available for primitives created with Lingo.

File-defined model resource properties

Model resources defined by a W3D file imported into Director or loaded via script have a `type` value of `#fromfile`. File-defined resources are automatically assigned level of detail (LOD) modifier settings that allow models using those geometries to adjust their level of detail as needed, depending on the model's distance from the camera. For more information, see "Level of detail (LOD) modifier properties" on page 366.

Primitives

Each type of primitive has its own set of methods and properties used to define its appearance.

Use the `newModelResource()` method to create the various primitives at runtime. Exceptions include mesh model resources, which require you to use the `newMesh()` method, and extruder resources, which require you to use the `extrude3D()` method of a text cast member.

Sphere properties

Spheres created at runtime aren't saved with the cast member's media when the Director movie is saved. Their `type` is `#sphere`. Their surface is generated by sweeping a two-dimensional semicircle arc in the *xy* plane from `startAngle` to `endAngle` in the *y*-axis. If `startAngle = 0.0` and `endAngle= 360.0`, a full sphere is generated. If `startAngle = 180.0` and `endAngle = 360.0`, a half sphere is generated. These properties can be modified or animated at runtime.

Property	Access	Description	Value Range
`radius`	Get and set	Radius of the sphere.	Positive floating-point value. The default is 25.0.
`resolution`	Get and set	Controls the number of polygons used in the creation of the sphere surface. The higher the value, the smoother the surface.	An integer value of 1 or greater. The default is 20.
`startAngle`	Get and set	Starting angle of the sweep.	Floating-point value of from 0.0 to 360.0. The default is 0.0.
`endAngle`	Get and set	Ending angle of the sweep.	Floating-point value of from 0.0 to 360.0. The default is 360.0.

Cylinder properties

Cylinders have a `type` property of `#cylinder`. Director generates a cylinder's surface by sweeping a 2D line around the *z*-axis in the *xy* plane from `startAngle` to `endAngle`. If `startAngle = 0.0` and `endAngle = 360.0`, a full cylinder is generated. If `startAngle = 180.0` and `endAngle = 360.0`, a half cylinder is generated. These properties can be modified or animated at runtime.

Property	Access	Description	Value Range
`topRadius`	Get and set	Radius of the top of the cylinder. Setting this value to 0 produces a cone.	Positive floating-point value. The default is 25.0.
`bottomRadius`	Get and set	Radius of the bottom of the cylinder.	Positive floating-point value. The default is 25.0.
`numSegments`	Get and set	Number of polygonal segments from bottom to top.	An integer value greater than 0.

Property	Access	Description	Value Range
`resolution`	Get and set	Number of polygonal segments around the circumference of the circle. Controls the smoothness of the cylinder's appearance.	An integer value greater than 1.
`height`	Get and set	Height of the cylinder along the *z*-axis.	Positive floating-point value. The default is 50.0.
`topCap`	Get and set	Value indicating whether the top of the cylinder is closed or open. `TRUE` = closed.	`TRUE` or `FALSE`. The default is `TRUE`.
`bottomCap`	Get and set	Value indicating whether the bottom of the cylinder is closed or open. `TRUE` = closed.	`TRUE` or `FALSE`. The default is `TRUE`.

Box properties

Boxes have a `type` property of `#box`. Box properties can be modified or animated at runtime.

Property	Access	Description	Value Range
`height`	Get and set	Height of the box, measured along the *y*-axis.	Positive floating-point value. The default is `50.0`.
`width`	Get and set	Width of the box, measured along the *x*-axis.	Positive floating-point value. The default is `50.0`.
`length`	Get and set	Length of the box, measured along the *z*-axis.	Positive floating-point value. The default is `50.0`.
`top`	Get and set	Value indicating whether the top of the box is closed or open. `TRUE (1)` = closed.	`TRUE (1)` or `FALSE (0)`. The default is `TRUE (1)`.
`bottom`	Get and set	Value indicating whether the bottom of the box is closed or open. `TRUE (1)` = closed.	`TRUE (1)` or `FALSE (0)`. The default is `TRUE (1)`.
`front`	Get and set	Value indicating whether the front of the box is closed or open. `TRUE (1)` = closed.	`TRUE (1)` or `FALSE (0)`. The default is `TRUE (1)`.
`back`	Get and set	Value indicating whether the back of the cylinder is closed or open. `TRUE (1)` = closed.	`TRUE (1)` or `FALSE (0)`. The default is `TRUE (1)`.
`left`	Get and set	Value indicating whether the left end of the box is closed or open. `TRUE (1)` = closed.	`TRUE (1)` or `FALSE (0)`. The default is `TRUE (1)`.
`right`	Get and set	Value indicating whether the right end of the box is closed or open. `TRUE (1)` = closed.	`TRUE (1)` or `FALSE (0)`. The default is `TRUE (1)`.

Property	Access	Description	Value Range
`lengthVertices`	Get and set	Number of vertices along the length of the box. Increasing the number of vertices improves lighting effects.	`2` or more. The default is `4`.
`width Vertices`	Get and set	Number of vertices along the width of the box. Increasing the number of vertices improves lighting effects.	`2` or more. The default is `4`.
`height Vertices`	Get and set	Number of vertices along the height of the box. Increasing the number of vertices improves lighting effects.	`2` or more. The default is `4`.

Plane properties

Planes are the default Director primitive. Planes, whose `type` property is `#plane`, are generated in the *xz* plane with script. Plane properties can be modified or animated at runtime.

Property	Access	Description	Value Range
`width`	Get and set	Width of the plane	Positive floating-point value. The default is 1.0.
`length`	Get and set	Length of the plane	Positive floating-point value. The default is 1.0.
`length Vertices`	Get and set	Number of vertices along the length of the plane	2 or more. The default is 2.
`width Vertices`	Get and set	Number of vertices along the width of the plane	2 or more. The default is 2.

Mesh generator properties

The mesh generator is the most complex model resource. It allows experienced 3D programmers to create complicated geometries at runtime.

The mesh generator primitive's `type` property is `#mesh` and is created by the member's `newMesh()` method. The parameters included with that method describe how large the mesh will be.

You can use the mesh deform modifier to manipulate vertex positions at runtime for `#mesh` or any other type of model resource. You can also use the `#mesh` primitive to change mesh properties directly, but this is usually not practical, because the mesh must be rebuilt mathematically after each modification.

Use these properties to work with mesh primitives:

Property	Access	Description	Value Range
`vertexList`	Get and set	Vector values for each vertex in the mesh. Several faces may share a single vertex.	Set the value to the number of vectors specified in your `newMesh` call.
`normalList`	Get and set	Vector values for each normal in the mesh. Several faces may share a single normal. A normalized vector is one in which all components are of unit length. You can use the `generateNormals()` method instead of specifying normals yourself. In that case, set 0 as the number of normals in your `newMesh()` call. The normals are calculated based on a clockwise vertex winding. That is to say, if you imagine the vertices being wound down a spindle, they would be wound from left to right, in a clockwise manner.	No default. Instead, set the value to the number of vectors specified in your `newMesh` call.

`texture Coordinate List`
 	
`texcoordlist`
	Get and set	A list of sublists identifying locations in an image used for texture-mapping a triangle. Each sublist contains two values between 0.0 and 1.0 that define a location and can be arbitrarily scaled to any texture size.	No default. Instead, set the value to the number of two-element sublists specified in your `newMesh` call.

`colorList`	Get and set	List identifying every color in the mesh. Any color can be shared by several faces. Alternatively, specify texture coordinates for the mesh faces and apply a shader to models using this model resource.	No default. Instead, set the value to the number of colors specified in your `newMesh` call.
`face.count`	Get	Number of triangles in the mesh.	The number of faces specified in your `newMesh` call.
`face[`*`index`*`].vertices`	Get and set	List indicating which vertices to use for faces at designated index points.	Set the value to a list of three integers specifying the indexes of the vertices in the `vertexList` that define this face.

Property	Access	Description	Value Range
`face[`*`index`*`].normals`	Get and set	List indicating which normals to use for faces at designated index points.	Set the value to a list of three integers specifying the indexes of the normals in the `normalList` that each point of the triangle should use. Don't set a value if you aren't defining your own normals.

`face[`*`index`*`].textureCoordinates`
 `face[`*`index`*`].texcoords`
	Get and set	List indicating which texture coordinates to use for faces at designated index points.	Set the value to a list of three integers specifying the indexes of the texture coordinates in the `textureCoordinates` List that each point of the triangle should use.
			 Don't set a value if you aren't defining your own texture coordinates.

`face[`*`index`*`].colors`	Get and set	List indicating which colors to use for faces at designated index points.	Set the value to a list of three integers specifying the indexes of the colors in the colorList that each point of the triangle should use. Don't set a value if you aren't defining your own colors.
`face[`*`index`*`].shader`	Get and set	Shader used for rendering the face.	Shader defined for use with this face.

Mesh generator methods

Use these methods to work with mesh primitives:

Method	Description	Returns
`build()`	Builds the mesh according to the current property values. (The mesh construction properties specified in the previous table have no effect until `build()` is called.) Generates a script error if any properties specify an invalid list.	Nothing
`generateNormals (`*`style`*`)`	Generates a new normal for every vertex in every triangle. The *`style`* parameter can be `#flat`, so that each triangle is clearly delineated, or `#smooth`. The method assumes that all triangles were specified in a clockwise order.	Nothing

Particle system properties

Particle systems are unique among model resources in that they include animation by default. Particle systems, whose `type` is `#particle`, can have an almost infinite variety of appearances, simulating fire, smoke, running water, and other streaming or bursting effects.

Use these properties to work with particle systems:

Property	Access	Description	Value Range
`lifetime`	Get and set	Lifetime of all particles emitted, in milliseconds.	Positive integer. The default is `10.000` ms.
`colorRange. end`	Get and set	Color value of a particle at the end of its life.	Any color value. The default is `color(255, 255, 255)`.
`colorRange. start`	Get and set	Color value of a particle at the start of its life.	Any color value. The default is `color(255, 255, 255)`.
`tweenMode`	Get and set	The variation of a particle's color throughout its life. The change can be based on either velocity or age.	`#velocity`: Alter particle color between `colorRange. start` and `colorRange. end` based on velocity. `#age`: Alter particle color between `colorRange.start` and `colorRange.end` based on the particle's lifetime.
`sizeRange. start`	Get and set	The size of a particle at the start of its life.	Positive integer. The default is `1`.

Property	Access	Description	Value Range
`sizeRange.end`	Get and set	Size of a particle at the end of its life. The size is linearly interpolated between *`startSize`* and *`endSize`*.	Positive integer. The default is `1`.
`blendRange.start`	Get and set	Opacity of a particle at the start of its life.	Any value between `0.0` and `100.0`.
`blendRange.end`	Get and set	Opacity of a particle at the end of its life.	Any value between `0.0` and `100.0`.
`texture`	Get and set	Texture to use when drawing each particle. The default is `void`.	Texture object.
`emitter.numParticles`	Get and set	Number of particles in a burst or stream.	Positive integer. The default is `1000`.
`emitter.mode`	Get and set	Mode in which particles are emitted.	`#burst`: All particles emitted at once. `#stream`: X particles emitted per frame with X equalling `emitter.numParticles/(`*`lifetime`*`*millisecondsPerFrame)`. Note: `millisecondsPerFrame` is the time elapsed between rendered frames.
`emitter.loop`	Get and set	`TRUE (1)` or `FALSE (0)` value indicating whether particles die `(FALSE)` or are recycled (`TRUE`).	`0` or `1`.
`emitter.direction`	Get and set	Vector of original emission. At 1,0,0, the default, particles are emitted randomly over a sphere.	Any vector.
`emitter.region`	Get and set	Point, line, or region from which particles are emitted.	Possible values: single vector for point source two vectors for line segment four vectors for quadrilateral
`emitter.distribution`	Get and set	Half the angle over which particles are distributed, measured from the top of the screen.	`0` to `180`.

Property	Access	Description	Value Range
`emitter.path`	Get and set	Vector positions that define the path the particles follow.	Vector list.
`emitter.pathStrength`	Get and set	Degree to which particles remain on a path.	Percentage between `0.0` and `100.0`.
`emitter.minSpeed`	Get and set	Minimum emission speed. (Particles are emitted at random speeds between a minimum and a maximum.)	Settable value. The default is `1.0`.
`emitter.maxSpeed`	Get and set	Maximum emission speed. (Particles are emitted at random speeds between a minimum and a maximum.)	Settable value. The default is `1.0`.
`emitter.drag`	Get and set	A drag value affecting simulation at each animation step.	Percentage between `0.0` and `100.0`.
`emitter.gravity`	Get and set	Vector representing simulated gravity. The vector's length indicates its strength.	Any vector.
`emitter.wind`	Get and set	A vector representing simulated wind pushing particles in a given direction. The vector's length indicates its strength.	Any vector.

The extruder model resource

You can create extruder model resources only by using an existing text cast member. In many cases, you may choose to use the 3D text capabilities of the Property inspector instead.

Creating an extruder model resource is simple. If member 1 is a text cast member and member 2 is a 3D cast member, use the following script:

```
member(1).extrude3d(member(2))
```

This generates a model resource in member 2 that is an extrusion of the 2D text in member 1.

Extruder model resources inherit all the 3D-related properties of the source text cast member.

For example:

```
mr = member("text").extrude3d(member("3D"))
put mr.smoothness
-- 5
put mr.tunnelDepth
-- 50
```

Cast member methods

If the models and model resources you need aren't contained in a particular cast member, the following methods let you create models and model resources using other 3D cast members at runtime.

Method	Description	Returns
`loadFile` (*`fileName, Overwrite, GenerateUniqueNames`*)	This method loads a W3D format file from *`fileName,`* adds all models as children of the world, and updates all palettes. You can call this method only if the cast member's state property is either `-1`, meaning that an error occurred during a previous attempt to load the file, or `4`, meaning that media loading is complete. If an attempt is made to call `loadFile` while the cast member is streaming media in, a script error is generated. *`Overwrite`* is an optional variable that can be `TRUE (1)` or `FALSE (0)`: `TRUE (1)` means the old world is replaced by the contents of the file. `FALSE (0)` means the new file is merged into the existing world. *`GenerateUniqueNames`* is a variable that has no meaning unless *`Overwrite`* is `FALSE (0)`. If *`Overwrite`* is `FALSE (0)`, then if *`GenerateUniqueNames`* is `TRUE (1)`, all new elements sharing the same name as existing elements are assigned a new, algorithmically determined unique name. If *`GenerateUniqueNames`* is `FALSE (0)`, all existing elements sharing the same name as new elements being read into the file are replaced by the new elements.	Nothing if the operation is successful, or a script error if the operation fails
`cloneModelFromCastmember` (*`name, model, castmember`*)	Performs a deep clone of a model from one cast member and puts it into another cast member. The model, its resources, its children, and its children's resources all are put into the new cast member.	A model object
`cloneMotionFromCastMember` (*`name`*, *`motion`*, *`castmember`*)	Performs a deep clone of a motion from one cast member and puts it into another cast member.	A motion object

Method	Description	Returns
`newModelResource(`*`name, type`*`)`	Creates a new model resource and adds it to the model resource palette. The *`type`* can be `#plane`, `#box`, `#sphere`, `#cylinder`, or `#particle`. The *`type`* cannot be `#mesh`. To create a new mesh model resource, use the `newMesh` method detailed below.	New model resource object
`newMesh(`*`name, numFaces, numVertices, numNormals numColors, numTextureCoordinates`*`)`	Creates a new mesh model resource. *`numFaces`* is the user-specified number of triangles. *`numVertices`* is the user-specified number of vertices. A vertex can be used by more than one face. *`numNormals`* is the user-specified number of normals. Enter `0` or omit this step to use the `generateNormals()` method. *`numColors`* is the user-specified number of colors. You can specify a color for each point of a triangle. *`numTextureCoordinates`* is the number of user-specified texture coordinates. Enter `0` or omit this step to get the default coordinates.	New mesh model resource

Models

Models can be referred to by name or number. Models can be added to or removed from the world at any time.

In the member's parent-child hierarchy, each model can have a maximum of one parent, but it can have an unlimited number of children. Models with no parent can exist in the 3D world but are not rendered. A child's position and orientation depend on its parent's position and orientation, and it changes when the position and orientation of the parent changes. Models can have any other group, light, camera or model as their parent, or they can have no parent specified. In this case, their `transform` property describes their position and rotation in the 3D world, and is identical to their `getWorldTransform()` property. All models that have parents have a relationship both to their immediate parent and to the world parent. You can add or remove models from the 3D world at any time by using the `addToWorld()` or `removeFromWorld()` methods.

For example, if the first child of the model named `car1` is a wheel model, the following `transform` script would refer to the position of the wheel relative to the model named `car1`:

```
car1.child[1].transform.position
```

To refer to the position of the wheel model relative to the world itself, use `getWorldTransform()`:

```
car1.child[1].getWorldTransform().position
```

Node types

A model is one of four types of objects that share the same transform, parent, and child properties. The others are cameras, lights, and groups. Models, cameras, lights, and groups are generically referred to as *node types* or *nodes.* Nodes can be each other's parents or children, so long as any one node has exactly one parent. A node can have any number of children. A model, for example, can be the child of a light and the parent of a group.

Model properties

The properties of a model determine its particular appearance and relationship to the rest of the 3D world.

Property	**Access**	**Description**	**Value**
`name`	Get	Unique string name.	Any string.
`parent`	Get and set	This model's parent; either another object or the 3D cast member itself.	An object or cast member.
`child.count`	Get	Number of children (but not grandchildren) of a given model.	An integer.
`transform`	Get and set	Transform object representing this model's position and orientation relative to its parent's position and orientation: `transform.position` gives the relative position. `transform.rotation` gives the relative rotation.	Set: a transform object. Get: reference to a transform object.
`userData`	Get and set	A property list containing all properties assigned to the model. Users can add, remove, get, and set properties on this list.	The default list includes the properties assigned in the 3D modeling tool. Additional properties may also be added.
`resource`	Get and set	Model resource object defining model's geometry.	Model resource object.
`shaderList`	Get and set	List of all shaders used by the model. Setting this property to a single shader sets every element of the `shaderList` to that shader.	List.
`shaderList.count`	Get	Number of shaders the model uses.	Positive integer.
`shaderList.[`*`index`*`]`	Get and set	Provides access to a particular shader used in a specific region of the model.	List.
`shader`	Get and set	Provides access to the first shader in the shader list.	Shader object.

Property	Access	Description	Value
bounding Sphere	Get	A list containing a vector and a floating-point value. The vector represents the world position and the value represents the radius of a bounding sphere surrounding the model and all its children.	[vector (0,0,0), 0.0]
world Position	Get and set	Position of the model in world coordinates. Shortcut for the method node.getWorldTransform ().position.	Vector object.
visibility	Get and set	The way in which the sides of the model's resource are drawn. The choices are as follows: #none, in which no polygons are drawn and the model is invisible. #front, in which only polygons on the outer surface of the model are drawn, so that, if the camera were inside the model, the model wouldn't be seen. Also known as "back face culling," this option optimizes performance. #back, in which only polygons on the inside of the object are drawn, so that if the camera were outside the model, the model wouldn't be seen. #both, in which all polygons are drawn and the model is visible regardless of orientation. This may solve drawing problems, but it can also affect performance because twice as many polygons must be drawn. The default is #front.	#none: #front #back #both
debug	Get and set	Value indicating whether debug information is drawn for this model. If the value is TRUE (1), lines from the x, y, and z axes are drawn sprouting up from the model to indicate its orientation, and a bounding sphere is drawn around the model.	TRUE (1) or FALSE (0). The default is FALSE (0).
bounding Sphere	Get	A list containing a vector and a floating-point value. The vector represents the position of the model in world space, and the floating-point value represents the radius of the bounding sphere that contains the model and its children.	bounding Sphere
pointAt Orientation	Get and set	A list of two orthogonal vectors, [objectRelativeDirecton, objectRelativeUp], that control how the model's pointAt() method works.	pointAt Orientation

Model methods

Use these methods to work with models:

Method	Description	Returns
`addChild(`*`aNode, preserveWorld`*`)`	Adds *`aNode`* to this model's list of children. An equivalent operation is to set *`aNode.parent`* to equal *`this model`*. The *`preserveWorld`* argument is optional. It can have two values: `#preserveWorld` or `#preserveParent`. If the value is `#preserveWorld`, the default, the world transform of the child being added remains intact. If the value is `#preserveParent`, the child's existing transform is interpreted as parent-relative.	Nothing
`child[`*`index`*`]`	Returns the child at the specified position in the index.	Node object
`child(`*`name`*`)`	Returns the child named *`name`*.	Node object
`clone(`*`name)`*	Clones a model named *`name`*, adds it to the child list of the model's parent, and adds it to the world. The clone shares the same model resource, shader list, and parent as the original model, but it has unique copies of the model's transform and modifier properties. All children of the model are automatically cloned. This can be avoided by removing the children, performing the cloning operation, and then adding the children back. If the name is omitted or is `""`, the clone isn't added to the model palette, has no parent, and has no children. This option lets you quickly create temporary model instances.	Model object
`cloneDeep(`*`name)`*	Clones both the model and the model resource used by the model's children. Modifications to the clones' resource don't affect the source model's resource. This is a more memory-intensive operation than `clone(`*`name`*`)`.	Model object
`addtoWorld()`	Adds the model to the currently active 3D world, setting its parent as "world." Equivalent to `model.parent=member("scene").group ("world")` All newly created models are added to the world by default, without it being necessary to use this method.	Nothing
`remove FromWorld()`	For models whose parent hierarchy terminates in the world, this sets their parent to `void` and removes them from the world. Otherwise it does nothing.	Nothing

Method	Description	Returns
`isInWorld()`	Returns a Boolean value indicating if the model is currently in the world (TRUE) or not (FALSE). This is useful for detecting if a given node's parent-heirarchy tree terminates with the world group object or not.	`TRUE (1)` or `FALSE (0)`
`registerScript (`*`eventName, handlerName, scriptInstance`*`)`	Registers a handler named *`handlerName`* that is called in the *`scriptInstance`* when the member method *`sendEvent()`* is called with *`eventName`* as an argument. If *`scriptInstance`* is 0, a movie script handler is called. The user defines what *`eventName`* is. The *eventName* specified can be one of a default set of events or a user defined custom event. The default events are #collideAny, #collideWith, #animationStarted, #animationEnded, #timeMS.	Nothing
`addModifier (`*`symbol`*`)`	Adds the modifier *`symbol`*.	`TRUE (1)` if *`symbol`* is a valid modifier `FALSE (0)` if *`symbol`* is not a valid modifier
`removeModifier (`*`symbol`*`)`	Removes the first modifier identified by *`symbol`*.	`TRUE (1)` if *`symbol`* is a valid modifier and attached to the model `FALSE (0)` if *`symbol`* is not a valid modifier or is not attached to the model
`update()`	Updates animation timing without rerendering. Used to force update of bone positions in an animation while inside a script call.	`TRUE (1)` or `FALSE (0)`
`translate (`*`directionVector, relativeTo`*`)`	Moves the model directionVector.length() in the direction of the vector *`directionVector`*. The *`relativeTo`* argument is optional and defaults to `#self`.	Nothing
`translate (`*`x,y,z, relativeTo`*`)`	Moves the model distance *x* along the *x*-axis, distance *y* along the *y*-axis, and distance *z* along the *z*-axis. The *`relativeTo`* argument is optional and defaults to `#self`. This method can also be written as `translate(vector(x,y,z)` *`relativeTo`*`)`.	Nothing
`rotate(`*`x,y,z, relativeTo`*`)`	Rotates the model by *x*° around the *x*-axis, *y*° around the *y*-axis, and *z*° around the *z*-axis. The *`relativeTo`* argument is optional and defaults to `#self`. If included, it defines the coordinate space of the axes. This method can also be written as `rotate(vector(`*`x,y,z`*`)` *`relativeTo`*`)`.	Nothing

Method	Description	Returns
`rotate (`*`position, axis, angle, relativeTo`*`)`	Rotates the model around the axis vector in the specified position the specified number of degrees. The *`relativeTo`* argument is optional and defaults to `#self`.	Nothing
`scale(`*`uniform Scale`*`)`	Scales the model the same amount in all directions.	Nothing
`scale(`*`x, y, z`*`)`	Scales the model by a factor of *x* in the *x* dimension, *y* in the *y* dimension, and *z* in the *z* dimension. Scaling is applied in object-relative space.	Nothing
`pointAt(`*`world Position, worldUp`*`)`	Points the node's "front" at the world position and then tries to align the node's "up" with the worldUp specified, and that the node's "front" and "up" are determined by the node's *`pointAtOrientation`* property. Both the object-relative axes are defined by the *`pointAtOrientation`* property. Default values are an object-relative forward direction of vector (`0`, `0`, `-1`) and an object-relative up direction of vector (`0`, `1`, `0`).	Nothing
`getWorld Transform()`	Calculates and returns a transform that converts object-relative positions for this model into world-relative positions.	A transform object

Moving models

Because the 3D world has no absolute frame of reference, moving and rotating is much more complex than in 2D, where all movement is in relation to screen position.

In 3D, everything is drawn relative to the camera's frame of reference. If the camera is behind an object, when the object moves to the left relative to the center of the world, or *world origin*, it appears to move toward the right of the screen.

Each piece of position and orientation information can be expressed relative to one or more frames of reference. A model's transform property, for instance, expresses its position and rotation relative to the model's parent. In general, there are four frames of reference to consider: relative to the object (model, light, camera) itself, relative to the object's parent, relative to the world, and relative to some other object.

- Object-relative: When you create a model in a 3D modeling program, you build it relative to its own frame of reference. For instance, when you create a model of a car, the front of the car may be pointed along its *z*-axis and the antenna may be pointed along its *y*-axis. To move such a car forward (along its *z*-axis) regardless of which direction it is pointing relative to the camera or the world, use `car.translate(0,0,10)`. To turn the car left, use `car.rotate(0,45,0)`.

  The car model might have wheel models as children. To rotate the wheel of a car relative to itself, rather than relative to its parent (the car), use the following script:

  ```
  wheel.rotate(0,10,0)
  ```

or

`car.child[1].rotate(0,10,0, #self)`

where the fourth parameter of the `rotate` method is the object the rotation should be relative to.

- Parent-relative: A model's `transform` property expresses its position and rotation relative to the model's parent. If you want the wheels of the car to move outward regardless of how the wheels are turned, use `car.child[1].translate(10,0,0,#parent)` or `car.child[1].transform.translate(10,0,0)`. If you want a planet model that is a child of the sun to orbit around the sun, use `planet.rotate(0,5,0, #parent)`.
- World-relative: If you want the car to move along the world's *x*-axis regardless of which way it is facing, use `model.translate(10,0,0,#world)`. If you want to rotate the car 20° around the world *y*-axis, with the rotation taking place at the world location vector (10, 10, 10), use `model.rotate(vector(10,10,10), vector(0,1,0), 20, #world)`.
- Relative to another object: If you want to move an object so that it goes toward the right edge of the screen, use `model.translate (vector(10,0,0), sprite(1).camera)`. If you want to rotate the object parallel to the camera and around the center of the screen, use `model.rotate(vector(0,0,0), vector(0,0,1), 20, sprite(1).camera)`.

Shaders

A model resource defines a model's shape, and shaders define the model's surface colors and reflectivity. You can use just one shader or more than one. Each mesh in a model resource can have its own shader. For example, a box might have six different shaders, one for each mesh (a box is actually composed of 6 plane meshes carefully arranged). If you do not specify a shader, the default `#standard` shader is used. If the shader's properties are modified, the change affects all models that use that shader.

Models that are created with script are assigned the standard shader. You can replace the default shader of a model with any of the other types of shaders.

Properties of the standard shader

The standard shader makes the surface of a model appear in a photorealistic style. Use these properties to work with the standard shader:

Property Name	Access	Description	Default
`name`	Get	The string name of this shader.	None
`ambient`	Get and set	A color object describing the surface's reaction to ambient light.	`color(63,63,63)`
`diffuse`	Get and set	A color object describing the surface's reaction to diffuse light. Ambient and diffuse color objects together describe a model resource's base color.	`color(255,255,255)`
`specular`	Get and set	A color object describing the surface's specular highlight color. This setting has an effect only if there are lights in the scene whose specular property is `TRUE (1)`.	`color(255,255,255)`

Property Name	Access	Description	Default
`shininess`	Get and set	An integer between `0` and `100` indicating how shiny a surface is.	`30.0`
`emissive`	Get and set	A color object describing the color of light this object seems to give off. This does not turn the surface using this shader into a light source; it just gives it the appearance of being one.	`color(0,0,0)`
`blend`	Get and set	An integer between `0` and `100` indicating how transparent `(0)` or opaque `(100)` this surfaces is. Unlike with a texture that includes alpha information, this setting affects the entire surface uniformly.	`100`
`transparent`	Get and set	This property controls whether or not the model is blended using alpha values or rendered as opaque. The default is `TRUE (1)` (alpha blended). The functionality of `shader.blend` is dependent on `shader.transparent`.	`TRUE (1)`

`renderStyle`	Get and set	This property can take the following values:
		`#fill`
		`#wire`
		`#point`
		When `shader.renderStyle = #fill`, the polygons of the mesh are filled.
		When `shader.renderStyle = #wire`, the polygon edges of the mesh are rendered.
		When `shader.renderStyle = #point`, the vertices of the mesh are rendered, provided that `#Fill` is supported by the `#software` renderer. The `#point` and `#wire` values do not work with the software renederer.
			`#fill`

`flat`	Get and set	When `shader.flat = TRUE (1)`, the mesh should be rendered with flat shading instead of Gouraud shading, which shades each polygon separately. Flat shading shades the mesh as a whole.	`FALSE (0)`
`textureList`	Get and set	A shader can use up to eight layers of textures. This eight-element list defines which texture is used for which layer. Get: Returns a list of texture objects, one per layer. Set: Specifies a texture object to be applied to all layers. An argument of `void` disables texturing for all layers.	`void`

Property Name	Access	Description	Default
`textureList [index]`	Get and set	A shader can use up to eight layers of textures. This property gives access to the texture at the indicated index position.	`void`
`texture`	Get and set	This property allows access to the texture for the first layer. It is equivalent to `textureList[1]`. An argument of `void` can be used to disable texturing for the first layer.	`void`

`reflectionMap`	Get and set	Get: Returns the texture associated with the third layer.
		Set: Specifies a texture to be used in the third layer and applies the following values:
		`textureModeList[3] = #reflection`
		`blendFunctionList[3] = #blend`
		`blendSourceList[3] = #constant`
		`blendConstantList[3] = 50.0`
			`void`

diffuseLightMap	Get and set	Get: Returns the texture associated with the second layer.
		Set: Specifies a texture to be used in the second layer and applies the following values:
		`textureModeList[2] = #diffuse`
		`blendFunctionList[2] = #multiply`
		`blendFunctionList[1] = #replace`
			`void`

`specularLight Map`	Get and set	Get: Returns the texture associated with the fifth layer.
		Set: Specifies a texture to be used in the fifth layer and applies the following values:
		`textureModeList[5] = #specular`
		`blendFunctionList[5] = #add`
		`blendFunctionList[1] = #replace`
			`void`

`glossMap`	Get and set	Get: Returns the texture associated with the fourth layer.
		Set: Specifies a texture to be used in the fourth layer and applies the following values:
		`textureModeList[4] = #none`
		`blendFunctionList[4] = #multiply`
			`void`

Property Name	Access	Description	Default
textureMode List[*index*]	Get and set	This property allows access to the texture coordinate generation method used for a texture at the texture level and then to allows you to change how textures are applied to a model's surface. The property can take the following values: #none #wrapPlanar #wrapCylindrical #wrapSpherical #reflection #diffuseLight #specularLight	#none
textureMode List	Get and set	Get: Returns a list of texture coordinate generation methods, one per layer. Set: Specifies texture coordinate generation modes to be applied to all layers. Possible values are as follows: #none #wrapPlanar #wrapCylindrical #wrapSpherical #reflection #diffuseLight #specularLight	#none
textureMode	Get and set	Access to the texture coordinate generation method for the first layer. Possible values are as follows: #none #wrapPlanar #wrapCylindrical #wrapSpherical #reflection #diffuseLight #specularLight	#none
wrapTransform List[*index*]	Get and set	Access to the texture coordinate generation transform associated with a specified layer. This transformation has effect only if the textureModeList[*index*] is #wrapPlanar, #wrapSpherical, or #wrapCylindrical. Controls the orientation of texture coordinate generation in model-relative space. Use this property to change the orientation, offset, and scale of how the wrapTransformList[*index*] is applied on the model.	transform(50.0000,0.0000,0.0000,0.0000, 0.0000,50.0000,0.0000,0.0000, 0.0000,0.0000, 50.0000,0.0000, 0.0000,0.0000, 0.0000,1.0000)

Property Name	Access	Description	Default
`wrapTransform List`	Get and set	Controls the orientation of UV generation in model space. Get: Returns a list of texture coordinate generation transforms, one per layer. Set: Specifies a texture coordinate generation transform to be applied to all layers.	`transform(50.0000,0.0000,0.0000,0.0000, 0.0000,50.0000,0.0000,0.0000, 0.0000,0.0000, 50.0000,0.0000, 0.0000,0.0000, 0.0000,1.0000)`
`wrapTransform`	Get and set	Access to the texture coordinate generation transform for the first layer. Controls the orientation of the UV generation in model space.	`transform(50.0000,0.0000,0.0000,0.0000, 0.0000,50.0000,0.0000,0.0000, 0.0000,0.0000, 50.0000,0.0000, 0.0000,0.0000, 0.0000,1.0000)`
`texture TransformList`	Get and set	Access to the list of texture coordinate modifier transforms, one per texturing layer. The `textureTransform` is applied to all texture coordinates regardless of the `textureMode` property setting. This is the last modification of the texture coordinates before they are sent to the renderer. Allows you to manipulate the scale, orientation, and positional offsets of the source image before it's wrapped. `WrapTransformList` changes the projection of the transformed texture. The `textureTransform` matrix operates on the texture in `textureImage` space, which is defined to exist only on the x,y plane. Rotations about the *z*-axis are rotated around the `(0,0)` point, which maps to the upper left corner of the texture. Translating by integers when `textureRepeat` is `TRUE (1)` has no effect, because the width and height of the textures are defined to be `1.0` in `textureImage` space. Care must be taken not to scale any dimension (even *z*) by `0`.	Identity transform
`textureTransform List[`*`index`*`]`	Get and set	Access to the texture coordinate modifier transform associated with a specified layer.	Identity transform
`texture Transform`	Get and set	Access to the texture coordinate modifier transform for the first layer.	Identity transform

Property Name	Access	Description	Default
blendFunction List[*index*]	Get and set	Access to the blending method associated with a texture layer at the position indicated by *index*, which must be a positive integer smaller than or equal to 8. Possible values are as follows: #replace #multiply #add #blend #alpha #constant For detailed information about all of these options, see the Scripting Reference topics in the Director Help Panel.	#multiply
blendFunction	Get and set	Access to the list of blending methods, #multiply, #replace, #blend, and #add, for the first layer.	#multiply
blendFunction List	Get and set	Access to the list of blending methods, #multiply, #replace, #blend, and #add, for all layers.	#multiply
blendSource List[*index*]	Get and set	Access to the blending source associated with a specified layer. When the blendFunction property is set to #blend for the ‹index›th layer, this results in the ‹index›th texture being combined with the result of the previous layers for the entire texture using a single blending ratio. The blending ratio, in this case, is the value of blendConstant for layer ‹*index*›. For example, if the layer at that index position's blendConstant value is 0.9, the resultant texture will be 90% of the texture at that index position and 10% of the result of the previous texture layers Possible values are #constant and #alpha.	#constant
blendSource List	Get and set	Access to the blending sources for each layer, providing that the blend method is set to #blend. Possible values are #constant and #alpha.	#constant
blendSource	Get and set	Access to the blending sources for the first layer, providing that the blend method is set to #blend. Possible values are #constant and #alpha.	#constant

Property Name	Access	Description	Default
`blendConstantList[`*`index`*`]`	Get and set	The blending ratio used for a specific layer when the blend method is set to `#blend` and `blendSourceList[`*`index`*`]` is set to `#constant`. Returns a floating-point value from `0.0` to `100.0`.	`50.0`
`blendConstantList`	Get and set	The blending ratio used for any layer when the blend method is set to `#blend` and `blendSourceList[`*`index`*`]` is set to `#constant`. Returns a floating-point value from `0.0` to `100.0`.	`50.0`
`blendConstant`	Get and set	The blending ratio used for the first layer when the blend method is set to `#blend` and `blendSourceList[`*`index`*`]` is set to `#constant`. Returns a floating-point value from `0.0` to `100.0`.	`50.0`
`textureRepeatList[`*`index`*`]`	Get and set	Allows you to get or set the texture clamping behavior associated with a specified layer. Texture clamping refers to how a texture "clamps" to its shader. If the ratio of the texture to the shader is less than 1 to 1 and `textureRepeatList` is set to `TRUE (1)`, the texture tiles over the shader. If `textureRepeatList` is set to `FALSE (0)`, the texture isn't repeated but appears only once in one part of the shader. If the ratio of the texture to the shader is greater than 1 to 1 and `textureRepeatList` is set to `FALSE (0)`, the border of the texture is extended past the unit UV coordinate range.	`TRUE (1)`
`textureRepeatList`	Get and set	Access to the list of texture clamping behaviors, one per layer. When set to `FALSE (0)`, the border of the texture is extended past the unit UV coordinate range. Get: Returns a list of texture clamping behaviors, one per layer. Set: Specifies a texture clamping behavior to be applied to all layers.	`TRUE (1)`
`textureRepeat`	Get and set	Access to the texture clamping behavior for the first layer. When set to `FALSE (0)`, the border of the texture is extended past the unit UV coordinate range.	`TRUE (1)`

Properties of the painter shader

The #painter shader gives the model a painted effect. Use these properties to work with the painter shader:

Property	Access	Description	Default
name	Get and set	Name of shader	None
style	Get and set	Possible values: #toon: sharp transitions between available colors #gradient: smooth transitions between available colors #blackAndWhite: sharp transitions between black and white	#gradient
colorSteps	Get and set	Number of color steps used for lighting calculations	2
shadow Percentage	Get and set	Percentage of lighting intensity that is the threshold between highlight and shadow	50
highlight Percentage	Get and set	Percentage of lighting steps to be treated as highlight	50
shadowStrength	Get and set	Factor controlling darkness of shadowed areas	1.0
highlight Strength	Get and set	Factor controlling brightness of highlighted areas	1.0

Properties of the newsprint shader

The #newsprint shader creates a dithering effect similar to a newspaper photograph. Use these properties to work with the newsprint shader:

Property	Access	Description	Default
name	Get and set	Name of shader	None
brightness	Get and set	Value controlling amount of white in shader	0.0
density	Get and set	Value controlling density of dots in newsprint image	45.0

Properties of the engraver shader

The `#engraver` shader gives the effect of an engraved metal surface. You can control the size and number of etched lines by adjusting the `brightness` and `density` properties, respectively. Use these properties to work with the engraver shader:

Property	Access	Description	Default
`name`	Get and set	Name of shader	None
`brightness`	Get and set	Value controlling amount of white in shader	`0.0`
`density`	Get and set	Value controlling number of lines used to shade an area	`40.0`
`rotation`	Get and set	Angle describing 2D rotational offset for engraving lines	`0.0`

Textures

Each shader can have textures applied to it. Textures are 2D images drawn on the surface of a model. The appearance of the model's surface is the combined effect of the shader and textures applied to it. If you do not specify a texture, a default red-and-white bitmap is used.

The pixel height and width of the 2D images you use as textures should be powers of 2 (that is, 2, 4, 8, 16, 32, and so on). This is because most video cards scale images to powers of 2. If the image used does not have pixel dimensions that are a power of 2 (values including 2, 4, 8, 16, etc.), there will be a decrease in both rendering performance and visual quality. In addition, all the textures used in a 3D scene must be able to fit in the computer's video RAM at the same time. If not, Director switches to software rendering, which slows performance.

Be aware of the limitations of your video RAM and that of your intended audience. Some video cards have as little as 4 megabytes of video RAM. Carefully budget your total texture size when designing your 3D world.

Texture properties

Use these properties to work with textures:

Property	Access	Description	Default
`name`	Get and set	Name of texture.	None
`type`	Get	Possible values: `#fromfile`: bitmap defined as part of 3D import `#fromCastmember`: bitmap derived from Director cast member `#fromImageObject`: the texture was created from a Director image object	None
`member`	Get and set	If the type is `#castmember`, this property identifies the source of the bitmap. If the type if `#fromfile`, this property is `void`.	None

Property	Access	Description	Default
`width`	Get	Width, in pixels.	None
`height`	Get	Height, in pixels.	None
`quality`	Get and set	Property with the following possible values: `#low`: texture is not mipmapped `#medium`: mipmapping is at a low bilinear setting (default) `#high`: the mipmapping is at a high trilinear setting For more information, see the Scripting Reference topics in the Director Help Panel.	`#medium`
`nearFiltering`	Get and set	Determines whether bilinear filtering is used when rendering a projected texture map that covers more screen space than the original. For more information, see the Scripting Reference topics in the Director Help Panel.	`TRUE (1)`

Property	Access	Description	Default
`compressed`	Get and set	The property can be `TRUE (1)` or `FALSE (0)`: `TRUE (1)`: the texture is compressed `FALSE (0)`: the texture is not compressed The value changes automatically from `TRUE (1)` to `FALSE (0)` when the texture is to be rendered. The value can be set to `FALSE (0)` to decompress or to `TRUE (1)` to remove the decompressed representation from memory.	`TRUE (1)`

renderFormat	Get and set	This property determines how many bits are used to render the texture. It takes the following values:
		`#default`: the texure is rendered based on the value of the `getRendererServices().textureRenderFormat` property.
		`#rgbaWXYZ`: renders the texture using W bits for the red channel, X bits for the green channel, Y bits for the blue channel and Z bits for the alpha channel. This is available using the following possible combinations:
		`#rgba8888`
		`#rgba8880`
		`#rgba5650`
		`#rgba5550`
		`#rgba5551`
		`#rgba4444`
			`#default`

Texture methods

The pixel height and width of the 2D images you use as textures should be powers of 2 (that is, 2, 4, 8, 16, 32, and so on). If not, the image will be resized to a dimension that is a power of 2. The `scaleDown()` method lets you retain manual control over this procedure at the texture level.

Method	Description	Returns
`scaleDown()`	Reduces the height of the texture to the next lowest power of 2. This is useful for dynamically resizing textures to fit on a client machine.	Nothing

Groups

Groups have many of the same properties and methods as models, except that you need to substitute the word `group` for the word `model` when writing scripts. A group is a node that can have a parent and/or children. These can be models, lights, cameras, or other groups.

Group properties

The properties of the group determine its particular appearance and relationship to the rest of the 3D world.

Use these properties to work with groups:

Property	Access	Description	Value
`name`	Get	Unique string name.	Any string.
`parent`	Get and set	This group's parent; either another object or the 3D cast member itself.	An object or cast member.
`child.count`	Get	Number of children (but not grandchildren) of a given group.	An integer.
`transform`	Get and set	Script transform object representing this group's position and orientation relative to its parent's position and orientation. `transform.position` gives the relative position. `transform.rotation` gives the relative rotation.	Set: a transform object. Get: reference to a transform object.
`userData`	Get and set	A property list containing all properties assigned to the group. Users can add, remove, get, and set properties on this list.	The default list includes the properties assigned in the 3D modeling tool. User properties are then added.
`bounding Sphere`	Get	A list containing a vector and a floating-point value. The vector represents the position of the group in world space. The floating-point value represents the radius of the bounding sphere that contains the group and its children.	A list, with the default value of `[vector (0,0,0), 0.0]`.
`world Position`	Get	Position of the group in world coordinates. A quick shortcut for `group.getWorldTransform().position`	A vector object.
`pointAt Orientation`	Get and set	A list of two orthogonal vectors, `[objectRelativeDirection, objectRelativeUp]`, that control how the groups `pointAt()` method works.	A vector list.

Group methods

Use these methods to work with groups:

Method	Description	Returns
`addChild(`*`aNode, preserveWorld`*`)`	Adds *`aNode`* to this group's list of children. An equivalent operation is to set *`aNode.parent`* to equal *`this group`*. The *`preserveWorld`* argument is optional. It can have two values: `#preserveWorld` or `#preserveParent`. If the value is `#preserveWorld`, the world transform of the child being added remains intact. If the value is `#preserveParent`, the child's existing transform is interpreted as parent-relative.	Nothing
`child[`*`index`*`]`	Returns the child at the specified position in the index.	Script group object
`child(`*`name`*`)`	Returns the child named *`name`*.	Script group object
`clone(`*`name`*`)`	Clones a group named *`name`*, adds it to group's parent's child list, and adds it to world All children of the group are automatically cloned. This can be avoided by removing the children, performing the cloning operation, and then adding the children back. If the name is omitted or is `""`, the clone isn't added to the group palette, has no parent, and has no children. This option lets you quickly create temporary group instances.	Script group object
`cloneDeep(`*`name`*`)`	Clones both the group and the parent used by the group's children. Modifications to the clones' resource don't affect the parent. This is a more memory-intensive operation than `clone(`*`name`*`)`.	Script group object
`addtoWorld()`	Adds the group to the currently active 3D world, setting its parent as "world." Equivalent to `group.parent=member("scene").group ("world")`. All newly created groups are added to the world by default, without it being necessary to use this method.	Nothing
`remove from world`	For groups whose parent hierarchy terminates in the world, this sets their parent to `void` and removes them from the world. Otherwise it does nothing.	Nothing
`isInWorld()`	For groups whose parent hierarchy terminates in the world, the value is `TRUE (1)`.	`TRUE (1)` or `FALSE (0)`

Method	Description	Returns
`registerScript (eventName, handlerName, scriptInstance)`	Registers a handler named `handlerName` that is called in the `scriptInstance` when the member method `sendEvent()` is called with `eventName` as an argument. If `scriptInstance` is 0, a movie script handler is called. The user defines what `eventName` is. The *eventName* specified can be one of a default set of events or a user defined custom event. The default events are #collideAny, #collideWith, #animationStarted, #animationEnded, #timeMS.	Nothing
`translate (xIncrement, yIncrement, zIncrement, relativeTo)`	Moves the group by `xIncrement` along the *x*-axis, `yIncrement` along the *y*-axis, and `zIncrement` along the *z*-axis. The `relativeTo` parameter is optional. It determines how arguments are interpreted. The possible values are as follows: `#self`: the default. Increments are applied relative to the group's local coordinate system. `#parent`: increments are relative to the group's parent's coordinate system. `#world`: increments are relative to the world coordinate system. Equivalent to `#parent` if parent is the world. node (group, light, camera, or group): increments are relative to the coordinate system of the argument object.	Nothing
`translate (direction Vector, relativeTo)`	Moves the group `directionVector.length()` in the direction of the vector `directionVector`. The `relativeTo` argument is optional and defaults to `#self`.	Nothing
`translate (x,y,z, relativeTo)`	Moves the group distance *x* along the *x*-axis, distance *y* along the *y*-axis, and distance *z* along the *z*-axis. The `relativeTo` argument is optional and defaults to `#self`. This method can also be written as `translate(vector(x,y,z) relativeTo)`.	Nothing
`rotate(x,y,z, relativeTo)`	Rotates the group by *x* degrees around the *x*-axis, *y* degrees around the *y*-axis, and *z* degrees around the *z*-axis. The `relativeTo` argument is optional and defaults to `#self`. If included, it defines the coordinate space of the axes. This method can also be written as `rotate(vector(x,y,z) relativeTo)`.	Nothing
`rotate (position, axis, angle, relativeTo)`	Rotates the group around the axis vector in the specified position the specified number of degrees. The `relativeTo` argument is optional and defaults to `#self`.	Nothing

Method	Description	Returns
`pointAt(`*`worldPosition, worldUp`*`)`	Rotates the model until it points at the world-relative position worldPosition. The optional *`worldUp`* argument gives the general position of the model's Up axis. The exact position can't be determined using this method. Both the object-relative axes are defined by the `pointAtOrientation` property. Default values are an object-relative forward direction of vector (`0, 0, -1`) and an object-relative up direction of vector (`0, 1, 0`).	Nothing
`getWorldTransform()`	Calculates and returns a transform that converts object-relative positions for this group into world-relative positions.	A transform object

Modifiers

Modifiers control how a model is rendered or how it animates. They are attached to a model with the `addModifier()` method. Once a modifier has been attached, its properties can be manipulated with script. The tables that follow detail the properties of the modifiers included with Director.

Level of detail (LOD) modifier properties

The level of detail (LOD) modifier provides per-model control over the number of polygons used to render a model, by allowing you to reduce the number to a lower value, based on the model's distance from the camera. This modifier is attached to all imported models.

The LOD modifier can work in one of two ways: automatically using the distance from the camera or manually. To use manual mode, disable auto mode and then set the properties yourself.

Use these properties to work with the level of detail modifier:

Property	Access	Description	Default
`*whichModel*.lod.auto`	Get and set	`TRUE (1)` means that polygons are automatically reduced based on the distance from the camera. The fewer polygons that are drawn, the faster performance will be. The lod.bias property controls how aggressively this takes place. `FALSE (0)` means that you can control polygon reduction on a per-model basis, provided you've attached the level of detail modifier to the model. The level of detail modifier lets you override the default settings. To release level of detail data from memory once the model has been streamed in, set the `userData` property `sw3D` to `TRUE (1)`.	`TRUE (1)`.
`*whichModel*.lod.bias`	Get and set	Aggressiveness with which the level of detail is reduced when in automatic mode. A value of `0.0` is most aggressive and removes all polygons. A value of `100.00` should result in no visible degradation of the geometry. A middle level can be used to remove polygons at runtime that were not removed during authoring.	A value between `0.0` and `100.0`. The default is `100.0`.
`*whichModel*.lod.level`	Get and set	The percentage of the model resource mesh resolution to use when the automatic mode is `FALSE (0)`.	A value between `0.0` and `100.0`. The default is `100.0`.

Toon modifier properties

The toon modifier changes a model's rendering to imitate a cartoon drawing style. Only a few colors are used, and the model's shader, texture, and related properties are ignored.

Use these properties to work with the toon modifier:

Property	Access	Description	Default
whichModel.toon.style	Get and set	The following are the possible values: `#toon`: sharp transitions between available colors `#gradient`: smooth transitions between available colors `#black and white`: sharp transitions between black and white	`#gradient`
whichModel.toon.colorSteps	Get and set	Maximum number of colors available, rounded to the nearest power of `2`, with a limit of `16`	`2`
whichModel.toon.shadowPercentage	Get and set	The percentage of color steps to be used in shadows	`50`
whichModel.toon.highlightPercentage	Get and set	The percentage of color steps to be used in highlight	`50`
whichModel.toon.shadowStrength	Get and set	A floating-point value that determines shadow darkness	`1.0`
whichModel.toon.highlightStrength	Get and set	A floating-point value that determines highlight brightness	`1.0`
whichModel.toon.lineColor	Get and set	Color object describing line color	Black `(rgb 0,0,0)`
whichModel.toon.silhouettes	Get and set	`TRUE (1)` or `FALSE (0)` value indicating presence or absence of lines around silhouettes	`TRUE (1)`
whichModel.toon.creases	Get and set	`TRUE (1)` or `FALSE (0)` value indicating whether lines are drawn when mesh boundaries meet at a crease	`TRUE (1)`
whichModel.toon.creaseAngle	Get and set	A floating-point value controlling crease angle detection	`0.01`
whichModel.toon.boundary	Get and set	`TRUE (1)` or `FALSE (0)` value indicating whether lines are drawn at boundary of surface	`TRUE (1)`

Inker modifier properties

The inker modifier adds silhouette, crease, and boundary edges to an existing model. Silhouettes are edges along the border of a model. Crease edges are created when the angle between two areas of the mesh exceeds a given threshold.

Use these properties to work with the inker modifier:

Property	Access	Description	Default
whichModel. inker.lineColor	Get and set	Color object describing line color	Black (rgb 0,0,0)
whichModel. inker. silhouettes	Get and set	TRUE (1) or FALSE (0) value indicating presence or absence of lines around silhouettes	TRUE (1)
whichModel. inker.creases	Get and set	TRUE (1) or FALSE (0) value indicating whether lines are drawn when mesh boundaries meet at a crease	TRUE (1)
model.inker. creaseAngle	Get and set	A floating-point value controlling crease angle detection	0.01
whichModel. inker.boundary	Get and set	TRUE (1) or FALSE (0) value indicating whether lines are drawn at boundary of surface	TRUE (1)

Subdivision surfaces modifier properties

The subdivision surfaces (SDS) modifier causes the model to be rendered with additional geometric detail in the area of the model that the camera is currently looking at. The additional detail must be created in a third-party modeling application (usually by having the 3D application render the extra polygons during export of the model) and imported into Director along with the 3D cast member. The SDS modifier is available only for models created outside of Director. The SDS modifier should not be combined with the level of detail or toon modifier on the same model.

Use these properties to work with the SDS modifier:

Property	Access	Description	Default
whichModel.sds.enabled	Get and set	Enables/disables subdivision surfaces modifier functionality.	TRUE (1)
whichModel.sds. subdivision	Get and set	The following are the possible values: #uniform: mesh is uniformly scaled up in detail, with each face subdivided the same number of times #adaptive: additional detail is added only when there are major orientation changes and only to those areas of the mesh that are currently visible	#uniform
whichModel.sds.depth	Get and set	Maximum recursion depth, with a range of 0 to 5, to which the subdivision surfaces modifier is applied. At a value of 0, no change occurs.	1

Property	Access	Description	Default
whichModel.sds.tension	Get and set	Percentage of matching between modified and original surfaces.	65
whichModel.sds.error	Get and set	Percentage of error tolerance. This property applies only if sds.subdivision equals #adaptive.	0.0

Collision modifier properties

The collision modifier allows a model to be notified of and respond to collisions. You can access a model's collision modifier properties using syntax such as model.collision.*whichProperty*.

Detecting collisions and responding to collisions are separate tasks. If the enabled property is set to TRUE, and a script has been registered to be notified of collisions using the setCollisionCallback() method, that script instance receives a callback. However, the collision isn't resolved unless the resolve property is also set to TRUE.

This separation is deliberate and valuable: it can be important for a collision to be registered. In a game, for example, a projectile could strike a wall and the player's score could be increased. In that same game, however, you might not want the projectile to bounce off the wall. In that case, you'd set the enabled property to TRUE and set the resolve property to FALSE.

Collision notification can also be implemented by using registerScript() on a specific model in addition to using the setCollisionCallback() technique.

Use these properties to work with the collision modifier:

Property	Access	Description	Default
whichModel.collision.enabled	Get and set	TRUE (1) or FALSE (0) value indicating whether collisions between this model and other models will trigger a collision event.	TRUE (1)
whichModel.collision.resolve	Get and set	TRUE (1) or FALSE (0) value indicating whether collisions are automatically resolved. If the value is TRUE (1) and if the other model has the collision modifier applied and has enabled set to TRUE (1), the models will be moved back to the position of their original contact.	TRUE (1)

Property	Access	Description	Default
whichModel.collision.immovable	Get and set	TRUE (1) or FALSE (0) value indicating whether the model can be moved. If a model cannot be moved, the 3D Xtra can save time by not checking it for collisions with other models that also have their immovable property set to TRUE.	FALSE (0)
whichModel.collision.mode	Get and set	Indicates the geometry to be used in the collision detection algorithm. Using simpler geometry such as the bounding sphere leads to better performance. The possible values for this property are: #mesh: uses the actual mesh geometry of the model's resource. This gives one-triangle precision and is usually slower than #box or #sphere. #box: uses the bounding box of the model. This is useful for objects that can fit more tightly in a box than in a sphere, such as a wall. #sphere: is the fastest mode, because it uses the bounding sphere of the model. This is the default value for this property.	

Collision modifier methods

Use this method to work with the collision modifier:

Method	Description	Returns
whichModel.collision.setCollisionCallback(*#handlerName, scriptObjectName*)	If collision.enabled is set to TRUE (1), this method registers the script instance to receive an event when a collision occurs. If collision.enabled is set to FALSE (0), no event occurs. What happens when a collision occurs depends on the value assigned to the resolve property. You can override this value by using the collisionData.resolveA() or collisionData.resolveB() methods. The collisionData object will be the second argument passed to *#handlerName* in the specified script object *scriptObjectName*.	Nothing

Collision modifier events

These events are generated when collisions occur:

Event Name	Description
`#collideAny`	The first event called when any collision occurs.
`#collideWith`	The event called whenever a collision with a specified model occurs. It is implicitly registered for when `setCollisionCallback()` is called.

Animation modifiers

Once you've created animations in your modeling software, you apply animation modifiers to models to play them back in Director.

Director supports both model keyframe and character bone animations, and modifiers are available to enable both. Keyframe animations modify a model's `transform` properties over time. Bones animations modify the model's geometry over time. Creating bones animation in a 3D modeling application can be complex, but it results in more natural-looking movements.

The two animation types can be combined. You might, for example, combine a "run in place" bones animation with a "move around the room" keyframe animation.

Bones animations use the Bones player modifier. Keyframe animations use the Keyframe player modifier. Most methods and properties are available to both players. The bones and keyframe player modifiers, like all modifiers, can only be attached to geometric nodes (models). This is only really an issue for the keyframe player because you might think to keyframe animate a camera or a light but this isn't possible.

- Motions: A 3D cast member contains a set of motions authored in your 3D modeling application. For bones animation, each motion contains a list of tracks, and each track contains the keyframes for a particular bone. A bone is a segment of the skeleton of a model. For example, track 14 of the motion named `Run` could be named `RtKneeTrack` and move a bone named `RtKnee`. These names are defined in the 3D modeling application.
- Play list: The Bones player manages a queue of motions. The first motion in the play list is the motion that is currently playing or paused. When that motion finishes playing, it's removed from the play list and the next motion begins. Motions can be added with *`bonesPlayerOrKeyframePlayer`*`.play("run")`, which adds the motion to the top of the list, or *`bonesPlayerOrKeyframePlayer`*`.queue("motion")`, which adds it to the end of the list. Using the `play` method starts the motion immediately. The motion previously at the beginning of the play list is halted unless `autoBlend` is turned on. When you use `queue`, the motion is added to the end of the play list. Motions are removed from the play list automatically when they are complete, or you can remove them explicitly using `bonesPlayer.playNext()`.
- Motion blending: If `autoblend` is `TRUE`, an ending motion blends smoothly into the next motion using the *`bonesPlayerOrKeyframePlayer`*`.blendTime` property to determine how long the blend should be. You can control this manually by setting `bonesPlayerOrKeyframePlayer.autoBlend` to `FALSE` and using *`bonesPlayerOrKeyframePlayer`*`.blendFactor` to control the blend frame by frame.
- Motion mapping: You can create new motions by combining existing motions. For example, a walking motion could be combined with a shooting motion to produce a walk-and-shoot motion. This is available only with Bones player animations.

You can add the Keyframe player modifier at runtime to a model created in Director, but you cannot add the Bones player modifier at runtime. The Bones player modifier is automatically attached to models with bones animation exported from a 3D modeling application. Since the required bones information can't be assigned in Director, it has to exist before the model is imported into Director.

Bones player methods

Use these methods to work with bones animations:

Method	Description	Returns
`whichModel.bonesPlayer.play("name", looped, startTime, endTime, playRate, timeOffset)`	Plays the motion named `name` starting at the time `timeOffset`, with the currently playing motion being pushed down the play list. If `looped` is `FALSE (0)`, the preceding motion begins again when this motion completes. `StartTime` can be an integer number of milliseconds, or it can be the symbol `#synchronized`. Use `#synchronized` to start this new motion at the same relative time offset to it's total duration as the currently playing motion is to it's total duration. The `playRate` parameter indicates how fast to play the motion. A value of 2 doubles the speed of the motion. This value is multiplied by the value of the `bonesPlayer.playRate` property. If blending is enabled, blending begins the instant `play()` is called.	Nothing
`whichModel.bonesPlayer.playNext()`	Ends the currently playing motion, removes it from the play list, and begins the next motion. if blending is enabled, blending begins the instant `playNext()` is called.	Nothing
`whichModel.bonesPlayer.queue("name", looped, startTime, endTime, playRate, timeOffset)`	Adds the specified motion to the end of the play list. The parameters are same as those for the `play()` method.	Nothing
`whichModel.bonesPlayer.removeLast()`	Removes the most recently added motion from the play list. The motion will be removed from the play list even if it is also the currently playing motion.	Nothing
`whichModel.bonesPlayer.pause()`	Pauses the Bones player.	Nothing

Bones player properties

Use these properties to work with bones animations:

Property	Access	Description	Default
whichModel.bonesPlayer.playing	Get	TRUE (1)= playing; FALSE (0)= paused.	TRUE (1)
whichModel. bonesPlayer. playList	Get	A linear list of property lists, where each property list yields the parameters for the currently playing and queued animations. For example, [[#name: "Walk_rt_turn", #loop: 0, #startTime: 0, #endTime: 4000, #scale: 1.0000], [#name: "Walk", #loop: 1, #startTime: 0, #endTime: -1, #scale: 1.0000]].	Empty list []
whichModel. bonesPlayer. currentTime	Get and set	Current local time of motion at the top of the play list, in milliseconds. The motion's duration property tells you how long the animation lasts.	0
whichModel. bonesPlayer. playRate	Get and set	A value indicating how quickly or slowly to play back the motion. For example, a value of 2.0 doubles the speed of the motion; a value of 0.5 halves the speed of the motion. This value is multiplied by the value of the *playRate* parameter of the play or queue method.	1.0
whichModel. bonesPlayer. playList.count	Get	Current linear list of property lists, with each property list containing the name of a motion and its playback properties.	0
whichModel. bonesPlayer. rootLock	Get and set	TRUE means the model's root bone remains at its current position. The root bone is the central bone from which all other bones branch. If this property is set to TRUE during a walking motion, the model appears to walk in place.	FALSE
whichModel. bonesPlayer. currentLoopState	Get and set	A value of TRUE means the top motion in the play list loops. A value of FALSE turns off looping for the motion at the top of the play list.	FALSE
whichModel. bonesPlayer. blendTime	Get and set	Length in milliseconds of the period when blending takes place between motions. The blendTime property is linked to motion duration. Motion blending is disabled if blendTime = 0 and autoBlend = TRUE.	500

Property	Access	Description	Default
whichModel. bonesPlayer. autoBlend	Get and set	If TRUE, automatic linear blending (from 0.0 to 100.0) is applied over the blend time. Otherwise, blendTime is ignored, and the amount of blending is user-determined by the blendFactor property.	TRUE (1)
whichModel. bonesPlayer. blendFactor	Get and set	The degree of blending between motions, expressed as a floating-point value between 0.0 and 100.0. A value of 0.0 uses all the previous motion. A value of 100.0 uses all of the next motion in the play list. The blend factor can be changed frame by frame to create custom blending effects.	0
whichModel. bonesPlayer. bone[*boneID*] transform	Get and set	A transform relative to the parent bone. You can get and set the entire transform, but you can't call any methods of this property.	Depends on the bone
whichModel. bonesPlayer. bone[*boneID*] worldTransform	Get and set	A transform relative to the world coordinates. You can get and set the entire transform to move a bone.	Depends on the bone
whichModel. bonesPlayer. positionReset	Get and set	TRUE (1) = object returns to starting position at end of animation. FALSE (0) = object remains at final animation position after motion completes.	TRUE (1)

Property	Access	Description	Default
whichModel. bonesPlayer. rotationReset	Get and set	Normally a model snaps back to its original rotation after a motion finishes playing. This property maintains any or all of the rotational changes after playing is complete. The values are as follows: #none #x #y #z #xy #xz #all	#all

whichModel. bonesPlayer. lockTranslation	Get and set	Defines the axis of translation to ignore when playing back a motion.
		The values are as follows:
		#none
		#x
		#y
		#z
		#xy
		#xz
		#all
		To keep a model locked to a ground plane with the top pointing along the *z*-axis, set lockTranslation to #z.
		lockTranslation = #all is equivalent to rootLock = TRUE (1).
			#none

Bones player events

These events are generated by bones animations:

Event name	Description
#animation Started	This is a system-defined notification event triggered when a motion begins playing. If looping is on, this event is triggered only by the first playthrough. During a blend of two animations, this event is triggered as the blend begins.
#animation Ended	This is a system-defined notification event triggered when a motion ends. If looping is on, this event is triggered only by the first playthrough. If blending is on, this event is generated for the first animation when the blend is complete. There may be some latency because of the overhead of scheduling all other Director events.

Keyframe player methods

Use these methods to work with keyframe animations:

Method	Description	Returns
`whichModel.keyframePlayer.play ("name", looped, startTime, endTime, playRate, timeOffset)`	Plays the motion named `name` starting at the time `startTime`, with the currently playing motion being pushed down the play list. If `looped` is `FALSE (0)`, the preceding motion begins again when this motion completes. The `startTime` parameter can be an integer number of milliseconds, or it can be the symbol `#synchronized`. Use `#synchronized` to start this new motion at the same relative time offset to its total duration as the currently playing motion is to its total duration. The `playRate` parameter indicates how fast to play the motion. A value of 2 doubles the speed of the motion. This value is multiplied by the value of the `keyframePlayer.playRate` property. If blending is enabled, blending begins the instant `play()` is called.	Nothing
`whichModel.keyframePlayer.playNext()`	Ends the currently playing motion, removes it from the play list, and begins the next motion. If blending is enabled, blending begins the instant `playNext()` is called.	Nothing
`whichModel.keyframePlayer.queue("name", looped, startTime, endTime, playRate, timeOffset)`	Adds the specified motion to the end of the play list. The parameters are same as those for the `play()` method.	Nothing
`whichModel.keyframePlayer.removeLast()`	Removes the most recently added motion from the play list. The motion will be removed from the play list even if it is also the currently playing motion.	Nothing
`whichModel.keyframePlayer.pause()`	Pauses the Keyframe player.	Nothing

Keyframe player properties

Use these properties to work with keyframe animations:

Property	Access	Description	Returns
whichModel. keyframePlayer. playing	Get	TRUE (1)= playing; FALSE (0)= paused.	TRUE (1)
whichModel. keyframePlayer. playList	Get	A linear list of property lists, where each property list yields the parameters for the currently playing and queued animations. For example, [[#name: "Walk_rt_turn", #loop: 0, #startTime: 0, #endTime: 4000, #scale: 1.0000], [#name: "Walk", #loop: 1, #startTime: 0, #endTime: -1, #scale: 1.0000]].	Empty list []
whichModel. keyframePlayer. currentTime	Get and set	Current local time of motion at top of play list, in milliseconds.	0
whichModel. keyframePlayer. playRate	Get and set	A value indicating how quickly or slowly to play back the motion. For example, a value of 2.0 doubles the speed of the motion; a value of 0.5 halves the speed of the motion. This value is multiplied by the value of the *playRate* parameter of the play or queue method.	1.0
whichModel. keyframePlayer. playList.count	Get	Current number of motions in the play list.	0
whichModel. keyframePlayer. rootLock	Get and set	TRUE (1)= root translational component of the model remains at its referenced unanimated position (and therefore cannot disappear offstage).	FALSE (0)
whichModel. keyframePlayer. currentLoopState	Get and set	TRUE (1)= animation loops; FALSE (0)= animation plays through once.	FALSE (0)
whichModel. keyframePlayer. blendTime	Get and set	Length in milliseconds of the period when blending takes place between motions. The blendTime property is linked to motion duration. Motion blending is disabled if blendTime = 0 and autoBlend = TRUE.	500
whichModel. keyframePlayer. autoBlend	Get and set	If TRUE, automatic linear blending (from 0.0 to 100.0) is applied over the blend time. Otherwise, blendTime is ignored, and the amount of blending is user-determined by the blendFactor property.	TRUE (1)

Property	Access	Description	Returns
whichModel. keyframePlayer. blendFactor	Get and set	The degree of blending between motions, expressed as a floating-point value between 0.0 and 100.0. A value of 0.0 uses all the previous motion. A value of 100.0 uses all of the next motion. The blendFactor can be changed frame by frame to create custom blending effects.	0
whichModel. keyframePlayer. positionReset	Get and set	TRUE (1) = object returns to starting position at end of animation; FALSE (0) = object remains at final animation position, and begins again from there if looping is on.	TRUE (1)
whichModel. keyframePlayer. rotationReset	Get and set	Normally a model snaps back to its original rotation after a motion finishes playing. This property maintains any or all of the rotational changes after playing is complete. The values are as follows: #none #x #y #z #xy #xz #all	#all

whichModel. keyframePlayer. lockTranslation	Get and set	Defines the axis of translation to ignore when playing back a motion.
		The values are as follows:
		#none
		#x
		#y
		#z
		#xy
		#xz
		#all
		To keep a model locked to a ground plane with the top pointing along the *z*-axis, set lockTranslation to #z.
		LockTranslation = #all is equivalent to rootLock = TRUE (1).
			#none

Keyframe player events

These events are generated by keyframe animations:

Event name	Description
`#animation Started`	This is a system-defined notification event triggered when a motion begins playing. If looping is on, this event is triggered only by the first playthrough. During a blend of two animations, this event will be triggered as the blend begins.
`#animationEnded`	This is a system-defined notification event triggered when a motion ends. If looping is on, this event is triggered only by the first playthrough. If blending is on, this event will be generated for the first animation when the blend is complete. There may be some latency because of the overhead of scheduling all other Director events.

Mesh deform modifier properties

The mesh deform modifier lets you alter an existing model resource's geometry at runtime. You can create twist, bend, and ripple effects. Unlike other modifiers, the mesh deform modifier directly affects model resources as well as the models that use those resources. For example, if three car models share the same model resource, adding this modifier to one model and then deforming it will deform all the car models.

The mesh deform modifier is complex and is primarily useful for users with a thorough understanding of 3D geometry. However, you can take advantage of much of the modifier's potential by using only the `vertexList` property.

Use these properties to work with the mesh deform modifier:

Property	Access	Description
`*whichModel*. meshDeform. mesh.count`	Get	Returns the number of meshes in a model.
`*whichModel*. meshDeform. mesh[*index*]. vertexList`	Get and set	Returns a list of the vertices for the specified mesh. To modify the vertices in this mesh, set this property to a list of modified vertex positions, or modify individual vertices using bracket analysis.
`*whichModel*. meshDeform. mesh[*index*]. normalList`	Get and set	Returns a list of the normals for the specified mesh.
`*whichModel*. meshDeform. mesh[*index*]. texture CoordinateList`	Get and set	Returns a list of the texture coordinates for the specified mesh.
`*whichModel*. meshDeform. mesh[*index*]. face.count`	Get	Returns the number of triangular faces in a given mesh.

Property	Access	Description
whichModel. meshDeform. mesh[*index*]. face[*index*]	Get	Returns a list of three indexes into the vertex, normal, texture coordinate, and color lists. These indexes correspond to the corners of the face for the specified mesh.
whichModel. meshDeform. mesh[*index*]. face[*index*]. neighbor[*index*]	Get	Returns a list of lists describing the neighbors of a particular face of a mesh opposite the face corner specified by the neighbor index (1,2,3). If the list is empty, the face has no neighbors in that direction. If the list contains more than one list, the mesh is nonmanifold. This is rare. Usually the list contains four integer values. The first value is for the index into the mesh[] list, where the neighbor face lives. The second is FaceIndex, the index of the neighbor face in that mesh. The third is vertexIndex, the index within the neighbor face. The last is for Flipped, which describes whether the neighbor face is oriented in the same (0) or the opposite (1) way as the original face.
whichModel. meshDeform. face.count	Get	Returns the total number of faces in the model, which is equivalent to the sum of all the *model*.meshDeform.mesh[*index*].face.count properties in a given model.

Motions

Motions are simply animations that have been predefined in a 3D modeling application. They are included in the file that's exported from the 3D application and imported into Director.

Motions can be reused on any model in the 3D cast member, as long as the motion is appropriate to the geometry of the model. The properties and commands that follow can be used with either keyframe or bones motions.

Motion properties

Use these properties to work with motions:

Property	Access	Returns
name	Get	Name of motion.
duration	Get	Time in milliseconds motion needs to play to completion.
type	Get	The type of motion with the following values: #keyFrame: suitable for use with the Keyframe player #bones: suitable for use with the Bones player #none: no mapping has been made for this motion The default is #none.

Motion methods

Use this method to work with motions:

Method	Description
`map(motion, "bone name")`	Maps the given motion into the current motion beginning at the named bone. If no bone name is specified, the mapping begins at the root bone. The `map()` method will replace any motion tracks mapped previously to the specified bone and all of its children. Motion mapping has no effect on motions that are already on a play list. The `map()` method does not work with keyframe animations.

About lights and cameras

Lights illuminate the 3D world and the models in it. Without lights, the world exists, and actions can take place, but users see nothing. You can add lights to your 3D world in your 3D modeling application or with the Property inspector. For information about the Property inspector, see Chapter 14, “3D Basics,” on page 303. You can also add and remove lights, change their color or position, and manipulate their parent-child relationships using methods and properties. Those methods and properties are detailed here. You can find the same lighting methods and properties, with more detailed syntax and coding examples, in the Scripting Reference topics in the Director Help Panel.

Cameras act as windows into a 3D world. Each camera that exists in a 3D cast member offers a different view into it, and each sprite that uses a 3D cast member uses one of these cameras. A camera’s position can be moved with the Property inspector or the Shockwave 3D window. You can also use the Director 3D behaviors or script to manipulate camera positions. For information about the Property inspector and the Shockwave 3D window, see Chapter 14, “3D Basics,” on page 303. For information about behaviors, see Chapter 15, “The 3D Cast Member, 3D Text, and 3D Behaviors,” on page 315. More complex manipulations require the use of methods and properties. Accessing the properties and methods of a light or camera requires that the member be on the Stage or explicitly and completely loaded with the `preLoad()` method. When using the `preLoad()` method, you can verify that the load is complete by testing whether `member.state = 4` (loaded).

Lights and cameras have the same `transform` methods and parent-child properties as models and groups. Lights and cameras can be added, deleted, cloned, moved, and rotated in the same ways as models and groups. You can access their names, parents, children, and other properties in the same way you would with models and groups. However, there are some important differences, which arise from the specific roles that lights and cameras play in the 3D world.

Light properties

Use these properties to work with lights:

Property Name	Access	Description	Default
`name`	Get	Unique name of this light. If the light was exported from a 3D modeling package, the name is the name assigned there.	None
`parent`	Get and set	The model, light, camera, or group that is this light's parent. If the light has no parent, it cannot contribute light.	`Group ("World")`
`child.count`	Get	Number of immediate children (no grandchildren) that the light has.	0
`transform`	Get and set	Transform object representing light's position relative to its parent's transform. The `transform.position` gives the relative position; `transform.rotation` gives the relative rotation.	Whichever transform is required to represent the light's position and orientation in space.
`userData`	Get and set	A property list associated with this light. The list defaults to the properties assigned in the 3D modeling tool, but users can add or delete properties at any time.	Properties assigned in 3D modeling tool
`type`	Get and set	The kind of light this is. Must be one of the following: `#ambient`: applied to all sides of the model `#directional`: applied to those parts of the light facing the light's direction. Distance to the light isn't important. `#point`: Like a bare light bulb, omnidirectional and illuminating all parts of the model facing the light `#spot`: Like a spotlight, casting light on model parts that face it, with brighter illumination the closer the model is. Similar to `#directional`, except that the apparent distance from the light is taken into account.	None
`color`	Get and set	Color object defining color and intensity. Ranges from color(255,255,255), which is pure white to color(0,0,0), which is no light at all.	`color(191,191, 191)`

Property Name	Access	Description	Default
`spotAngle`	Get and set	Angle of the light's projection cone. If type equals `#spot`, setting a value less than the umbra causes a "property not found" error.	`90.0`
`attenuation`	Get and set	A three-value vector controlling the constant, linear, and quadratic attenuation factors for spotlights.	`vector (1.0,0.0,0.0)`
`specular`	Get and set	`TRUE (1)`/`FALSE (0)` value that controls whether or not the light produces specular effects on surfaces. The property is ignored for ambient lights. Although `TRUE (1)` is the default, switching to `FALSE (0)` may improve performance.	`TRUE (1)`
`spotDecay`	Get and set	`TRUE (1)`/`FALSE (0)` value that controls whether or not spotlight intensity falls off with camera distance.	`FALSE (0)`
`pointAt Orientation`	Get and set	Two orthogonal vectors (`objectRelativeDirection` and `objectRelativeUp`) controlling how the light's `pointAt` method works.	`None`
`boundingSphere`	Get	A list containing a vector and a floating-point value, with the vector representing the position and the value the radius of a bounding sphere surrounding the light and all its children.	`[vector (0,0,0), 0.0]`
`worldPosition`	Get and set	Position of the light in world coordinates. Shortcut for the method `node.getWorldTransform ().position`.	Vector object

Light methods

Use these methods to work with lights:

Method	Description	Returns
`addChild (`*`aNode, preserveWorld`*`)`	Adds the node *`aNode`* to this light's list of children. An equivalent operation is to set *`aNode.parent = this light`*. The *`preserveWorld`* argument is optional. It can have two values: `#preserveWorld` or `#preserveParent`. If the value is `#preserveWorld`, the world transform of the child being added remains intact. If `#preserveParent`, the child's transform is interpreted as remaining parent-relative.	Nothing
`child[`*`index`*`]`	Returns the child at the specified position in the index.	Light object
`child(`*`name`*`)`	Returns a reference to the named child.	Light object

Method	Description	Returns
`clone(`*`name`*`)`	Clones a light named *`name`*, adds it to light's parent's child list, and adds it to the world. All children of the light are automatically cloned.This can be avoided by removing the children, performing the cloning operation, and then adding the children back. If the name is omitted or is `""`, the clone isn't added to the light palette, has no parent, and has no children. This option lets you quickly create temporary light instances.	Light object
`cloneDeep (`*`name`*`)`	Clones both the light and all resources used by the light's children.	Light object
`addtoWorld()`	Adds light to currently active 3D world, setting its parent as `"world"`. All newly created lights are added to the world by default, without it being necessary to use this method.	Nothing
`removeFrom World()`	For lights whose parent hierarchy terminates in the world, this sets their parent to `void` and removes them from the world. Otherwise it does nothing.	Nothing
`isInWorld()`	Returns a Boolean value indicating if the light is currently in the world (TRUE) or not (FALSE). This is useful for detecting if a given node's parent-heirarchy tree terminates with the world group object or not.	`TRUE (1)` or `FALSE (0)`
`registerScript (`*`eventName, handlerName, scriptInstance`*`)`	Registers a handler named *`handlerName`* that is called in the *`scriptInstance`* when the member method *`sendEvent()`* is called with *`eventName`* as an argument. If *`scriptInstance`* is 0, a movie script handler is called. The user defines what *`eventName`* is. The *eventName* specified can be one of a default set of events or a user defined custom event. The default events are #collideAny, #collideWith, #animationStarted, #animationEnded, #timeMS.	Nothing
`translate (`*`xIncrement, yIncrement, zIncrement, relativeTo`*`)`	Moves the light forward by *`xIncrement`* along the *x*-axis, *`yIncrement`* along the *y*-axis, and *`zIncrement`* along the *z*-axis. The *`relativeTo`* parameter is optional. It determines how arguments are interpreted. The possible values are as follows: `#self`: the default. Increments are applied relative to the light's local coordinate system. `#parent`: increments are relative to the light's parent's coordinate system. `#world`: increments are relative to the world's coordinate system. Equivalent to `#parent` if the parent is the world. `node` (model, light, camera, or group): increments are relative to the argument's coordinate system.	Nothing

Method	Description	Returns
`translate(x,y,z, relativeTo)`	Moves the light distance *x* along the *x*-axis, distance *y* along the *y*-axis, and distance *z* along the *z*-axis. The *`relativeTo`* argument is optional and defaults to `#self`. This method can also be written as translate `(vector(x,y,z) relativeTo)`	Nothing
`rotate(x,y,z, relativeTo)`	Rotates the light by *x* degrees around the *x*-axis, *y* degrees around the *y*-axis, and *z* degrees around the *z*-axis. The *`relativeTo`* argument is optional and defaults to `#self`. If included, it defines the coordinate space of the axes. This method can also be written as rotate `(vector(x,y,z) relativeTo)`	Nothing
`rotate (position, axis, angle, relativeTo)`	Rotates the light around the axis vector in the specified position the specified number of degrees. The *`relativeTo`* argument is optional and defaults to `#self`.	Nothing
`pointAt(worldPosition, worldUp)`	Points the node's "front" at the world position and then tries to align the node's "up" with the worldUp specified, and that the node's "front" and "up" are determined by the node's *`pointAtOrientation`* property. Both the object-relative axes are defined by the *`pointAtOrientation`* property. Default values are an object-relative forward direction of vector (`0, 0, -1`) and an object-relative up direction of vector (`0, 1, 0`).	Nothing
`getWorldTransform()`	Calculates and returns a transform that converts object-relative positions for this light into world-relative positions.	A transform object

Cameras

Cameras act as view ports into the 3D world. By default, a newly added camera's view is positioned at the world's origin, the vector (0,0,0), and looks down the negative *z*-axis. Changing a camera's `transform` property affects the camera's position and orientation. When a 3D sprite is created from a 3D cast member, the sprite uses one of the cast member's cameras. Changing the camera that the sprite is using changes what's seen in the sprite.

Cameras can also have overlays and backdrops. Overlays are 2D images drawn in front of the camera's lens. Backdrops are 2D images drawn behind the 3D scene. Backdrops provide a background image for the scene regardless of which way the camera is pointing.

Camera properties

Use these properties to work with cameras:

Property Name	Access	Description	Default
`name`	Get and set	Unique name of this camera. If the camera was exported from a 3D modeling program, the name is the name assigned there.	None
`parent`	Get and set	The model, light, camera, or group that is this light's parent. If the camera has no parent, it cannot contribute light.	`group ("world")`
`child.count`	Get	Number of immediate children (no grandchildren) the camera has.	`0`
`transform`	Get and set	Transform object representing camera's position relative to its parent's transform. The `transform.position` property gives the relative position; `transform.rotation` gives the relative rotation.	Identity transform
`userData`	Get and set	A property list associated with this camera. The list defaults to the properties assigned in the 3D modeling tool, but users can add or delete properties at any time.	Properties assigned in 3D modeling tool
`hither`	Get and set	A specified distance from the camera that defines the near *z*-axis clipping of the view frustum. Objects closer than hither are not drawn.	`5.0`
`yon`	Get and set	A specified distance from the camera that defines the far *z*-axis clipping of the view frustum. Objects farther than yon are not drawn.	`3.403e38`
`rect`	Get and set	The rectangle controlling the screen size and position of the camera, with the coordinates given relative to the upper left corner of the sprite.	`rect(0,0,320,200)`
`projectionAngle`	Get and set	The vertical projection angle of the view frustum.	`30.0`

Property Name	Access	Description	Default
`colorBuffer.clearAtRender`	Get and set	`TRUE (1)` or `FALSE (0)` value indicating whether color buffer is or isn't cleared out after each frame. If value is set to `FALSE (2)`, the effect is similar to the trails ink effect, although it requires a semi-transparent model behind the whole scene.	`TRUE (1)`
`colorBuffer.clearValue`	Get and set	Color object defining color used to clear out buffer if `colorBuffer.clearAtRender` is `TRUE (1)`.	`color(0,0,0)`
`fog.enabled`	Get and set	`TRUE (1)` or `FALSE (0)` value indicating whether camera adds fog to the scene.	`FALSE (0)`
`fog.near`	Get and set	Distance to start of fog.	`0.0`
`fog.far`	Get and set	Distance to maximum fog intensity.	`1000.0`
`fog.color`	Get and set	Color object describing fog color.	`color(0,0,0)`
`fog.decayMode`	Get and set	How fog varies between near and far, with the following possible values: `#linear`: density is linearly interpolated between `fog.near` and `fog.far`. `#exponential`: `fog.far` is saturation point; fog.near is ignored. `#exponential2`: `fog.near` is saturation point; `fog.far` is ignored.	`#exponential`
`projection`	Get and set	Method of determining the vertical field of view, which must be of type `#perspective` or `#orthographic`.	`#perspective`
`fieldOfView`	Get and set	A floating-point value specifying the vertical projection angle in degrees.	`30.0`
`orthoheight`	Get and set	The number of perpendicular world units that fit vertically into the sprite.	`200.0`
`rootNode`	Get and set	Property controlling which objects are visible in a particular camera's view. Its default value is the world, so all cameras you create show all nodes within the world. If, however, you change `rootNode` to be a particular node within the world, a sprite of the cast member will show only the root node and its children.	`group ("world")`

Property Name	Access	Description	Default
`overlay[`*`index`*`].loc`	Get and set	Location, in pixels, of the overlay, as measured from the upper left corner of the sprite's `rect` to the `overlay[`*`index`*`].source's regPoint`.	`point(0,0)`
`overlay[`*`index`*`].source`	Get and set	Texture object used as the source for this overlay.	None
`overlay[`*`index`*`].scale`	Get and set	Scale value used by a specific overlay in the camera's list of overlays.	`1.0`
`overlay[`*`index`*`].regPoint`	Get and set	Texture-relative rotation point, similar to a sprite's `regPoint`.	`point(0.0)`
`overlay[`*`index`*`].rotation`	Get and set	Rotation value used by a specific overlay in the camera's list of overlays.	`0, 0`
`overlay[`*`index`*`].blend`	Get and set	Blend value used by a specific overlay in the camera's list of overlays. `100` is fully opaque; `0` is fully transparent.	`100.0`
`overlay.count`	Get and set	Number of overlays in use on this sprite.	`0`
`backdrop[`*`index`*`].loc`	Get and set	Location, in pixels, of the backdrop, as measured from the upper left corner of the sprite's rect to the `backdrop[`*`index`*`].source's regpoint`.	`point(0,0)`
`backdrop[`*`index`*`].source`	Get and set	Texture object used as the source for this backdrop.	None
`backdrop[`*`index`*`].scale`	Get and set	Scale value used by a specific backdrop in the camera's list of backdrops.	`1.0`
`backdrop[`*`index`*`].rotation`	Get and set	Rotation value used by a specific backdrop in the camera's list of backdrops.	`0.0`
`backdrop[`*`index`*`].regPoint`	Get and set	Texture-relative rotation point, similar to a sprite's `regPoint`.	`point(0,0)`
`backdrop[`*`index`*`].blend`	Get and set	Blend value used by a specific backdrop in the camera's list of backdrops.	`100.0`
`backdrop.count`	Get	Number of backdrops in use on this sprite.	`0`

Property Name	Access	Description	Default
`boundingSphere`	Get	A list containing a vector and a floating-point value, with the vector representing the position and the value the radius of a bounding sphere surrounding the camera and all its children.	`[vector (0,0,0), 0.0]`
`worldPosition`	Get and set	Position of the camera in world coordinates. Shortcut for the method `node.getWorldTransform ().position`.	Vector object

Camera methods

Use these methods to work with cameras:

Method	Description	Returns
`addChild(`*`aNode, preserveWorld`*`)`	Adds *`aNode`* to this camera's list of children. An equivalent operation is to set `aNode.parent` to equal *`thisCamera`*. The *`preserveWorld`* argument is optional.I t can have two values: `#preserveWorld` or `#preserveParent`. If the value is `#preserveWorld`, the world transform of the child being added remains intact. If `#preserveParent`, the child's transform is interpreted as remaining parent-relative.	Nothing
`child[`*`index`*`]`	Returns the child at the specified position in the index.	Camera object
`child(`*`name`*`)`	Returns the child named *`name`*.	Camera object
`clone(`*`name`*`)`	Clones a camera named *`name`*, adds it to the cameras's parent's child list, and adds it to the world. All children of the camera are automatically cloned.This can be avoided by removing the children, performing the cloning operation, and then adding the children back. If the name is omitted or is `""`, the clone isn't added to the camera palette, has no parent, and has no children. This option lets you quickly create temporary camera instances.	Camera object
`cloneDeep(`*`name`*`)`	Clones both the camera and all resources used by the camera's children.	Camera object
`addtoWorld()`	Adds a camera to the currently active 3D world, setting its parent as `"world"` Equivalent to setting the camera's parent to the world group. All newly created cameras are added to the world by default, without it being necessary to use this method.	Nothing
`removeFrom World()`	For cameras whose parent hierarchy terminates in the world, this sets their parent to `void` and removes them from the world. Otherwise it does nothing.	Nothing

Method	Description	Returns
`isInWorld()`	Returns a Boolean value indicating if the camera is currently in the world (TRUE) or not (FALSE). This is useful for detecting if a given node's parent-heirarchy tree terminates with the world group object or not.	`TRUE (1)` or `FALSE (0)`
`registerScript (eventName, handlerName, scriptInstance)`	Registers a handler named *handlerName* that is called in the *scriptInstance* when the member method *sendEvent()* is called with *eventName* as an argument. If *scriptInstance* is 0, a movie script handler is called. The user defines what *eventName* is. The *eventName* specified can be one of a default set of events or a user defined custom event. The default events are #collideAny, #collideWith, #animationStarted, #animationEnded, #timeMS.	Nothing
`translate (directionVector, relativeTo)`	Moves the camera *directionVector.length()* in the direction of the directionVector. The relativeTo argument is optional and defaults to `#self`.	Nothing
`translate(x,y,z, relativeTo)`	Moves the camera distance *x* along the *x*-axis, distance *y* along the *y*-axis, and distance *z* along the *z*-axis. The *relativeTo* argument is optional and defaults to `#self`. This method can also be written as `translate(vector(x,y,z) relativeTo)`.	Nothing
`rotate(x,y,z, relativeTo)`	Rotates the camera by *x* degrees around the *x*-axis, *y* degrees around the *y*-axis, and *z* degrees around the *z*-axis. The *relativeTo* argument is optional and defaults to `#self`. If included, it defines the coordinate space of the axes. This method can also be written as rotate`(vector(x,y,z) relativeTo)`	Nothing
`rotate (position, axis, angle, relativeTo)`	Rotates the camera around the axis vector in the specified position the specified number of degrees. The *relativeTo* argument is optional and defaults to `#self`.	Nothing
`pointAt(worldPosition, worldUp)`	Points the camera's "front" at the world Position and then tries to align the node's "up" with the worldUp specified, and that the node's "front" and "up" are determined by the node's *pointAtOrientation* property. Both the object-relative axes are defined by the *pointAtOrientation* property. Default values are an object-relative forward direction of vector (`0, 0, -1`) and an object-relative up direction of vector (`0, 1, 0`).	Nothing
`getWorldTransform()`	Calculates and returns a transform that converts object-relative positions for this camera into world-relative positions.	A transform object

CHAPTER 17
Controlling the 3D World

Macromedia Director MX 2004 provides powerful methods for overall control of the three-dimensional (3D) world, including Lingo and JavaScript syntax for handling new 3D-generated events, selecting models (picking), vector math operations, and transforms. In addition, the properties and methods of the Director global renderer services object supply common rendering properties for all 3D sprites and cast members. Finally, 3D cast member and sprite properties and methods allow additional control of their content during playback.

The methods and properties that you see here in tabular form are also available with accompanying syntax, definitions, and examples in the Scripting Reference topics in the Director Help Panel.

3D Lingo or JavaScript syntax events

Event handling lets you use the `registerForEvent` method to specify a handler to run when a particular event occurs within a specific cast member. With the `registerForEvent` method, you specify the type of event that will trigger the handler, the handler name, and the script object that contains the handler. The Lingo or JavaScript syntax object can be a child script, an instance of a behavior attached to a sprite, or any other script. If the object isn't specified, the handler is called in the first movie script in which it is found.

Use these methods to set up event handling:

Method	Description	Returns
`registerForEvent (`*`eventName, handlerName, scriptInstance, model`*`)`	The event *`eventName`* is one of the following: `#collideAny`: Called when any collision occurs. `#collideWith`: Called when a collision with a specific model occurs and implicitly registered when `setCollisionCallback(...)` is called. Equivalent to calling `model.collision.setCollision Callback`. `#timeMS`: Sets up a time-based simulation callback using the format `registerForEvent (timeMS,` *`handlerName, scriptInstance`*`)` *`begin, period, repetitions`*. The *`begin`* and *`period`* arguments are in milliseconds. If *`repetitions`* is set to `0`, the simulation continues indefinitely. `#animationStarted`: Called whenever a keyFrame or bones motion begins. `#animationEnded`: Called whenever a keyFrame or bones motion ends. Any user-defined symbol: Registers to receive any user-defined event sent from a SendEvent call.	`TRUE (1)` if the operation succeeds. `FALSE (0)` if the operation fails.
`unregister AllEvents()_`	Unregisters all events.	Nothing. A script error is generated if the operation fails.
`sendEvent (`*`eventName, arg1,arg2...`*`)`	Sends an event named *`eventName`* to all scripts registered to receive it. Similar to `sendAllSprites()` except that the event is delivered only to scripts that are registered to receive it.	Nothing. A script error is generated if the operation fails.

Collisions

By attaching the collision modifier (`#collision`) to a model, you can enable that model to automatically respond to collisions with other models. By using the properties of the collision modifier, you can control the details of how the model responds to collisions. For more information about collisions, see "Modifiers" on page 366.

Collision properties

When a collision occurs, it generates a `collideAny` event or a `collideWith` event.The `collideWith` event passes an argument to the handler that is declared with the `registerForEvent`, `registerScript`, or `setCollisionCallBack` method. This argument is called a `collisionData` object and contains a property list with detailed information about the collision.

These properties are included in the `collisionData` object passed to the handler:

Property	Access	Description
`modelA`	Get	One model in the collision. If the script includes registration for collision with a particular model, `modelA` is that model.
`modelB`	Get	The second model in the collision.
`pointOf Contact`	Get	Vector describing world-space location of collision. Available only if the collision has been resolved. Occurs if the model's collision modifier `resolve` property is `TRUE (1)` or either the `collisionData resolveA()` or `collisionData resolveB()` method is called.
`collision Normal`	Get	Vector indicating direction of collision. Available only if the collision has been resolved. Occurs if the model's collision modifier `resolve` property is `TRUE (1)` or either the `collisionData resolveA()` or `collisionData resolveB()` method is called.

Collision methods

Collision methods let you override certain aspects of the default behavior set for models during collisions. If neither of the models involved in the collision has `resolve` set to `TRUE`, you can manually resolve the collision using `resolveA(true)` for model A or `resolveB(true)` for model B.

Method	Description	Returns
`collisionData.resolveA (`*`trueOrFalse`*`)`	Resolves collision for model A.	Nothing
`collisionData.resolveB (`*`trueOrFalse`*`)`	Resolves collision for model B. If the argument is `FALSE (0)`, the collision won't be resolved. This overrides the `collision.resolve` property, if any, set for model B.	Nothing

Selecting models

Selecting models (picking) refers to clicking on models in a 3D cast member. Because models are objects that exist within a 3D cast member and a 3D sprite, they are not normally sensitive to mouse clicks. Normally it is only the entire sprite that is sensitive to mouse clicks.

You can use scripts to determine specifically which models have been clicked when a user clicks within a 3D sprite. In practice, this allows for changing model positions to make it appear that an action such as a button being pushed or a door being opened has taken place, or to allow the user to select or drag an object. Picking can be accomplished by using cast member or camera methods.

Camera methods

These camera methods let you determine which models have been clicked when a user clicks the mouse within a 3D sprite. You can also translate coordinates in 3D space to coordinates in 2D sprite space and vice versa.

Method	Description	Returns
`worldSpaceToSpriteSpace(`*`vector`*`)`	Returns the 2D sprite-space coordinates of a point from a 3D world vector.	A point.

`spriteSpaceToWorldSpace(`*`point`*`)`	The opposite of the `worldSpaceToSpriteSpace(`*`vector`*`)`, this method returns a world-space vector on the camera's projection plane from a sprite-space point.
	Multiple world-space positions can map to the same sprite-space point. A round-trip
	`y=worldSpaceToSpriteSpace(x)`
	`z=worldSpaceToSpriteSpace(y)`
	won't necessarily result in `x=z`.
		A vector.

`modelUnderLoc(point)`	Returns the first model intersected by a ray from a location *`point`* within the `rect` of the sprite using this camera. The location *`point`* is relative to the upper left corner of the sprite, in pixels. The ray is cast forward in the direction the camera is looking. This method is useful for picking in conjunction with an `onMouseDown` handler. For accuracy, be sure to subtract the upper left corner of the sprite's `loc` from the `mouseLoc`.	The first model intersected by the ray. A value of `void` means there is no model under the ray.
`modelsUnderLoc(point, optionalMaxNumberOfModels)`	Returns a list of all models intersected by a ray from a location *`point`* within the `rect` of the sprite using this camera. The location *`point`* is relative to the upper left corner of the sprite, in pixels. The ray is cast forward in the direction the camera is looking. This method is useful for picking in conjunction with an `onMouseDown` handler. For accuracy, be sure to subtract the upper left corner of the sprite's `loc` from the `mouseLoc`.	The first model intersected by the ray or a list of up to the specified maximum. If no maximum is specified, the method returns all models under the ray. `A value of void` means there is no model under the ray.
`modelsUnderRay(locationVector, directionVector, optionalMaxNumberOfModels)`	Returns a list of models under the ray starting at the vector *`locationVector`* and pointing down the vector *`directionVector`*, with both vectors specified in world-relative coordinates.	The first model intersected by the ray plus a list of up to the specified maximum number of models. If the maximum number of models isn't specified, all models the ray intersects are returned.

Vector math

A 3D vector describes both direction and location in 3D space. Vector objects include floating-point values for position along each of the *x*-, *y*-, and *z*-axes. Vectors can be node-relative or world-relative. If they are node-relative, their *x*, *y*, and *z* values are relative to the position of the node. If they are world-relative, their *x*, *y*, and *z* directions are relative to the world.

Vector math operations perform calculations using each of the *x*, *y*, and *z* values. These calculations are useful for performing intelligent movement and rotation of models.

Vector creation methods

Use these methods to create vectors:

Method	Description	Returns
`vector (x,y,z)`	Creates a vector from arguments representing all axes.	A vector object
`random Vector()`	Creates a vector describing a randomly chosen point on the surface of a unit sphere. Differs from `vector(random(10)/10.0, random(10)/10.0, random(10)/ 10.0)` because the `randomVector()` method always results in a unit vector.	A unit vector

Vector properties

Use these properties to work with vectors:

Property	Access	Description
`magnitude`	Get	The magnitude of the vector. Equivalent to the length of the vector.
`length`	Get	The length of the vector. Equivalent to the magnitude of the vector.
`[`*`index`*`]`	Get and set	Returns the value of a vector at a specified point in an index between 1 and 3.
`x`	Get and set	The *x* component of a vector.
`y`	Get and set	The *y* component of a vector.
`z`	Get and set	The *z* component of a vector.

Vector methods

Use these methods to work with vectors:

Method	Description	Returns
`normalize()`	Normalizes the vector by modifying it into a unit vector of length 1. This is done by dividing each component of the vector by the vector's original length. That original length is the square root of the sum of the squares of each component.	Nothing. Vector is modified.
`getNormalized()`	Returns a normalized version of the vector.	A new vector object.
`dot(`*`vector2`*`)`	Returns the dot (inner) product of the first vector and the second vector (*`vector2`*). If both vectors are of unit length, the result is the cosine of the angle between the two vectors.	Dot product of the two vectors.
`angleBetween(`*`vector2`*`)`	Returns the angle between *`vector`* and *`vector2`*, in degrees.	Value of the angle in degrees.
`cross(`*`vector2`*`)` or `crossProduct(`*`vector2`*`)` or `perpendicularTo(`*`vector2`*`)`	Returns a vector perpendicular to the original vector and to *`vector2`*.	A new vector object.
`distanceTo(`*`vector2`*`)`	Returns the distance between *`vector`* and *`vector2`*. If these vectors represent positions in the 3D world, this is the distance between them.	Floating-point value of distance.
`duplicate()`	A copy of the vector.	A new vector object.

Vector binary operations

Use these syntaxes to perform additional vector math calculations:

Note: JavaScript does not support these operations for vector objects. In JavaScript, you must write the code to perform the vector math calculations using the vectors' *`x`*, *`y`*, and *`z`* coordinates.

Operator	Description	Returns
`vector1 +vector2`	Returns a new vector equaling `vector1 +vector2` for *`x`* equaling 1 through 3.	A new vector object
`vector1 -vector2`	Returns a new vector equaling `vector1 -vector2` for *`x`* equaling 1 through 3.	A new vector object
`vector1* vector2`	Returns the product of the two vectors.	A floating-point value
`vector1/vector2`	Not supported.	`0`
`vector2*scalar`	Returns a new vector equaling `vector2 * scalar` for *`x`* equaling 1 through 3.	A new vector object

Operator	Description	Returns
`vector2/scalar`	Returns a new vector equaling `vector2/scalar`.	A new vector object
`transform* vector`	Returns a new vector resulting from applying the positional and transformation changes defined by *`transform`* to *`vector`*. Note that vector*transform is an invalid operation.	A new vector object
`scalar-vector1`	Returns a new vector equaling `scalar-vector1`.	A new vector object
`vector1-scalar`	Returns a new vector equaling `vector1-scalar`.	A new vector object
`scalar + vector1`	Returns a new vector equaling `scalar + vector1`.	A new vector object
`vector1 + scalar`	Returns a new vector equaling `vector1 + scalar`.	A new vector object

Transforms

A transform is a data object describing a model's position, orientation, and scale in the 3D world. Transform methods can be used to move a given vector, light, camera, or model from its current location to a new position and/or orientation.

Transform creation method

Use the `transform()` method to create a new `transform` data object:

Method	Description	Returns
`transform()`	Creates a new transform initialized as the identity transform. The identity transform has no rotation and a vector position of `(0,0,0)`	A new transform object

Transform properties

Use these properties to work with transforms:

Property	Access	Description	Default
`position`	Get and set	Script vector object describing the position of a transform with the value `vector(`*`xOffset, yOffset, zOffset`*`)`. A model.transform position represents the model's position in relation to its parent.	`vector (0,0,0)`
`scale`	Get and set	Script vector object describing the x, y, and z scale of the transform with the vector value `vector(`*`xScale, yScale, zScale`*`)`. Scaling is always applied model-relative.	`vector (1,1,1)`

Property	Access	Description	Default
`rotation`	Get and set	Script vector object describing the *`xRotation`*, *`yRotation`*, and *`zRotation`* components of the transform with the value `vector(`*`xRotation`*`, `*`yRotation`*`, `*`zRotation`*`)`, with the rotation values defined in degrees. This value can vary because of the permissible types of transform operation. For example, `translate` followed by `rotate` gives a different value than `rotate` followed by `translate`, and the results can't be differentiated after the fact from the rotational information alone. The `rotate()` and `preRotate()` methods are the preferred way to modify a transform's orientation. Rotation is generally relative to the object's original orientation at the start of the movie.	`vector (0,0,0)`
`axisAngle`	Get and set	A list including a vector and a floating-point value that describes this transform's rotation as an axis/angle pair. The vector represents the direction, and the angle represents the rotation around the vector.	`[vector (1.0000, 0.0000, 0.0000) A]`

`x axis`	Get and set	A vector representing the transform's canonical x axis in transform space. Example:
		`transform.identity()`
		`transform.rotate(0,90,0))`
		`put transform.xaxis`
		`--vector(0,0,-1)`
		Canonical means reduced to the simplest possible mathematical expression.
			`vector (1,0,0)`

`y axis`	Get and set	A vector representing the transform's canonical y axis in transform space. Example:
		`transform.identity()`
		`transform.rotate(90,0,0)`
		`put transform.yaxis`
		`--vector(0,0,1)`
			`vector (0,1,0)`

`z axis`	Get and set	A vector representing the transform's canonical z axis in transform space. Example:
		`transform.identity()`
		`transform.rotate(0,90,0)`
		`put transform.zaxis`
		`--vector(1,0,0)`
			`vector (0,0,1)`

Transform methods

Use these methods to work with transforms:

Method	Description	Returns

`rotate` (*`xAngle, yAngle, zAngle`*)	Applies a rotation transformation after the current transformation:
	`model.transform.identity()`
	`model.transform.translate(100,0,0)`
	`model.transform.rotate(0,0,90)`
	After this series of transformations, performed in this order, the model's local origin will be at `(0,100,0)`, assuming the model's parent is the world.
		Nothing

`preRotate` (*`xAngle, yAngle, zAngle`*)	Applies a rotation transformation before the current transformation:
	`model.transform.identity()`
	`model.transform.translate(100,0,0)`
	`model.transform.preRotate(0,0,90)`
	After this series of transformations, performed in this order, the model's local origin will be at `(100,0,0)`, assuming the model's parent is the world.
		Nothing

`rotate` (*`point, vector, angle`*)	Similar to `transform.rotate(`*`xAngle, yAngle, zAngle`*`)`, except that the arguments are two vectors specifying an axis of rotation as a point and a vector, plus an angle specifying the clockwise rotation around that axis:
	`model.transform.identity()`
	`model.transform.translate(-50,0,0)`
	`model.transform.rotate(vector(100,0,0) vector(0,1,0))`
	After this series of transformations, performed in this order, the model's local origin will be at `(250,0,0)`, assuming the model's parent is the world.
		Nothing

`preRotate` (*`point, vector, angle`*)	Similar to `transform.preRotate(`*`xAngle, yAngle, zAngle`*`)`, except that the arguments are two vectors specifying an axis of rotation as a point and a vector, plus an angle specifying the clockwise rotation around that axis.
	`model.transform.identity()`
	`model.transform.translate(-50,0,0)`
	`model.transform.preRotate(vector(100,0,0)`
	`vector(0,1,0))`
	After this series of transformations, performed in this order, the model's local origin will be at `(150,0,0)`, assuming the model's parent is the world.
		Nothing

`translate` (*`xIncrement,yIncrement,zIncrement`*)	Translates the position of the transform relative to the transform's current orientation:
	`model.transform.identity()`
	`model.transform.rotate(0,90,0)`
	`model.transform.translate(100,0,0)`
	After this series of transformations, performed in this order, the model's local origin will be at `(100,0,0)`, assuming the model's parent is the world.
		Nothing

Method	Description	Returns

`preTranslate (xIncrement, yIncrement, zIncrement)`	Translates the position of the transform before the current transformation:
	`model.transform.identity()`
	`model.transform.rotate(0,90,0)`
	`model.transform.translate(100,0,0)`
	After this series of transformations, performed in this order, the model's local origin will be at `(0,0,100)`, assuming the model's parent is the world.
		Nothing

`multiply (transform2)`	Alters the original transform by applying the positional/ rotational/scaling effects of *`transform2`* to the original transform. If `transform2` describes a rotation of 90° around the x axis and this transform describes a translation of 100 units in the y axis, `transform.multiply(`*`transform2`*`)` alters this transform so that it describes a translation followed by a rotation.	Nothing
`preMultiply (transform2)`	Alters the original transform by preapplying the positional/ rotational/scaling effects of `transform2` to the original transform. If *`transform2`* describes a rotation of 90° around the x axis and this transform describes a translation of 100 units in the y axis, `transform.preMultiply(`*`transform2`*`)` alters this transform so that it describes a rotation followed by a translation.	Nothing
`interpolate (oTransform2, fPercentage)`	Returns a new transform by interpolating from the original transform to *`transform2`* by *`fPercentage`*. The value of `fPercentage` should be between 0 and 100.	A new transform object
`interpolateTo (oTransform2, fPercentage)`	Modifies the existing transform by *`fPercentage`*. The value of *`fPercentage`* should be between 0 and 100.	Nothing
`duplicate()`	Returns a new transform that is a copy of the original transform.	A new transform object
`identity`	Resets the transform to an identity transform: position: –`0,0,0` rotation: `0,00` scale: `1,1,1`	Nothing
`invert()`	Turns the transform into the inverse of its previous position and rotation. If you multiply a vector by a transform, the rotational and positional changes described by the transform are applied to the vector. Inverting the transform and multiplying the vector again restores the vector to its original.	Nothing
`inverse()`	Same as `invert()` except that the original transform is unaffected.	A new transform object

Transform operator

Use the asterisk (*) to multiply two transforms:

Operator	Description	Returns
*transform1 * transform2*	Returns a new transform that is the product of the two original transforms. Useful for combining the effects of two transforms.	A new transform object

Rendering functionality

The Director global `rendererServices` object encapsulates information about the functionality common to all 3D cast members and sprites in a movie. It provides a single place to query for the 3D mesh generators and modifiers available to all cast members.

Renderer services object properties

The global `getRendererServices()` object contains a property list with the following properties. For example, use the syntax `getRendererServices().renderer` to determine the currently active renderer.

Property	Access	Description	Default
`renderer`	Get and set	The rasterizer library all 3D sprites use to draw themselves. This property must be set before any 3D sprite comes into existence. Its default value is determined by the `preferredRenderer` property of the first cast member loaded from file. This is a run-time property that is not saved. Possible values are as follows: `#openGL`: openGL drivers for a hardware accelerator `#directX7_0`: DirectX7_0 drivers for a hardware accelerator `#directX5_2`: DirectX5_2 drivers for a hardware accelerator `#software`: built-in Director software renderer	None
`renderer DeviceList`	Get	A list of available rasterizer libraries. Possible values are as follows: `#openGL`: openGL drivers for a hardware accelerator `#directX7_0`: DirectX7_0 drivers for a hardware accelerator `#directX5_2`: DirectX5_2 drivers for a hardware accelerator `#software`: built-in Director software renderer	None

Property	Access	Description	Default

`modifiers`	Get	A list of modifiers available for 3D cast members. Possible values are as follows:
		`#lod`
		`#toon`
		`#sds`
		`#bonesPlayer`
		`#keyframePlayer`
		`#inker`
		`#collision`
		`#meshDeform`
			None

`primitives`	Get	A list of basic 3D shapes available for all 3D cast members. Possible values are as follows:
		`#box`
		`#sphere`
		`#plane`
		`#particle`
		`#cylinder`
			None

`textureRender Format`	Get and set	A four-digit integer identifying the pixel format used for textures on the 3D hardware accelerator card. Adjust this to improved color fidelity or to fit more textures on the card. You can fit twice as many 16-bit textures as 32-bit textures in the same space. If a movie tries to use more textures than will fit on a card at a single time, Director switches to software rendering. Possible values are as follows: `#rgba8888`: one byte for red, green, blue, and alpha `#rgba8880`: same as above, without alpha opacity `#rgba5650`: 16-bit color with no alpha; 5 bits for red, 6 for green, 5 for blue `#rgba5550`: 16-bit color with no alpha; 5 bits each for red, green, and blue `#rgba5551`: 5 bits each for red, green, and blue; 1 bit for alpha `#rgba4444`: 4 bits each for red, green, blue, and alpha	`#rgba5551`
`depthBuffer Depth`	Get and set	Either 16 or 24, depending on the hardware card. Controls the precision of the hardware depth buffer.	None

Property	Access	Description	Default
`colorBuffer Depth`	Get	Either 16 or 32, depending on the hardware card. Controls the precision of the hardware output buffer.	None
`getHardware Info()`	Returns a property list of the specifics of the hardware card (if any) on the client's machine	A property list with the following entries: `#present`: `TRUE (1)` if the card is present; `FALSE (0)` if the card is absent `#vendor`: the vendor name as a string, with a value of Unknown if the name can't be determined `#model`: the name of the model of the hardware card, as a string `#maxTextureSize[`*`maxWidth, maxHeight`*`]`: maximum height and width of textures. Textures are reduced in size if they exceed these maximums. `#supportedTexturePixelFormat`: texture pixel formats supported by card. `#textureUnits`: number of texture units the card has `#depthBufferRange`: list of bit-depth resolutions available `#colorBufferRange`: list of bit-depth resolutions	

Movie properties

Use these properties to control which renderer the movie uses:

Property	Access	Description
`preferred3d Renderer`	Get and set	The renderer a particular movie prefers. The default value is `#auto`, which allows the movie to pick the best available renderer. This property is not the same as the `currentRenderer` property. The `preferred3dRenderer` property stipulates which renderer the movie prefers, whereas the `currentRenderer` property gives the renderer currently in use. The possible values for the `preferred3dRenderer` property are as follows: `#openGL`: openGL drivers for a hardware accelerator `#directX7_0`: DirectX7_0 drivers for a hardware accelerator `#directX5_2`: DirectX5_2 drivers for a hardware accelerator `#software`: built-in Director software renderer
`active3d Renderer`	Get	The renderer the movie is actually using. Equivalent to the `RendererServices` object `currentRenderer property`. Possible values are as follows: `#openGL`: openGL drivers for a hardware accelerator `#directX7_0`: DirectX7_0 drivers for a hardware accelerator `#directX5_2`: DirectX5_2 drivers for a hardware accelerator `#software`: built-in Director software renderer

Cast member properties

You can control most cast member properties using the Property inspector. For more information, see "Using the Property inspector for 3D" on page 308.

Use the following properties to work with 3D cast members in script:

Property	Access	Description	Default
preload	Get and set	TRUE (1) or FALSE (0) specification of whether the member is preloaded before display and playback or streamed in during playback. This property is only available for linked cast members.	None
animation Enabled	Get and set	TRUE (1) or FALSE (0) specification of whether animation, if any, will play.	TRUE (1)
loop	Get and set	TRUE (1) or FALSE (0) specification of whether animation loops or not.	TRUE (1)
directTo Stage	Get and set	TRUE (1) or FALSE (0) specification of whether rendering occurs directly to the Stage or to the Director offscreen buffer. If TRUE (1), other sprites that intersect with this sprite may flicker. If FALSE (0), rendering layers well, but speed declines.	TRUE (1): rendering occurs directly to the Stage
camera Position	Get and set	Independent *x*,*y*,*z* translation for the default camera with values ranging from *Float_Min* to *Float_Max*.	vector (0.0, 0.0, 250.0)
camera Rotation	Get and set	Independent *x*,*y*,*z* rotation transforms for the default camera with values ranging from *Float_Min* to *Float_Max*.	vector (0.0, 0.0, 0.0)
ambient Color	Get and set	Light applied to entire scene.	rgb(0,0,0)
bgColor	Get and set	Background color in all views.	rgb(0,0,0)
directional Color	Get and set	Color of single "default" directional light.	rgb(255, 255,255)
directional Preset	Get and set	Absolute position of the single "default" directional light: #None #TopLeft #TopCenter #TopRight #MiddleLeft #MiddleCenter #MiddleRight #BottomLeft #BottomCenter #BottomRight	#TopCenter
specularColor	Get and set	Specular color of first shader: the color of reflections from the shader.	rgb(255, 255,255)
reflectivity	Get and set	Reflectivity of first shader, with values from 0.0 to 100.0.	0.0

Property	Access	Description	Default
`diffuseColor`	Get and set	Diffuse color of first shader: the shader's overall color.	`rgb(255, 255,255)`
`textureType`	Get and set	Default texture type for world. Values are as follows: `#none`: no texture `#default`: use original texture from Shader `#member`: use image from specified cast member	`#default`
`texture Member`	Get and set	Name of cast member to use as the source for the default texture when `textureType` is set to `#member`.	No texture
`percent Streamed`	Get	Percentage of file that has been downloaded, with values from `0` to `100`.	None
`bytes Streamed`	Get	Integer number of bytes that have been downloaded, with values from 0 to the size of the file in bytes.	None
`streamSize`	Get	Total size of stream to be downloaded, with values from `0` to the size of the file in bytes.	None
`state`	Get	Current state of streaming. Values are as follows: `0`: unloaded `1`: headerLoading `2`: headerLoaded `3`: mediaLoading `4`: =mediaLoaded `-1`: error After state `3` or `4` has been reached, it's safe to execute script that manipulates the 3D world. Before then, access to particular models may fail because those model definitions may not have been downloaded. Also, the `loadFile()` cast member method fails except at states `0` or `4`.	None

Cast member methods

The following methods let you reset cast member properties to original values they had at the time the cast member was imported into Director:

Method	Description	Returns
`revertToWorldDefaults()`	Resets all cast member properties to the values stored in the original 3D world	Nothing
`resetWorld()`	Resets the 3D cast member to the state it was in when the movie first loaded	Nothing

Sprite properties

You can control the `directToStage` and `bgColor` properties using the Property inspector. For more information, see "Using the Property inspector for 3D" on page 308. Use the following properties to work with 3D sprites in scripts.:

Property	Access	Description	Default
`directTo Stage`	Get and set	`TRUE (1)` or `FALSE (0)` specification of whether rendering occurs directly to the Stage or to the Director offscreen buffer. If `TRUE (1)`, other sprites that intersect with this sprite may flicker. If `FALSE (0)`, rendering layers well but speed declines.	`TRUE (1)`: rendering occurs directly to the Stage
`bgColor`	Get and set	Background color in all views.	`rgb(0,0,0)`
`camera`	Get and set	Determines which camera this sprite is using.	None
`targetFrame Rate`	Get and set	Controls the desired playback speed. If the `useTargetFramerate` property is `TRUE (1)`, then the `lod.bias` property of all model resources is dynamically altered until the target frame rate is met.	`30`
`useTarget FrameRate`	Get and set	If a target frame rate has been set and you want to use it, set this property to `TRUE (1)`.	`TRUE (1)`

Sprite methods

Use these methods to work with 3D sprites:

Method	Description	Returns
`camera(`*`index`*`)`	Accesses a specific camera in the sprite's list of views.	The camera requested
`addCamera(`*`camera, index`*`)`	Adds a camera named *`camera`* at the specified index number. If the index number is greater than the number of cameras in the sprite's camera count, or if there is no index, the camera is added to the end of the list.	An error if a camera of that name can't be found
`deleteCamera (`*`cameraOrIndex`*`)`	If *`cameraOrIndex`* is a camera, the camera by that name is deleted. If *`cameraOrIndex`* is an index number, the camera at that index number is deleted. In either case, the cameras after *`cameraOrIndex`* move forward in the list and the camera count is decreased by 1.	An error if a camera of that name or index number can't be found
`cameraCount()`	Returns the number of cameras in the sprite's `cameraList`.	An integer

CHAPTER 18
Movies in a Window

Macromedia Director MX 2004 can play several movies simultaneously by creating windows in which additional movies can play. A movie in a window (MIAW) is a distinct Director movie that retains all its interactivity.

You can use a MIAW to play another movie in a separate window while the main movie plays on the Stage. In addition, movies in windows and the main movie can communicate and interact with each other. This lets you create a variety of interactive features, such as an interactive portfolio, a control panel for a second movie or digital video, or a status display window.

The following list shows the typical workflow for creating and using a movie in a window:

- Set up the movie's Display Template with default window properties.
- Create and set up the window.
- Assign a movie to the window.
- Open the window.
- Delete the window when the reason for playing the movie no longer applies.

When you create a MIAW, decide how you want it to function. For example, decide how you want to display it, how users should be able to move the window around the screen and dismiss it, and how the window should appear. You can specify the window's size and whether the window is visible, has a frame and title, or is in front of or behind other windows on the screen.

You can create and control movies in windows using behaviors from the Behavior Library or by writing your own scripts.

Macromedia Shockwave does not support MIAWs. Use MIAWs only with movies that you intend to distribute as projectors (see "About distribution formats" on page 451 and "About projectors" on page 460). However, you can use script to have Shockwave content target a URL in a browser window (see "Jumping to a URL" on page 293).

Note: You can also open MIAWs in authoring. Drop a .dir or .dcr file in the Xtras folder and the MIAW will appear in the Xtras menu.

About MIAWs

A MIAW is the combination of a window object and a movie object. In fact, there is no such thing as a MIAW object, but it is the name used to describe the concept of this unique combination of window and movie objects. Most of the properties described in this section are really window properties, but apply to a MIAW. MIAWs and windows are not identical, and it is important to keep that in mind.

Creating or declaring a MIAW

You must first explicitly create or declare the MIAW before you can use it or set its properties.

To explicitly declare a new MIAW in Lingo:

- Use the following script:

```
window().new ("window name")
```

To explicitly declare a new MIAW in JavaScript syntax:

- Use the following script:

```
new window("window name");
```

These statements create a window with the name and title "window name" and assign it the movie "window name" to play in that window.

Note: When you declare a MIAW, it automatically is added to the player’s window list.

Assigning a MIAW a filename and title

Although the simple MIAW declaration line works well when the movie and the window title and name are all the same, you can also explicitly set the filename and title of the MIAW after creating the MIAW.

You can use a movie’s filename as the argument for the `open window` method. This approach assigns that movie to a window and instructs Director to use the filename as the window title.

To assign a filename and title to a MIAW:

- Use the following script to associate the movie `"my_Movie"` with the window:

```
window("window name").filename = "my_Movie"
```

- Use the following script to assign the window's title to be `"window title"`.

```
window("window name").title = "window title"
```

The default behavior of a MIAW is to size itself using the `rect` property of the movie that is being opened in the window.

Opening and closing a MIAW

Use the methods in this section to open and close movies in windows. For more information, see the Scripting Reference topics in the Director Help Panel.

Opening a MIAW

Before you can play a movie in a separate window, you must first open the window explicitly. The window must have already been declared.

To open a MIAW:

- Use the following script:

  ```
  -- Lingo
  window().new("movieName")
  window("movieName").open()
  ```

  -or-

  ```
  window().new("movieName").open()
  ```

  ```
  // JavaScript syntax
  new window("movieName");
  window("movieName").open();
  ```

  -or-

  ```
  new window("movieName").open();
  ```

Unless you explicitly preload the movie through scripting, Director doesn't load the movie into memory until the window is opened, which can cause a noticeable pause. To load the first frame of the movie, use the `preLoadMovie` method.

You can specify other window characteristics before or after you open the window and by using the Display Template at authoring time. For more information, see "Window appearance properties" on page 413.

Closing a MIAW

You can close the window for a MIAW but leave the movie in memory, or you can close the MIAW and remove the movie from memory when it's no longer in use.

- If you leave a MIAW in memory, you get better performance if the window is reopened; however, the movie still consumes memory. You might want to use this option if you expect a MIAW to be reopened after it initially runs, or if other windows or global variables refer to the MIAW.
- If you remove a MIAW from memory, performance slows down if the window is reopened because the movie has to reload; however, it doesn't consume memory until the movie is reloaded. You might want to use this option if you don't expect a MIAW to be reopened after it initially runs or if you want to optimize memory on the computer running the MIAW.

To close a MIAW but keep it in memory:

- Use the `close()` method. After the window is closed, the window becomes invisible, but the movie continues playing. (You can also still access it in the `windowList`.) For example, `window("window name").close()`.

To close a MIAW and remove it from memory:

- Use the `forget()` method. The window is closed, and the movie is removed from memory. Use this method only if no other window or global variables still refer to the MIAW. When you remove a MIAW from memory, it is no longer available in the windowList. For example, `window("window name").forget()`. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

Setting the window size and location for a MIAW

Setting the screen coordinates for a MIAW lets you control the size of the window and where it appears. Setting the coordinates before the movie appears controls the initial position of the window; setting them after the window appears moves the window.

About rect, drawRect, and stageRect

There are three rect values of concern: rect, drawRect and sourceRect. All three properties are available off of the stage or the window:

```
_movie.stage.rect
_movie.stage.drawRect
_movie.stage.sourceRect
window("stage").rect
window("stage").drawRect
window("stage").sourceRect
```

sourceRect: This is a property of the movie's stage, it indicates the "template" rect position at which the stage should appear; therefore, when a window is opened that is to display a particular movie, that movie's sourceRect is used to determine the window's initial position on the user's monitor.

Note: The window's sourceRect property is inherited from the movie that it contains

rect: This is a property of the window playing a particular movie and it indicates the current rect coordinates of the window on the user's monitor; this property gets its initial value (when the window is created) from the window's/movie's sourceRect property

Note: The movie's stage.rect property is inherited from the window that it is playing in.

drawRect - this is a window property that indicates the rect area within the window into which the playing movie is to be drawn; this property is initially set to rect(0,0,w,h) where w and h are the width and height of the window (by default we fill the window with the movie)

Note: The movie's stage.drawRect property is inherited.

To specify the screen coordinates for a MIAW:

Set the `rect` property to the coordinates of the location where you want the window to appear.

- Define the coordinates as a rectangle in this order: left, top, right, and bottom, as shown in the following statement:

  ```
  window("Sample").rect = rect(0, 0, 200, 300)
  ```

  For your convenience, assign the coordinates to a variable and use the variable in your script statements.

For more information, see the Scripting Reference topics in the Director Help Panel.

Controlling the appearance of a MIAW

You can control the appearance of a MIAW by setting different appearance properties. You can do this by using Lingo or JavaScript syntax. Some default appearance properties can also be set by using the Display Template tab found in the Property inspector. You can set the type of window you want—for example, a tool window—and how it will behave and act. For example, a document window can have a titlebar, be resizable, and have a minimize and maximize box. Because the Windows and Macintosh operating systems handle windows differently, some window properties are only available on the Windows platform. For more information about Lingo and JavaScript syntax, see the Scripting Reference topics in the Director Help Panel.

Window appearance properties

Window appearance properties let you customize how your MIAWs look and behave. Depending on what you purpose the MIAW serves, different properties can help you achieve the effects you need. You can set these properties using Lingo or JavaScript syntax. The Display Template, available from the Property Inspector, also allows you to define some default values for a movie's window appearance properties.

General

General properties let you set what type of MIAW you want, its title, whether it is resizable and more. Some properties are for Windows only.

Title lets you set the window's title. It appears in the titlebar of the window.

Type determines the type of window. It can be a tool, document, or dialog window type. If Tool is selected, the window can appear with a short titlebar and a small close box. Tool windows do not receive activate or deactivate events, because tool windows are always active. They appear on top of document windows but layer with each other. If Document is selected, the window can appear with a standard titlebar, a close box, and a minimize and maximize box. If Dialog is selected, the window has a standard titlebar and a close box. Dialog windows are always on top and are modal.

Location determines the distance of the window from the left side and top of the screen. These values specify pixels, and they apply only if the Stage is smaller than the current monitor's screen size.

Centered places the window in the center of the monitor. This option is useful if users might play a movie that was created for a 13-inch screen on a larger screen or if you are creating a movie on a large screen that will be seen on smaller screens.

Resizable determines whether the MIAW is resizable or not.

Docking determines whether a MIAW is a dockable window when opened for authoring. For more information, see "About dockable MIAWs" on page 414.

Titlebar options

Title bar properties determine what icons appear in the title bar of a window. These are accessed as part of the `titleBarOptions` list.

Maximize Box determines whether a maximize box appears in the title bar of a window.

Close Box determines whether a close box appears in the window.

Minimize Box determines if a minimize box appears in the title bar. For example:

window("window_name").titlebarOptions.maximizebox = true

Visible determines whether the title bar of a window is visible. When you set this property to false, the other options (Maximize Box, Close Box, and so forth) remain untouched. The next time the window's title bar is visible, all other title bar properties are maintained. In other words, it lets you override or keep changes made to these properties.

Icon lets you determine the icon that appears in the title bar of a MIAW. You must select a cast member to act as the icon.

Side Titlebar (Macintosh only property) determines whether or not the title bar appears on the top or the left side of the window.

Appearance options

Appearance properties let you determine how a window looks. For example, if you are designing a window that appears in a Macintosh application, you can set the appearance to mimic a typical Macintosh window. These properties are accessed as part of the `appearanceOptions` list.

Drop Shadow (Mac only) lets you determine if there is a shadow around the window. Most Mac windows have it turned on.

Metal (Mac only) lets you specify whether a window will have a metal or ice appearance.

Mask lets you assign a bitmap cast member as a mask for the window. For example:

window("window_name").appearanceOptions.mask = member("my mask")

Drag Mask lets you use a bitmap cast member as a mask to determine which areas of the window a user can click on to move the window. You can use this to create your own custom nonrectangular title bars or to make the entire window draggable if you wish.

Border lets you set the type of border a window should use. None indicates a borderless window, and line specifies a 1-pixel black border. However, to create a borderless window, you will also need to set both "resizable" and "titlebarOptions.visible" to false, as follows:

```
window("win").resizable = false
window("win").titlebarOptions.visible = false
window("win").appearanceOptions.border = false
```

Live Resize (Macintosh only property) determines whether or not the window contents and the size of the window will update while the user is dragging to resize the window.

About dockable MIAWs

Tool and document MIAWs can now be dockable in authoring. To make a MIAW dockable, set the `dockingEnabled` property to `TRUE`. Setting the type determines what the MIAW can dock with. For example, dockable tool MIAWs can dock to each other, to other tool windows such as the Property inspector, and can be docked to the side channels. They can also be grouped with other tool windows. Dockable document MIAWs can dock to each other and to other document windows such as the Stage, Score, and Cast windows, and all Media Editors.

- To dock a MIAW, click and drag it to a compatible window by the header bar.
- To group a MIAW, click on the Panel menu on the right side of the header bar and select a Panel group.

Window appearance methods

There are several new window appearance methods in Director MX 2004 that apply to MIAWs. For more information about these methods, see the Scripting Reference topics in the Director Help Panel.

The following are three of these methods:

- `minimize()` lets you minimize your windows. You can call this function when they make custom title bars. Here is an example:
  ```
  window ("miaw").minimize()
  ```
- `maximize()` lets you maximize windows. You can call this function when you make custom title bars. Here is an example:
  ```
  window ("miaw").maximize()
  ```
- `restore()` lets you restore a window after it has been maximized. You can call this function when you set custom title bars. Here is an example:
  ```
  window ("miaw").restore()
  ```
- `notifyUser()` allows developers to notify the user that the application needs the user's attention. On Windows, the window's task bar entry will blink. On Macintosh, the projector's icon will bounce in the dock. The method arguments are flash count and rate. Setting the flash count to -1 causes the flash or bounce to continue indefinitely until the user responds. Here is an example:
  ```
  window("miaw").notifyUser(-1, 10000)
  ```
- `displaySystemTrayMessage()` (Windows only) This method allows users to pop up a message over their system tray icon. The message will appear in a balloon tooltip. This is enabled only if the Window's `systemTrayTooltip` property is set to true. The method arguments are Message Title, Message Content, and display duration. Here is an example:
  ```
  window("miaw").displaySystemTrayMessage("Message Title", "Message....",
    1000);
  ```

Setting default MIAW properties using the Display Template

A movie can define the default properties of its window by using the Display Template. When a MIAW is created, it will use its movie's Display Template properties to determine the initial state and appearance of its window. You can set the Display Template properties through the Property Inspector when authoring a movie. For more information about the different properties, see "Controlling the appearance of a MIAW" on page 413.

To set the Display Template:

1. With a movie open in the Stage, select Window > Property inspector.
2. Click the Display Template tab. A list of MIAW properties appears.
3. Set the General properties:
4. Set the Titlebar options.
5. Set the Appearance options.

Controlling MIAW layering

To control whether a movie appears in front of or behind other windows:

- Use the `moveToFront()` and `moveToBack()` methods. For more information about these methods, see the Scripting Reference topics in the Director Help Panel.

This code moves a MIAW to the front:

```
window("miaw").moveToFront()
```

This code moves a MIAW to the back:

```
window("miaw").moveToBack()
```

MIAW events

Lingo or JavaScript syntax provides event handlers for typical events that can occur while a MIAW is playing, such as the movement of a window by the user. Such a handler is a good place for instructions that you want to run in response to an event that involves a window.

For example, to cause a sound to play whenever the user closes a MIAW, use the `queue()` and `play()` methods in an `on closeWindow` handler in a movie script within the movie that plays in the window. The `on closeWindow` handler will run whenever the MIAW that contains the handler closes.

Other handlers include:

- `on moveWindow` (called when a user moves the movie's window)
- `on resizeWindow` (called when a user resizes the movie's window)
- `on zoomWindow` (called when the minimize or maximize box of a movie's window is pressed)
- `on openWindow` (called when the movie's window is opened)
- `on closeWindow` (called when the movie's window is closed)
- `on activateWindow` (when a movie's window is activated)
- `on deactivateWindow` (when a movie's window is deactivated)
- `on revealWindow` (when a dockable MIAW in authoring is revealed)
- `on collapseWindow` (when a dockable MIAW in authoring is collapsed)
- `on activateApplication` (when the entire application has been activated)
- `on deactivate Application` (when the entire application has been deactivated)
- `on trayIconMouseDown` (Windows only) This event handler is triggered when a user mouses down on the Window's system tray icon. This event is only meaningful if the `systemTrayIcon` property is set to true.
- `on trayIconDoubleClick` (Windows only) This event handler is triggered when a user double-clicks on the Window's system tray icon. This event is only meaningful if the `systemTrayIcon` property is set to true.
- `on trayIconRightMouseDown` (Windows only) This event handler is triggered when a user right mouses down on the Window's system tray icon. This event is only meaningful if the `systemTrayIcon` property is set to true. For example:

  ```
  on trayIconRightMouseDown
    alert ("right mouse down")
  end
  ```

For more information, see the Scripting Reference topics in the Director Help Panel.

Listing the current MIAWs

The `windowList` property displays a list of all known MIAWs in the main movie. For example, the following statement displays a list of current MIAW names in the Message window:

```
put_player.windowList
```

For more information about this property, see the Scripting Reference topics in the Director Help Panel.

Controlling interaction between MIAWs

MIAWs can interact with other MIAWs by accessing a window's movie property. With a window's movie, a user can access the movie's handlers, variables, members, and so on.

Here are a couple of examples:

```
window("other MIAW").movie.member(1).name = "changed name"
```

or

```
window("other MIAW").movie.someHandler()
```

Global variables can be declared in the main movie (the Stage) or in a MIAW. No matter where they are declared, they are available to the main movie and to all MIAWs. For more information about global variables, the Scripting Reference topics in the Director Help Panel.

A MIAW can also interact with the main movie via `window("stage")`.

For example:

```
window("stage").movie.member(1).name = "changed name"
```

or

```
window("stage").movie.someHandler( )
```

CHAPTER 19
Using the XML Parser Xtra

The XML Parser Xtra lets Macromedia Director MX 2004 movies read, parse, and use the contents of Extensible Markup Language (XML) documents. Using the XML Parser Xtra requires that you understand the structure and content of the documents you are parsing. You can then access the XML document's contents through Lingo or JavaScript syntax or convert the contents to a script list that is meaningful to you and your movie. After your movie has read an XML document, it can perform actions that you define based on the contents of the document.

For example, an XML document might describe the structure of a molecule for an educational chemistry application. After the movie has parsed the XML that describes the molecule, it can use that information to draw an accurate visual representation of the molecule on the screen. In this case, the movie must be programmed to know ahead of time that the XML document describes a molecule and not a grocery list, in order to make logical use of it.

About XML

XML is similar to HTML in that it uses markup tags to define content. However, HTML has predefined tags that you can use to format any data. Any application that reads HTML must understand the meaning of tags such as `TITLE`, `P`, and `BODY`. HTML tags also describe how information appears on the screen. XML, on the other hand, consists of a set of rules that let you define custom tags and the type of data they can contain, and it has no visual component. With XML, there is no predefined way to display any given type of data such as molecular structures or grocery lists. An XML document is merely a container for the data.The Director developer, by knowing what kind of data the XML document contains, can make intelligent decisions about what the Director movie should do with the information.

One key advantage of XML over a regular text document is that XML is not order-dependent. If an application refers to the third item in a line of data, inserting new data or making subsequent changes to the way the data is produced could cause the application to fail. With XML, you can refer to the individual data components by name. If you insert a new chunk of data before the one in use, the name is still valid. Existing code continues to work, and the newly inserted data is ignored.

There are many sources of information for understanding, creating, and editing XML on the Internet. The following websites offer useful information about XML:

- www.xml.com
- www.ucc.ie/xml/
- www.w3.org/TR/REC-xml
- www.w3.org/DOM/

Using XML parser objects

The XML Parser Xtra lets Director developers access the nodes of an XML document. A node can be a tag (similar to an HTML tag, also called an element), character data (text that does not appear inside the angle brackets of a tag), or a processing instruction (a special type of tag that passes data to the parsing application for special processing). You can extract information from the XML document by looking at its nodes with Lingo or JavaScript syntax. This access to XML data lets users incorporate XML documents into their movies and selectively extract data from the documents.

An XML document is well formed if it has a coherent nesting structure and no misplaced angle brackets. Some XML documents can be associated with a document type declaration (DTD) file that describes specific rules for the XML tags the document uses. The XML parser of the Xtra checks that the XML follows the general rules of creating and using XML tags to ensure that the document is well formed. However, the Xtra does not check the DTD file to ensure that the tags follow the specific rules for XML document tags, which is described by the DTD. For this reason, the Xtra is called *nonvalidating*. The creator of the original XML document must follow the rules described in the DTD. Your Director movie should also include script that checks for errors in the use of the XML document's tags.

To use the XML Parser Xtra, create a parser object by using Lingo or JavaScript syntax to assign a new instance of the Xtra to a variable. This variable now contains the parser object. Use a global variable if you need to access the XML data from anywhere in the Director movie.

```
global gParserObject
gParserObject = new(xtra "xmlparser")
```

The next step is to parse the XML data using the `parseString()` method. The data can come from the text of a cast member or a string variable. To parse XML from a URL, use `parseURL()` instead. `ParseString()` and `parseURL()` return either `VOID`, which indicates that the method is successful, or an error code that indicates a problem with the XML data.

The following script statement sets the variable `errCode` to the return value of the `parseString()` method:

```
errCode = gParserObject.parseString(member("XMLtext").text)
```

After the XML Parser Xtra parses the data, the parser object contains all the data from the XML document.

The XML data can be considered a tree structure because most documents have tags nested within other tags, with each tag being like a branch of the tree.

The following example shows a short XML document:

```
<?xml version="1.0"?>
<e1><tagName attr1="val1" attr2="val2"/><e2>element 2</e2><e3>element 3</e3>
  </e1>
```

The following example is the same XML with its tree structure shown more clearly:

```
<?xml version="1.0"?>
  <e1>
    <tagName attr1="val1" attr2="val2"/>
    <e2>element 2</e2>
    <e3>element 3</e3>
  </e1>
```

There are two ways to access a parsed XML document. You can use the `makeList()` method to convert the document into a nested property list and use the list-access methods of Lingo or JavaScript syntax, or you can use the special script methods of XML Parser Xtra to access the parsed data directly.

The following script statement uses the `makeList()` method to convert the parsed data to a property list:

```
theList = gParserObject.makeList()
```

If you choose to make a property list with the `makeList()` method, the result is a nested property list, reflecting the tree structure of the XML.

Each element in the document is represented by its own property list, with another property list for each child element that it contains. The name of the element is the property name, and the content of the element is the property value. Attributes of an element are stored in a child list with the name !ATTRIBUTES. The property list of attributes contains the name of each attribute and its value. Character data has the property name !CHARDATA, and the value is the string representation of the character data. A processing instruction is a property with the name !PROCINST; its value is another two-element property list. The first property of this sublist is NAME, and the value is the string that represents the name of the processing instruction. The second property of the sublist has the name TEXT and contains the rest of the text in the processing instruction.

The property list resulting from the previous XML example would look like the following code:

```
["ROOT OF XML DOCUMENT": ["!ATTRIBUTES": [:], "e1": ["!ATTRIBUTES": [:],
  "tagName": ["!ATTRIBUTES": ["attr1": "val1", "attr2": "val2"]], "e2":
  ["!ATTRIBUTES": [:], "!CHARDATA": "element 2"], "e3": ["!ATTRIBUTES": [:],
  "!CHARDATA": "element 3"]]]]
```

The following example is the same property list with its nested structure shown more clearly:

```
["ROOT OF XML DOCUMENT": ["!ATTRIBUTES": [:],
  "e1": ["!ATTRIBUTES": [:],
    "tagName": ["!ATTRIBUTES": ["attr1": "val1", "attr2": "val2"]],
    "e2": ["!ATTRIBUTES": [:], "!CHARDATA": "element 2"],
    "e3": ["!ATTRIBUTES": [:], "!CHARDATA": "element 3"]
  ]
]]
```

Together, the Lingo or JavaScript syntax statements that create a property list from a string of XML data would look like the following examples:

```
global gParserObject
gParserObject = new(xtra "xmlparser")
```

```
errCode = gParserObject.parseString(member("XMLtext").text)
theList = gParserObject.makeList()
```

After this code has been executed, the variable `gParserObject` contains the parsed node structure of the XML document, and the variable `theList` is a property list that contains all the information in the document broken into property name and value pairs. All the regular script methods for sorting and accessing lists can work normally with `theList`.

Using XML document nodes

The XML document can contain different types of nodes. Each node can contain different kinds of data, depending on the node type. You should check the node type before accessing its data so you know what type of data to expect. Nodes are read-only, so you can retrieve the type, but you cannot set it.

You use Lingo or JavaScript syntax to access the nodes of an XML document. The following table shows the script terms that refer to nodes and their properties:

Node Script	Return value if an element	Return value if text	Return value if Processing Instruction
type	`#element`	`#text`	`#procInst`
`name`	String representing the name of the element	VOID	String representing the name of the processing instruction
`child[N]` (N is an integer)	The Nth child node of the node; VOID is returned if no Nth child exists or there is a script error	VOID	VOID
`attributeName[N]` (N is an integer)	String representing the name of the Nth attribute; VOID is returned if no Nth attribute exists or there is a script error	VOID	VOID
`attributeValue[N]` (N is an integer)	String representing the value of the Nth attribute; VOID is returned if no Nth attribute exists or there is a script error	VOID	VOID
`attributeValue[N]` (N is a string)	String representing the value of the attribute with the name N; VOID is returned if the node does not have an attribute named N or there is a script error	VOID	VOID
`text`	VOID	String representing the character data contained in this node	String representing the data section of the processing instruction

Note: The subfield `count` exists for any field that is accessible with bracket access. You can specify `whichNode.child.count` to find how many children are in the specified node.

Using this XML document as a starting point, the following examples demonstrate how to use these script terms to access the data within various node levels of the XML structure.

The XML looks like the following example:

```
<?xml version="1.0"?>
  <e1>
    <tagName attr1="val1" attr2="val2"/>
    <e2>element 2</e2>
    <e3>element 3</e3>
    Here is some text
  </e1>
```

The following script returns the name of the first XML tag:

```
put gParserObject.child[1].name
-- "e1"
```

The `gParserObject` variable contains the parsed XML. When used in the preceding script, it refers to the root node of the XML document. The script term `child[1]` refers to the first level of nested tag, which is the `e1` tag.

To find out what kind of node the first tag is, use the `type` method, as shown in the following example:

```
put gParserObject.child[1].type
-- #element
```

To refer to nodes that are nested more than one level deep, use more than one level of child reference. The following script returns the name of the first tag that is nested within the `e1` tag:

```
put gParserObject.child[1].child[1].name
-- "tagName"
```

The following script returns the name of the second tag that is nested within the `e1` tag:

```
put gParserObject.child[1].child[2].name
-- "e2"
```

To refer to text data that occurs within a particular tag, use the `text` property. The text is a child node of the tag that contains it, so you need an additional level of child reference. This script returns the following string, which appears inside the `e2` tag from the previous XML example:

```
put gParserObject.child[1].child[2].child[1].text
-- "element 2"
```

In this example, the `gParserObject` variable refers to the root node of the XML. The `child[1]` refers to the `e1` tag, which occupies the first level down in the XML's nested structure. The `child[2]` refers to the second tag within the `e1` tag, which is the `e2` tag. The last `child[1]` refers to the text within the `e2` tag, which is `element 2`. Finally, the `text` property is specified, so script returns the text of the node rather than any other property of the node.

The fourth child of the `e1` tag is a line of text that reads `here is some text`. This text is a child the same as the XML tags that precede it. You can get the type of this child the same way you get other children.

The following script returns the type of the fourth child of the `e1` tag:

```
put gParserObject.child[1].child[4].type
-- #text
```

This script returns the text of the fourth child of the `e1` tag, as shown in the following example:

```
put gParserObject.child[1].child[4].text
-- "
     here is some text
   "
```

The text element includes the white space for Return, Space, and Tab characters as well as the string `"here is some text"`.

You can use the script `count` method to determine the number of children that exist at a particular level of the XML structure. The following script returns the number of children at the 2nd level in the previous XML example:

```
put gparser.child[1].child.count
-- 4
```

Accessing attributes

Some XML tags contain attributes with values. Use the `attributeName` and `attributeValue` properties to access the attributes of tags that have values. In the previous XML example, the first tag nested inside the `e1` tag is called `tagName` and has two attributes called `attr1` and `attr2`.

The following script uses the `attributeName` property to return the name of the first attribute of the tag called `tagName`, which is the first child of the `e1` tag:

```
put gParserObject.child[1].child[1].attributeName[1]
-- "attr1"
```

The following script uses the `attributeValue` property with an integer to return the value of the first attribute of the `tagName` tag:

```
put gParserObject.child[1].child[1].attributeValue[1]
-- "val1"
```

The following script uses the `attributeValue` property with a string to return the value of the `attr1` attribute:

```
put gParserObject.child[1].child[1].attributeValue["attr1"]
-- "val1"
```

The following script uses the `count` method with the `attributeName` property to return the number of attributes in the first child of the `e1` tag:

```
put gParserObject.child[1].child[1].attributeName.count
-- 2
```

Parser objects and XML nodes

As described in earlier sections, the parser object in the `gParserObject` variable stores the root of the parsed tree of the XML document. An XML node is a node within the tree. The root node is like an XML node because almost all the operations on XML nodes can be applied to the root node.

In the previous XML example, the root of the tree is an XML element node named "ROOT OF XML DOCUMENT" that has no attributes and one child (the `e1` tag). You can get the same information about the root node as for any of the child nodes.

The following script returns the root node's type, name, and number of children:

```
put gParser.type, gParser.name, gParser.count(#child)
-- #element "ROOT OF XML DOCUMENT" 1
```

The main difference between the root node and its child nodes is that there are several script methods that apply to the entire XML document and operate on the root node only. These methods include `doneParsing()`, `getError()`, `ignoreWhiteSpace()`, `makeList()`, `parseString()`, and `parseURL()`.

Treating white space

The default behavior of the XML Parser Xtra is to ignore character data between XML tags when all the characters are white space. This type of white space is usually due to Return characters and superfluous space characters, but sometimes it can have meaning to the XML document.

You can use the `ignoreWhiteSpace()` method to change the way the Xtra treats white space. By setting the `ignoreWhiteSpace()` to `FALSE` instead of its default value of `TRUE`, you can tell the Xtra to treat instances of white space as literal data nodes. This way, white space between elements is treated as actual data.

The following script statements leave `ignoreWhiteSpace()` set to the default `TRUE` value, and parse the given XML into a list. The `sample` element has no children in the list.

```
XMLtext = "<sample> </sample>"
parserObj.parseString(XMLtext)
theList = parserObj.makelist()
put theList
-- ["ROOT OF XML DOCUMENT": ["!ATTRIBUTES": [:], "sample": ["!ATTRIBUTES":
  [:]]]]
```

The following script statements set `ignoreWhiteSpace()` to `FALSE`, and parse the given XML into a list. The `sample` element now has a child that contains one space character.

```
XMLtext = "<sample> </sample>"
parserObj.ignoreWhiteSpace(FALSE)
parserObj.parseString(XMLtext)
theList = parserObj.makelist()
put theList
-- ["ROOT OF XML DOCUMENT": ["!ATTRIBUTES": [:], "sample": ["!ATTRIBUTES":
  [:], "!CHARDATA": " "]]]
```

If there are non-white space characters in a !CHARDATA node, all the characters of the node, including leading and trailing white space characters, are retained.

XML and character sets

When you use XML, remember that different computer systems use different binary encoding to represent text characters.

The XML Parser Xtra adheres strictly to the XML specification, which states that XML documents are, by default, encoded using the UTF-8 character set. If the document is not encoded in UTF-8, it must include a declaration of its character set in the first line of the document.

The following XML declares the IOS-8859-1 character set, also known as Latin1:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
```

This requirement applies to documents parsed by `parseString()` as well as files that are parsed by `parseURL()`.

The XML Parser Xtra supports the following character sets:

- **ISO-8859-1** Also called Latin 1. This is the most common Western encoding used on the web. It matches the default character set used on Windows in most Western countries. It does not match the character set used in most Western versions of the Mac OS (MacRoman) and does not match character sets commonly used in most non-Western countries. The first 127 characters (binary codes 1-127) are the same in most countries.
- **UTF-8** An 8-bit encoding of the Unicode character set. This is the XML default character set.
- **US-ASCII** Supports only 7-bit characters.
- **EUC-JP** The EUC character set, used widely in Japan.
- **SHIFT_JIS** Also used widely in Japan. Shift-JIS is the character set used by default in Japanese versions of Windows and the Mac OS.
- **UTF-16** A 16-bit encoding of the Unicode character set.

For many developers, Latin 1 is the most convenient encoding to use.

CHAPTER 20
Making Director Movies Accessible

As a Macromedia Director MX 2004 author, you might want to provide ways for computer users with disabilities to experience the movies that you create. Director includes accessibility features that let you do this. By using these features, you can make existing and new movies accessible to users who have hearing, visual, or mobility impairment.

With scripts and behaviors, you can add text-to-speech capability to your movies. This lets visually impaired users hear the text in your movie read aloud by the computer. You can provide captioning to help users with hearing impairment experience the audio portions of your movies. Finally, you can enable your movies to be navigated with the keyboard instead of a mouse, which benefits users with certain kinds of mobility impairment.

If you work on projects sponsored by the U.S. government, you might be required to follow government guidelines for providing accessibility to disabled users.

You add accessibility to your Director movies by using the included accessibility behaviors or special script methods. The text-to-speech behaviors and scripting require the Speech Xtra. If you use text-to-speech in your movie, you must add the Speech Xtra to your movie's Xtra extensions list. This is discussed in detail in this chapter.

About government requirements

The U.S. government has stated that multimedia created for the purpose of fulfilling a government contract must be made accessible to computer users with disabilities. The government requirements include text-to-speech for the visually impaired, captioning for the hearing impaired, and keyboard navigation for the mobility impaired.

The full text of the government requirements is available at www.section508.gov/.

Making Director movies accessible

You can make your Director movies accessible by adding behaviors or writing custom scripts. Director includes several behaviors that let you easily add text-to-speech, captioning, and keyboard navigation to your movies with simple drag-and-drop procedures. If you want to have more control over how you implement accessibility in your movies, you can write custom scripts that use the text-to-speech methods. For more information about text-to-speech script, see "Accessibility scripting with Lingo or JavaScript syntax" on page 434.

Using the Speech Xtra

The Speech Xtra adds special methods to scripting that enable the Director text-to-speech capability. The text-to-speech behaviors require this Xtra extension because they use these script methods. If you write custom text-to-speech scripts, you must include the Speech Xtra in your movie's Xtra extensions list. Keyboard navigation does not require the Speech Xtra.

The Speech Xtra supports Microsoft SAPI 4 and 5 on Windows. The Speech Xtra supports all versions of text-to-speech on the Macintosh.

Because the Speech Xtra has no specific cast member type associated with it, it is not added automatically to a movie's Xtra extensions list when a certain type of cast member is added to the movie. Therefore, when you use text-to-speech script or behaviors in a projector, you must remember to add the Speech Xtra to the Xtra extensions list manually. The Speech Xtra is available automatically in the Director application and in Macromedia Shockwave.

To add the Speech Xtra to a movie's Xtra extensions list:

1. Select Modify > Movie > Xtras.
2. Click Add.
3. In the alphabetical list of Xtra extensions, scroll to Speech, and select the Speech Xtra. (In Windows, the file name is Speech.x32.)
4. Click OK to close the Add Xtras dialog box.
5. Click OK to close the Movie Xtras dialog box.
6. Save your movie.

Testing the Speech Xtra

To verify that your computer is configured correctly to let the Speech Xtra work, you can perform a simple test.

- To test the Speech Xtra, type the following script into the Message window:

  ```
  put voiceInitialize()
  ```

  If the result is 1, then the Speech Xtra is working. If the result is 0, then you might not have text-to-speech software installed on your computer. For more information about text-to-speech software, see "System requirements" on page 436.

Using the Accessibility behavior library

You can make movies accessible in three ways. You can add keyboard navigation, text-to-speech, and captioning. The Director Library palette includes an Accessibility section that contains behaviors for enabling each capability.

For keyboard navigation, you use the Accessibility Target, Accessibility Item or Accessibility Text Edit Item, Accessibility Keyboard Controller, and Accessibility Group Order behaviors.

For text-to-speech, you use the keyboard navigation behaviors and then add either the Accessibility Speak or Accessibility Speak Member Text behavior. You can also use the Accessibility Speak Enable/Disable behavior to let users turn the text-to-speech feature on and off.

For captioning, you use the Keyboard navigation behaviors and Accessibility Speak behaviors and then add the Accessibility Captioning and Accessibility Synch Caption behaviors.

Enabling keyboard navigation

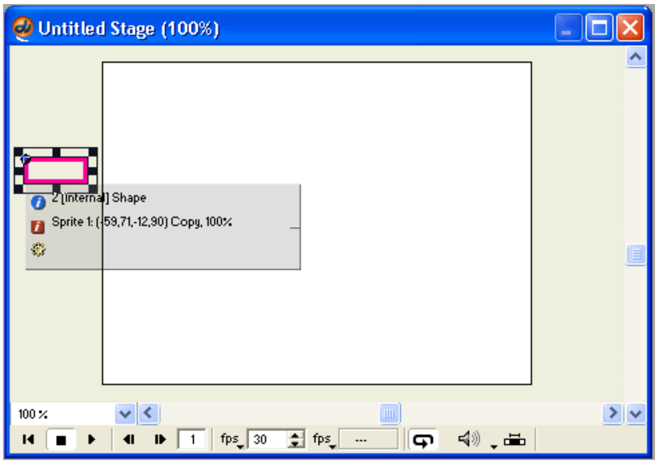
With the Accessibility Behavior library, you can easily make sprites on the Stage navigable with the keyboard. This lets mobility-impaired users select sprites and simulate mouse clicks without using a mouse. For example, if you have four button sprites on the Stage, you can apply accessibility behaviors that let them be selected by using the Tab key. As each sprite is selected, it is highlighted with a colored rectangle, called a focus ring, around its bounding rectangle. After a sprite is selected, the user can press Enter (Windows) or Return (Macintosh) to initiate the same action that a mouse click would initiate on the sprite.

The accessibility behaviors work with each other. Most of them cannot be used alone and must be used with other accessibility behaviors. To enable keyboard navigation, you must use the Accessibility Target, Accessibility Item, Accessibility Text Edit Item, Accessibility Group Order, and Accessibility Keyboard Controller behaviors together.

To apply the Accessibility Target behavior:

1 Create a shape sprite on the Stage using the Rectangle tool in the Tools panel. The accessibility behaviors use this sprite to create a focus ring around other sprites you define as navigable with the keyboard. The focus ring shows which sprite is the current selection during keyboard navigation.

2 Move the shape sprite off the Stage into the area immediately next to the Stage. You can expand the Stage window to display more of this area if necessary. The other accessibility behaviors move the sprite onto the Stage when keyboard navigation is used.



A shape sprite off the visible part of the Stage.

3 Open the Library palette by selecting Window > Library palette.

4 Select Accessibility from the Library List menu in the upper-left corner of the Library palette.

5 Drag the Accessibility Target behavior on to the shape sprite that is located next to the visible part of the Stage.

6 In the dialog box that appears, enter a name for the behavior group.

Because the accessibility behaviors work together and are dependent on one another, a group name associates all the accessibility behaviors used in a given scene of your movie with one another. When you apply behaviors to a different scene of your movie, you might want to use a different group name. For example, you might use Accessibility_Scene_1. In addition, there should only be one instance of the Accessibility Target behavior in any given scene of your movie.

7 Select an initial state for text-to-speech, either enabled or disabled. If you select disabled, you must use the Accessibility Speak Enable/Disable behavior to let users turn on text-to-speech later. For more information, see "Enabling text-to-speech" on page 431.

8 Click OK.

The next step in the process is to attach the Accessibility Keyboard Controller behavior to a sprite. This behavior intercepts keystrokes from the keyboard and enables those keyboard events to be used for navigation of the sprites on the Stage. You attach the Accessibility Keyboard Controller behavior to an editable text sprite that you place outside the visible part of the Stage.

To apply the Accessibility Keyboard Controller behavior to an off-Stage editable text sprite:

1 Place an editable text sprite on the Stage. Do not use a field sprite.

2 Move the text sprite off the Stage into the area immediately below the Stage. You can expand the Stage window to display more of this area if necessary.

3 Drag the Accessibility Keyboard Controller behavior from the Library palette to the sprite.

4 In the dialog box that appears, select the accessibility group name for the scene, such as Accessibility_Scene_1.

5 Click OK.

6 Select the editable text sprite if it is not still selected.

7 Using the arrow keys, move the sprite off the edge of the Stage until it is no longer on the visible part of the Stage.

Note: If the editable text sprite receives a Return character, its bounding rectangle can expand vertically. To prevent this from happening and unexpectedly causing the sprite to become visible on the Stage, set the sprite's Framing property to Fixed in the Property inspector's Sprite tab.

Now that the Accessibility Target behavior is attached to the shape sprite and the Accessibility Keyboard Controller behavior is attached to a text sprite, you are ready to apply the Accessibility Item or Accessibility Text Edit Item behaviors. To enable navigation to a sprite with the Tab key, attach the Accessibility Item behavior to the sprite. To enable navigation to an editable text sprite with the Tab key, attach the Accessibility Text Edit Item behavior to the sprite.

To apply the Accessibility Item or Accessibility Text Edit Item behaviors:

1 Drag the Accessibility Item or Accessibility Text Edit Item behavior from the Library palette to the designated sprite.

2 In the dialog box that appears, select the same group name that you used when applying the Accessibility Target behavior—for example, Accessibility_Scene_1.

3 You can specify a script string to be executed when the user presses Enter (Windows) or Return (Macintosh) while the sprite is selected. You can use any valid script string, such as a `go to frame 20` method or a call to a separate handler that you have written, such as `startAnimation`.

4 Click OK.

5 Repeat this process for each sprite in the scene that you want to be navigable.

Note: The Accessibility Item and Accessibility Text Edit Item behaviors cannot be applied to sprites that have the Accessibility Target or Accessibility Keyboard Controller behaviors attached. If you are testing your movie while applying Accessibility Item or Accessibility Text Edit Item behaviors, you might need to rewind your movie to restore the shape sprite with the Accessibility Target behavior to its original location.

After all the sprites have had an Accessibility Item or Accessibility Text Edit Item behavior attached to them, you can add the Accessibility Group Order behavior to them. This behavior lets you specify the order in which each sprite is selected on the Stage when the user presses the Tab key. Apply this behavior to each sprite you want to be navigable with the Tab key.

To apply the Accessibility Group Order behavior:

1 Drag the Accessibility Group Order behavior from the Library palette to one of the sprites that already has an Accessibility Item or Accessibility Text Edit Item behavior attached.

2 In the dialog box that appears, select the group name that is used by the other accessibility behaviors in the scene, such as Accessibility_Scene_1.

3 Enter the tab order for the sprite. This is the order in which the sprites are selected when the user presses the Tab key. Be sure to number each sprite consecutively and to use each number only once. When the movie begins playing, the focus ring automatically goes to the sprite whose group order is 1.

4 Click OK.

5 Repeat this process for each sprite in the scene that is navigable with the Tab key.

All the sprites can now be navigated to and activated with the keyboard. You can repeat this process for each scene in your movie.

Enabling text-to-speech

To help visually impaired users experience your Director movies, you can add the ability for your text cast members to be spoken aloud by the computer. You can also specify text to be spoken when sprites are selected. Because the text-to-speech feature in Director assumes that users are visually impaired, the feature is designed to be used with the keyboard navigation feature. Each text-to-speech behavior requires that the Accessibility Target, Accessibility Item or Accessibility Text Edit Item, Accessibility Group Order, and Accessibility Keyboard Controller also be applied to the appropriate sprites.

The following list describes the three text-to-speech behaviors:

- The Accessibility Speak behavior lets you specify a text string to be spoken when the user navigates to a sprite with the Tab key.
- The Accessibility Speak Member Text behavior lets you specify a text cast member to be spoken when a user navigates to a sprite with the Tab key.
- The Accessibility Speak Enable/Disable behavior lets you specify a user event that can turn the text-to-speech feature on or off.

To apply the Accessibility Speak behavior:

1 Begin by applying the keyboard navigation behaviors to sprites in your scene. For more information, see "Enabling keyboard navigation" on page 429.

2 Drag the Accessibility Speak behavior from the Library palette to the sprite you want to trigger the spoken text when the user navigates to it with the Tab key. This sprite must already be attached to an Accessibility Item or Accessibility Text Edit Item and an Accessibility Group Order behavior.

3 In the dialog box that appears, select the behavior group name for the scene, such as Accessibility_Scene_1. This group name enables the behaviors in a scene of your movie to communicate with each other so they can operate properly.

4 Enter the text to be spoken when the sprite is selected with the Tab key.

5 Click OK.

If you wish to have a large amount of text spoken, you can specify an entire text cast member to be spoken by using the Accessibility Speak Member Text behavior.

To apply the Accessibility Speak Member Text behavior:

1 Apply the keyboard navigation behaviors to sprites in your scene. For more information, see "Enabling keyboard navigation" on page 429.

  Drag the Accessibility Speak Member Text behavior from the Library palette to the sprite that you designate to trigger the spoken text when the user navigates to it with the Tab key. This sprite must already be attached to an Accessibility Item or Accessibility Text Edit Item and an Accessibility Group Order behavior.

2 In the dialog box that appears, select the behavior group name for the scene, such as Accessibility_Scene_1. This group name enables the behaviors in a scene of your movie to communicate with each other so they can operate properly.

3 Enter the name of the text cast member to be spoken when the sprite is selected with the Tab key.

4 Click OK.

You can let users toggle the text-to-speech feature on and off by using the Accessibility Speak Enable/Disable behavior. You add this behavior to a sprite that already has the keyboard navigation behaviors attached to it.

To apply the Accessibility Speak Enable/Disable behavior:

1 Apply the keyboard navigation behaviors to sprites in your scene. For more information, see "Enabling keyboard navigation" on page 429.

2 Drag the Accessibility Speak Enable/Disable behavior from Library palette to the sprite you want to use to toggle the text-to-speech behaviors.

3 In the dialog box that appears, enter the behavior group name for the scene so that this behavior is associated with the other behaviors in the scene.

4 Select an event to toggle the on/off state of the text-to-speech behaviors. This can be a mouse click or the beginning or ending of the sprite in the Score.

5 Select whether to turn speech on or off when the event you selected in the previous step occurs.

6 Enter the words to be spoken when text-to-speech is turned on.

7 Enter the words to be spoken when text-to-speech is turned off.

8 Click OK.

Using captioning

For users who are hearing impaired, you can add text captioning to your movies. Captioning is the practice of displaying text that corresponds to spoken narration or other sounds being played. Using the captioning behaviors in addition to the text-to-speech behaviors lets you make your movies accessible to users with all types of disabilities.

There are two captioning behaviors that are designed to be used together. Each is designed to be used with the keyboard navigation and text-to-speech behaviors. For information about applying text-to-speech behaviors, see "Enabling text-to-speech" on page 431.

To enable captioning, you attach the Accessibility Captioning behavior to an empty text sprite that displays the captions. Next, you attach the Accessibility Sync Caption behavior to a sprite that has already had speech enabled with the text-to-speech behaviors.

To apply the Accessibility Captioning behavior:

1 Place the text sprite that you want to display caption text on the Stage.

2 Drag the Accessibility Captioning behavior from the Library palette to the text sprite.

3 In the dialog box that appears, select the behavior group name for the scene. This step associates the accessibility behaviors in the scene with one another so they work properly.

4 Click OK.

Next, you attach the Accessibility Sync Caption behavior to a sprite that has already had speech enabled.

To apply the Accessibility Sync Caption behavior:

1 Apply the keyboard navigation behaviors to sprites in your scene. For more information, see "Enabling keyboard navigation" on page 429.

2 Apply the text-to-speech behaviors to sprites in your scene. For more information, see "Enabling text-to-speech" on page 431.

3 Drag the Accessibility Sync Caption behavior from the Library palette to the sprite that triggers the text-to-speech feature. This could be a text sprite that has the Accessibility Speak behavior attached or another sprite that triggers speech with the Accessibility Speak Member Text behavior attached.

4 In the dialog box that appears, select the behavior group name for the scene. This associates the accessibility behaviors in the scene with one another so they work properly.

5 In the first At Word text box, enter the number of the word in the text that is to be spoken where you want the captioning to begin. For example, if you want to begin captioning at the first word of the spoken text, enter the number **1**.

6 In the Initial Words text box, enter the number of words to display after the starting word number that you selected in the previous step. This is the number of words that can appear in the captioning sprite when the captioning begins. For example if you want the first section of text to begin at the first word and end at word 15, enter **1** in the previous step and **15** in the Initial Words text box. These words are replaced in the captioning display sprite by later sections of the text being spoken. You select the number of words in each section by using the remaining text boxes in the dialog box.

7 In the next At Word text box, enter the number of the word where the second section of text you want to display begins. For example, if you want the second section of displayed text to begin at word 16 of the spoken text, enter **16** in the At Word text box.

8 In the Display How Many of the Words That Follow text box, enter the number of words to display after the word number you selected in the previous step. For example if you want to display 22 words of text after word 16, enter **22** in this text box.

9 In the remaining text boxes, enter the numbers of the first word of each section of text followed by the number of words in each section. To use more than five sections, drop this behavior on the sprite again. This procedure can be repeated as often as necessary.

10 When you have finished entering values in the dialog box, click OK.

Accessibility scripting with Lingo or JavaScript syntax

If you have a basic understanding of Lingo or JavaScript syntax, you can write custom scripts to add text-to-speech functionality to your movies. For more information about Lingo and JavaScript syntax, see the Scripting Reference topics in the Director Help Panel.

First, you must initialize the speech software.

To initialize the text-to-speech software:

- Use the `voiceInitialize()` method.

The following frame script tests whether text-to-speech software is installed. If no software is installed, the script displays an alert dialog box.

```
on exitFrame
  if voiceInitialize() then
    _movie.go("Start")
  else
    alert "Text-to-speech is not available"
  end if
end
```

To determine the number of available voices:

- Use the `voiceCount()` method.

To return a property list that describes the name, gender, age and index number of the current voice:

- Use the `voiceGet()` method.

To return a list of property lists that describes all the available voices:

- Use the `voiceGetAll()` method.

To set a particular voice as the current voice:

- Use the `voiceSet()` method.

After you select a voice for speech synthesis, you can control the progress of the speech.

To begin speech synthesis:

- Use the `voiceSpeak()` method.

To temporarily pause the speech:

- Use the `voicePause()` method.

  Some speech engines might continue to speak for several seconds after the pause method is used.

To resume the speech:

- Use the `voiceResume()` method.

To stop speech synthesis:

- Use the `voiceStop()` method.

To check whether the speech is currently speaking, paused, or stopped:

- Use the `voiceState()` method.

To set the volume of the voice:

- Use the `voiceSetVolume()` method.

To set the pitch of the voice:

- Use the `voiceSetPitch()` method.

To determine the chronological number of the current word within the string being spoken:

- Use the `voiceWordPos()` method.

The following frame script tests whether the current voice is female and starts speech if it is. In this case, the `voiceSpeak()` method specifies the text of the cast member named "TextCommentary".

```
on exitFrame
  voiceProps = voiceGet()
  if voiceProps.gender = "female" then
    voiceSpeak(member("TextCommentary").text)
  end if
end
```

For a complete list of script terms that control text-to-speech and keyboard navigation, see the Scripting Reference topics in the Director Help Panel.

Deploying accessible movies

To successfully deploy an accessible movie, you must become familiar with the Speech Xtra extension's system requirements and download procedure as well as the experience users have with your accessible movie in real-world situations.

Adding the Speech Xtra

If you use the text-to-speech feature in a projector, you need to include the Speech Xtra in your movie's Xtra extensions list. Normally, Xtra extensions that are not included in a movie's default Xtra extensions list are added to the list when a cast member that requires one is added to the cast. You can view the default Xtra extensions list by opening a new movie and selecting Modify > Movie > Xtras.

The Speech Xtra is a script-only Xtra, which means that it adds methods and properties to Lingo or JavaScript syntax, but does not add support for any new cast member types. Because the Speech Xtra is not associated with any kind of cast member, you must add it to the movie's Xtra extensions list manually.

To add the Speech Xtra to the movie's Xtra extensions list:

1 Open the movie.

2 Select Modify > Movie > Xtras.

3 In the dialog box that appears, click Add.

A second dialog box appears.

4 In the alphabetical list of Xtra extensions, scroll to Speech, and select the Speech Xtra. (In Windows, the file name is Speech.x32.)

5 Click OK.

The Add Xtras dialog box closes.

6 Click OK again.

The Movie Xtras dialog box closes.

7 Save your movie.

System requirements

Windows computers must have Microsoft Speech Application Programming Interface (SAPI) 4.0 or later installed. SAPI 5 is recommended. Windows XP includes SAPI 5. Earlier versions of Windows do not include SAPI by default. It can be downloaded from the Microsoft website at www.microsoft.com/speech/. A separate screen reader application is not necessary.

Macintosh OS 8.6 and later include text-to-speech software. No additions are necessary.

Understanding the Xtra download process

Because the Speech Xtra is from Macromedia, it is considered a trusted download. The user does not have to interact with any dialog boxes for the download to occur. When a user encounters accessible Shockwave content for the first time, the Xtra downloads automatically. The Xtra is approximately 45K on Windows and 35K on the Macintosh. After the download is complete, the movie begins to play. When the user encounters accessible movies in the future, no download occurs because the Xtra is already present on their computer.

CHAPTER 21
Managing Memory

Macromedia Director MX 2004 has effective built-in memory management that is sufficient for most projects. To make memory available for new sprites, Director simply unloads the cast members used for sprites that are no longer on the Stage. However, sometimes large cast members, such as high resolution images, large sounds, or digital video, can take longer to load or unload than typical, smaller cast members. When a cast member takes a lot of time to load or unload, it can cause slight delays in movie playback.

If you test your movie on the lowest performance computers that you want it to be able to play back on, you can determine whether any of these delays occur and make changes to correct them. This chapter describes how Director's memory management works and the steps you can take to ensure smooth playback of movies with large amounts of media.

How Director unloads items from memory

To effectively manage memory while a Director movie is running, it is helpful to understand how automatic unloading of cast members works in Director. By becoming familiar with this process, you can make intelligent choices about when and how to perform memory management tasks yourself, if necessary.

A cast member is automatically loaded into memory when Director needs to draw a sprite of it on the Stage. Immediately after being drawn, each cast member is dealt with according to the value of its `purgePriority` script property. The default value of this property is 3. You can set this property in script or by selecting a number from the Unload menu in the Member panel in the Property inspector while the cast member is selected in the Cast.

The following are the possible values for `purgePriority`:

- When cast members with a purgePriority of 3 (normal) are no longer on the Stage, they can be unloaded from memory whenever Director needs memory for other tasks.
- Cast members with a purgePriority of 2 or 1 are only unloaded if memory is very low. They are added to the top of a list of recently used cast members that Director stores internally. This list is used to further prioritize cast members. Director assumes that the most recently used cast members are most likely to be used again and unloads them from memory only after all cast members with a `purgePriority` value of 3 are unloaded first.
- Cast members with a purgePriority of 0 are left in memory, and are not added to the recently used cast member list.

Monitoring memory use

The Memory inspector displays information about how much memory is available to Director for your movie and indicates how much memory different parts of the current movie use and the total disk space the movie occupies. It also can purge all removable items from RAM if you are about to perform a memory-intensive operation.

Note: The Memory inspector is not available on Macintosh.

To use the Memory inspector:

1 Select Window > Memory Inspector.

2 Observe the following memory use indicators:

Total Memory displays the total system memory available, including the amount of RAM installed on your computer and any available virtual memory.

Physical Memory shows the amount of actual RAM installed in the system.

Total Used indicates how much RAM is being used for a movie.

Free Memory indicates how much more memory is currently available in your system.

Other Memory indicates the amount of memory used by other applications.

Used by Program indicates the amount of memory used by Director (excluding the amount of memory used by the Director application file).

Cast & Score 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, cast members that use the Matte ink in the Score, thumbnail images in the Cast window, and any sounds, palettes, buttons, digital video movies, or linked files that are imported into the cast and are currently loaded into memory.

Screen Buffer shows how much memory Director reserves for a working area while executing animation on the Stage.

3 To remove all items that can be purged from RAM, including all thumbnail images in the Cast window, click Purge.

All cast members that have Unload (purge priority) set to a priority other than 0–Never (as specified in the Property inspector's Member tab) are removed from memory. This procedure is useful for gaining as much memory as possible before importing a large file. Edited cast members are not purged.

Whenever Director runs low on memory, such as when trying to load many large cast members, other cast members are automatically unloaded from memory, according to the following rules:

- The first cast members to be unloaded are those that have a `purgePriority` value of 3. This type of discarding is fast but is essentially random, which means you cannot predict which of these cast members unload at any given time. The worst-case scenario is that in order to load the cast member for channel 1 of a particular frame, the cast member that is needed for channel 2 of that same frame gets unloaded.
- When there are no more cast members with a `purgePriority` value of 3 left to unload, Director starts unloading the cast members that have a `purgePriorty` value of 2 and were least recently used. This type of unloading is slower but can be more predictable.
- If more memory is still needed after the cast members with a `purgePriority` value of 2 are unloaded, then the least recently used cast members with a `purgePriority` value of 1 are unloaded.

Using script to change a cast member's `purgePriority` property is a good technique for controlling memory management. However, changes to the `purgePriority` property do not take effect until the next time a cast member is used. For example, if you change the property in an `exitFrame` or `enterFrame` script in the same frame as a cast member is used, the cast member is treated as if it had its old priority because all drawing on the screen is done before any `enterFrame` or `exitFrame` scripts are performed.

Loading and unloading individual cast members

There are several script methods you can use to force specific cast members to load into memory or unload from memory. By using these methods, you can unload cast members that you know are no longer in use to make room for new ones. You can also force large cast members to load before they are actually needed. This prevents a pause that can occur when large cast members load normally, just before being drawn on the Stage.

Use the following methods to control cast member loading and unloading:

- To load a specific cast member or set of cast members, use the `preLoadMember()` method.
- To load all the cast members used in a specific frame or range of frames, use the `preLoad` method.
- To unload a specific cast member or set of cast members from memory, use the `unLoadMember()` method.
- To unload all the cast members used in a specific frame of your movie, use the `unload` method.
- To determine the number of bytes of memory required to display a range of frames, use the `ramNeeded()` method. To convert the number of bytes to kilobytes, divide the result by 1024.
- To determine the current amount of available memory, use the `freeBytes()` or `freeBlock()` method.

There are several more script terms related to memory management. For a complete list, see the Scripting Reference topics in the Director Help Panel.

Preloading digital video

It is recommended that you do not preload digital video cast members. Digital video is played by streaming the video file from a disk. As the file is streamed, it is decompressed into memory one section at a time. Preloading a digital video file causes the entire file to be decompressed into memory at once, which can cause low memory situations on most computers.

You can cause a digital video to preload only its first segment without consuming unnecessary memory by placing it in the Score a frame or two before it is actually needed.

To preload a digital video safely:

1. Add a sprite of the digital video to the Score.
2. Begin the sprite one or two frames before the frame where you want to display the video.
3. Locate this sprite off the Stage except for at least one pixel of one corner of the sprite. This is so the user won't notice it.
4. Set the sprite's `movieRate` property to 0, which prevents the movie from playing when first loaded.

5 In the frame where you want the video to appear, use script to set the sprite's `loc` property to place the sprite on the Stage in the location you select.

6 Set the `movieRate` property to 1 to start playing the video.

This way, the video loads its initial segment into memory and is ready to play immediately when your movie reaches the frame where you want it to appear. Experiment with preloading the video a few frames earlier if you find that preloading has not yet finished when your movie reaches the frame where the video appears.

CHAPTER 22
Managing and Testing Director Projects

As you work with Macromedia Director MX 2004 and develop projects with it, you might encounter some situations where a seemingly small change in the design of a project has a significant effect on the organization of the Director files and media files associated with it. A little bit of planning before you start a Director project can help avoid difficult changes during the course of development. Good testing practices can help you discover problems early in the project while they are still small and easily remedied.

This chapter provides a few simple guidelines that can streamline your development process.

Managing Director projects

By carefully managing the resources that go into your Director movies, you can avoid problems that might arise when you make changes to the movie or change the location of the movie file or its linked media.

The following guidelines can help make your project go smoothly:

- Before you begin a project, plan where media should be located when you deploy your movie and replicate that organization at the beginning of your project. This prevents the links between your Director movies and external media files from being broken when you move the project to a different location on a disk or to a different disk volume.

  The following structure is a good example of a simple file organization:

  ```
  Project Directory
    contains:
    MyDirectorMovie.dir
    Linked_Media subdirectory
      contains:
      sound files
      graphic files
      digital video files
      other linked assets
  ```

  By establishing the final organization of your files before you import them into Director, you prevent the need to update links later in the project.
- When you work on a large project, plan your basic approach to all aspects of the movie before you start construction. This way, you can find problems with your strategies before you have invested a lot of time building on them. Finding problems early in a project makes them much easier to solve.

- Organize the cast members in your cast in a logical way. You might choose to group all the cast members of a particular type together, or you might choose to group the cast members from each scene together. Choose a system that works for you and that will make it easy to find cast members when your cast becomes large. You can also choose to keep groups of cast members in separate casts.
- When referring to cast members, sprites, and frames in scripts, use the name of the cast member or sprite or the name of a frame marker. This avoids the need to change your code if you need to rearrange cast members, sprites, or frames during the project.

  For example, the following script refers to a cast member by its cast member number:

  ```
  member(16).text = "Good planning makes Director projects easy."
  ```

  If the text cast member 16 has to be moved in the cast, the script becomes invalid.

  Instead, use the following script:

  ```
  member("Output_text").text = "Good planning makes Director projects easy."
  ```

  Naming sprites makes changes to your script even easier. For example, the following script refers to a sprite by its number:

  ```
  sprite (1).text
  ```

  Instead, use the following script:

  ```
  sprite("input").member.text
  ```

  When you add a marker to a frame and use the marker name to refer to the frame, you can move the marker without breaking the script.

  The following script refers to a frame by its number:

  ```
  _movie.go(27)
  ```

  It is better to add a marker to the frame and use the marker name, as in the following script:

  ```
  _movie.go("Main_menu")
  ```

- During work sessions, save your movie file often. Save a copy of the movie after each milestone, such as a day of work or after adding a significant new feature or section. This way, if problems arise, you can easily compare the current version of the file to a slightly older version to locate the source of the problem. Keep several copies of your file at different stages of development in case you need to go back several steps.

About testing movies to avoid problems

While you develop movies, you might encounter some difficulties because creating interesting movies and trying out new ideas always involves some experimentation. By testing your movies according to the simple guidelines described in this section, you can prevent problems from becoming obstacles.

Testing early in development

When you begin a Director project, it is a good practice to test the functionality of your movie early in the development process to help ensure that you discover any problems while they are still minor. Waiting to test lets small problems become larger ones as you add features to your movie that depend on problematic functionality that was implemented earlier. By incorporating testing into your authoring process early, you will find these problems and have the opportunity to fix them before adding features to your movie.

Testing often during development

Testing should be an integral part of your Director development process. You should test the functionality of each small part of your movie as you add it rather than waiting until the movie's whole feature set is implemented.

When you build features that are interdependent, you should test each one before adding the next. If you test this way, you know that the most recently implemented feature is the most likely source of the problem. If you wait to test one feature until after the next feature is implemented and one of those features exhibits a problem, you have a more complex set of possibilities to evaluate.

Save multiple versions of your movie as you progress. When difficulties arise, compare the current version with the last saved version to help locate the source of the problem.

Testing on all target platforms

When you develop a Director movie, you should spend some time defining its audience. Part of this process is deciding what the minimum system requirements should be for the computers used by that audience.

You should determine the slowest processor speed you want your movie to play on and verify that the performance of your movie is acceptable on a processor of that speed. You should also determine if there is a range of configurations you have to support (such as Macintosh, Windows 2000, and Windows XP) and test enough of them to ensure success. Be sure to include parameters such as browser software, screen resolution, and available memory in your testing.

This approach can help you find problems that are specific to an operating system or configuration, which are distinct from authoring errors.

Testing strategies

You can use the following strategies to test your movies effectively:

- Testing in the Director authoring environment is different than testing in the Director Projector and Macromedia Shockwave environments for online content. The Preview in Browser command demonstrates the true behavior of the movie in the Shockwave environment.
- If you encounter a problem, try to isolate the problem in a new Director file that incorporates only the problem feature or item. Make a list of the minimum steps that are required to reproduce the problem in a new file. This process usually reveals the source of the problem in your movie. It also reveals whether the problem is limited to one feature or if it is caused by the interaction of two or more features in your movie.
- Try to re-create the problem with different media. Sometimes the source of a problem is within a specific media item used in your movie.
- Try to re-create the problem on a different computer. This helps isolate problems with hardware configuration or with the Director installation on a specific computer. If the problem exists only when the movie is posted to a server, determine whether the problem exists on only one server or all servers. Occasionally the server's MIME types might need editing to include the MIME types for Director. For more information, see Director TechNote 16509, Configuring your server for Macromedia Shockwave Player, at www.macromedia.com/support/director/ts/documents/shockwave_config.htm.
- When you use script, look for typing errors, missing punctuation, or inconsistent naming.
- JavaScript syntax is case-sensitive, while classic Lingo is not.

Printing movies

You can print movie content to review it and mark changes, to distribute edits to a team, to make handouts from a presentation, or to see your work on paper. You can print a movie while in authoring mode in several ways. You can print an image of the Stage in standard or storyboard format, 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, the Score, and sprite scripts), the comments in the Markers window, the Cast window artwork, or the entire Cast window.

You can also use script to control printing. For more information, see the Scripting Reference topics in the Director Help Panel.

To print part of a movie:

1 Select File > Print.

2 To specify what part of the movie to print, select an option from the Print pop-up menu.

   You can print an image of the Stage, the Score, all scripts or a range of scripts (movie, cast, the Score, and sprite scripts), cast text, cast art, cast thumbnails, and the comments in the Markers window.

   The Scripts, Cast Text, Cast Art, and Cast Thumbnails print options specify a range of casts and cast members—internal or external. Information that appears in the Print dialog box depends on the selection to be printed.

   Note: Selecting Cast Text on the Print pop-up menu, prints a table of text cast members your printer's set resolution.

3 To specify which frames of your movie are printed, select one of the following Frames options:

Current Frame prints the frame that is currently on the Stage.

Selected Frames prints the frames that are selected in the Score.

All prints all of the frames in your movie.

Range prints the range of frames specified in the Begin and End text boxes.

4 To specify which frames in the defined range to print, select one of the following Include options:

Every Frame is the default setting and prints every frame that is specified in Range.

One in Every [number] Frames prints frames at the interval you specify in the text box. For example, if you enter 10, Director prints every tenth frame.

Frames with Markers prints only the frames that have markers in the Score window.

Frames with Artwork Changes in Channel [number] prints the frames in which cast members move or in which new cast members are introduced in the Score. Specify the channel number in the text box.

5 To determine the layout of the items to print, click Options and select from the following:

Scale provides options to print at 100%, 50%, or 25% of the original size.

Frame Borders creates a border around each frame.

Frame Numbers prints the frame number with each frame.

Registration Marks places marks on every page to align the page for reproduction.

Storyboard Format is available only when you select 50% or 25% images to print. This option places marker comments next to the frame image.

Date and Filename in Header prints a header on each page. The header consists of the name of the Director movie and the current date.

Custom Footer prints a footer on each page. Type the footer in the text box.

The image at the left of the dialog box previews the layout options.

Resources

The following resource is available to help you with testing and troubleshooting your Director movies: index of Director testing and troubleshooting TechNotes at www.macromedia.com/go/director_troubleshooting

CHAPTER 23
Packaging Movies for Distribution

When you finish authoring your Macromedia Director MX 2004 movie, you have a choice of several ways to prepare it for distribution. You can distribute a movie in the Macromedia Shockwave format that plays in a browser, or you can distribute it as a stand-alone projector. Stand-alone projectors can contain the software necessary to play the movie, or they can use an installed Macromedia Shockwave Player to play the movie independent of a browser. You can also export a movie as a digital video.

You can use several Director features to prepare movies for distribution. These features include determining Publish settings and deciding which Xtra extensions to include, exclude, or download. You can also preview your movie in a browser and batch-process movie files to compress them and protect them from being edited.

About distributing movies

When you finish creating a movie, you have several choices about how to distribute it to users. You can distribute the movie as Shockwave content that plays within a web page or as a projector that downloads to the user's computer or is distributed on a disk.

- Shockwave content is a compressed version of the movie data only.
- A projector is a stand-alone version of a movie. You can include one movie in a projector that links to other external movies or include several movies in a single projector. Projectors are handled by the system as executable application files.

Movies that are distributed from the Internet can begin playing as soon as the content for the first frame is downloaded. This process is called streaming. You can control streaming with behaviors that make the movie wait for media at certain frames, or you can specify that a movie download completely before it begins playing. For more information, see "Setting movie playback options" on page 471.

- To create Shockwave content that can play in a web page, you use the File > Publish command. Director leaves your original movie in its DIR format. Director also creates Shockwave content in the DCR format.

If you use the default Publish settings, Director creates an HTML page that is completely configured with `EMBED` tags and everything else you need to run your movie in a browser. By default, Director saves all these new files in the same folder as your original Director movie. For more information about putting your Director movie on the web, see "Creating Shockwave content" on page 452.

For more information about how to distribute Xtra extensions with projectors, see TechNote 13965 in the Director Support Center at www.macromedia.com/go/director_support. Although the note might refer to Director 7, the information is the same for more recent versions of Director.

Shockwave browser compatibility

Shockwave can play Director movies in the following browsers on the listed platforms:

- For Microsoft Windows: An Intel Pentium II with 64 MB of available RAM running Windows 98, or an Intel Pentium III with 128 MB of available RAM running Windows 2000 or Windows XP; one of the following web browsers: Netscape 7.1, Microsoft Internet Explorer 5.01 Service Pack 2, Microsoft Internet Explorer 5.5 Service Pack 2, Microsoft Internet Explorer 6 Service Pack 1, and a color monitor
- For Macintosh OS X: A Power Macintosh G3 with 128 MB of available RAM running Mac OS X 10.2.6 or 10.3; one of the following web browsers: Netscape 7.1, Microsoft Internet Explorer 5.2 or later, or Safari; and a color monitor
- For Macintosh Classic: A Power Macintosh G3 with 64 MB of available RAM running System 9.2; Microsoft Internet Explorer 5.1; and a color monitor

When it first encounters an HTML page that references Shockwave content, Internet Explorer for Windows asks the user for permission to download the Shockwave ActiveX control if it is not already installed. If the user approves, it downloads and installs the control.

Previewing a movie in a browser

You can preview a movie in a browser on your local computer to view JPEG-compressed bitmaps, and to check the movie design, script, and any other performance issues related to playing a movie in a browser. Previewing a movie creates temporary Shockwave (DCR) and HTML files that open in a browser.

Note: When you use the Publish command rather than the Preview in Browser command (with the Preview after Publishing option selected in the Publish Settings Format tab), you can create permanent DCR and HTML files. These permanent files (as opposed to temporary ones created simply by using the Preview in Browser command) can then be placed on a web server for viewing with a browser via http.

You may notice that linked media do not work as expected when you preview a movie in a browser. Because of security restrictions, movies playing in browsers may not be able to read files from a local disk unless they are in the dswmedia folder (also called the support folder), which is a subfolder of the folder containing the Shockwave Player. Therefore, to preview a movie that uses linked media, you may need to put the movie and all of its linked media in the dswmedia folder. The movie can open any file in a subfolder of dswmedia, provided that the relative paths have not changed. If you move the movie and its media to another server, the linked media will continue to work if you preserve the same folder structure. For more information, see "Using dswmedia folders and the support folder to publish Shockwave content".

To specify the browser to use for previewing:

1 Select Edit > Preferences > Network.

 Note: If you are using a Macintosh OS X operating system, select the Director menu, instead of the Edit menu, to access Preferences.

2 In the Preferred Browser box, enter the path to the browser application file.

To preview a movie in a browser:

- Select File > Preview in Browser or press F12.

About Xtra extensions

All Xtra extensions a movie requires must be installed on your user's system when the movie runs. When you distribute a movie, you must either include these Xtra extensions or provide the user with the means to download them. Using the Movie Xtras dialog box (select Modify > Movies > Xtras), you can specify the Xtra extensions to include in a projector and whether Xtra extensions should download for use with Shockwave content. The Movie Xtras dialog box contains a list of the most commonly used Xtra extensions. Including all these Xtra extensions ensures that your movie will work in most cases but makes the projector much larger. You may want to remove Xtra extensions you know you aren't using to help manage memory.

Each time you create a sprite that requires an Xtra, Director adds the Xtra to the list of required Xtra extensions in the Movie Xtras dialog box. If you remove the sprite, Director does not remove the Xtra from the list, in case you later re-create the sprite. Director cannot detect Xtra extensions required in Lingo or JavaScript syntax. You must manually add any Xtra extensions required by your scripts to the list in the Movie Xtras dialog box. For more information, see "Managing Xtra extensions for distributed movies" on page 450.

Managing Xtra extensions controls the size and capabilities of the movie you distribute. Many important features in Director, such as text and vector shapes, are controlled by Xtra extensions, as is the ability to import all types of linked media. If you don't use a feature or import a media type that is controlled by an Xtra, you should not distribute the related Xtra with your movie. This is especially true for movies distributed on the Internet.

Including Xtra extensions

The Shockwave Player includes the Xtra extensions that support the most common features and media types. These include text; vector shapes; Macromedia Flash; BMP, PICT, JPEG, and GIF file importing; sound management; and Shockwave Audio.

Xtra extensions not included with the Shockwave Player must be installed in a user's system before the movie plays. Use the Download If Needed option in the Movie Xtras dialog box to make the movie prompt the user to download the Xtra. Director downloads Xtra extensions from the URL specified in the Xtrainfo.txt file in the Director application folder.

Xtra extensions downloading from projectors requires use of Lingo or JavaScript syntax. For more information, see the Scripting Reference topics in the Director Help Panel.

Xtrainfo.txt includes URLs for all Macromedia Xtra extensions included with Director, but you may need to manually edit Xtrainfo.txt to add the URL for third-party Xtra extensions or Macromedia Xtra extensions not included with Director. Xtrainfo.txt includes a description of how to enter this information. Xtra developers may also provide installation programs or other means of modifying Xtrainfo.txt automatically.

If a user chooses to download an Xtra extension, Director retrieves the Xtra extension from the URL specified in Xtrainfo.txt using the Verisign download security system. Verisign is a standard means of downloading software from secure sources.

You can also include Xtra extensions in projector files. Select the Include in Projector option in the Movie Xtras dialog box for any Xtra extension you want to include.

You can also exclude Xtra extensions in projectors. For more information, see "Excluding Xtra extensions from projectors" on page 462.

Xtra extensions usually required for the movie to play back correctly include the following:

- Xtra extensions that create cast members (text, Flash, vector shapes, Windows Media, QuickTime, and so on)
- Shockwave Audio Xtra extensions (if the movie uses files in the SWA format)
- Transition Xtra extensions (if the movie uses third-party transitions)
- Import Xtra extensions, if the movie uses nonstandard types of linked external cast members
- Network Xtra extensions required for a movie to access the Internet
- Scripting Xtra extensions (if the movie uses any special script that requires Xtra extensions).

Managing Xtra extensions for distributed movies

You can manage Xtra extensions for your movie using the Modify menu.

To manage Xtra extensions for the current movie:

1. Select Modify > Movie > Xtras.
2. To add or remove Xtra extensions, do any of the following:
   - To add the Xtra extensions required to connect a projector to the Internet, click Add Network.
   - To restore the list of default Xtra extensions, click Add Defaults.
   - To manually add an Xtra extension to the list (which you would do, for example, if you've used Lingo or JavaScript syntax that requires Xtra extensions), click Add and select from the list of Xtra extensions installed in your system.
   - To delete an Xtra extension from the list, highlight the Xtra extension and click Remove.
   - To get information about an Xtra extension (when available), highlight the extension and click Info.
3. To change settings for Xtra extensions in the list, select an Xtra extension and select either of the following options:

   Include in Projector makes Director include the selected Xtra extension in any projector that includes the current movie.

   Download If Needed makes the movie prompt the user to download a required Xtra extension if it is not installed in the user's system. The Xtra extension is downloaded from the location specified in the Xtrainfo.txt file and permanently installed in the user's system.
4. To get information about a selected Xtra extension, click Info.

   The information comes from an Internet source. Not all Xtra extensions include information. Third-party Xtra extensions often include some explanations and information about the developers.

Note: Another way to include Xtra extensions with a movie is to create an Xtras folder containing all required Xtra extensions in the same folder as a projector file. This allows you to see which Xtra extensions are included without opening the movie. If you use this method, you cannot include Xtra extensions in the projector file because the movie will fail to initialize.

About distribution formats

Before deciding how to distribute a movie, it helps to understand how Director plays movies. Director movies play either with the Shockwave Player or through a projector player. The Shockwave Player is a system component that plays movies in web browsers and also outside browsers as stand-alone applications. A projector player can only play movies independently of a web browser.

You can distribute movies as Shockwave content (with the DCR extension), projectors, Shockwave projectors, or protected movies (DXR extension). You should not distribute source movies (DIR extension) unless you want your users to be able to change the movie in the Director authoring environment.

- Shockwave content is a compressed version of a movie's data and does not include a player. Shockwave content is created primarily to distribute over the Internet for playback in a web browser. Another reason to create Shockwave content is to compress it for distribution on a disk when the movie is contained in a projector. In addition to compressing the data, saving a movie in the Shockwave format removes all information necessary to edit the movie.
- A projector is a movie intended for play outside of a web browser. A projector can include a player (called the Standard player), Xtra extensions, multiple casts, and linked media in a single file. A projector can also include several different movie files. Configured in this way, a projector can be a completely stand-alone application.
- A Shockwave projector makes a much smaller projector. A Shockwave projector uses an installed Shockwave Player on the user's system to play a movie instead of including the player code in the projector itself. If no Shockwave Player is installed on the user's system, the user must download a copy. A Shockwave projector is excellent for distributing movies on the Internet that you don't intend to play in a web browser.

  You can also reduce the file size of a projector by turning on projector options that compress the movie data, the player code, or both. In Windows, compressing the player code reduces the minimum projector size from approximately 2.1 MB to 1.1 MB for a projector, and to about 60K for a Shockwave projector.

  On the Macintosh, compressing the player code reduces the minimum projector size from approximately 2.5 MB to 1.2 MB for a projector, and to approximately 12K for a Shockwave projector.
- Protected movies (.DXR extension) are uncompressed movies that users can't open for editing. These can be useful when you want to distribute uncompressed movies on a disk, but you don't want users to edit the source file. Protected movies may play faster than Shockwave content from a disk because they do not need to be decompressed. These movies are preferable if disk space isn't limited. Like Shockwave content, protected movies do not include the information necessary to edit the movie or the software that plays the movie. They can be played only by a projector, a movie in a window, or the Shockwave Player.

Note: To edit a movie packaged for distribution, you must edit the source file (DIR) and create a new movie in one of the distribution formats. Always save your source files.

Using linked media on the Internet

When you distribute a movie on the Internet for playback in a web browser, the linked media must be at the specified URL when the movie plays. Otherwise, the user receives an error message.

Distributing movies on a disk

Whenever a movie plays from a disk, it accesses all external linked files the same way that it did in the authoring environment. All linked media—bitmaps, sounds, digital videos, and so on—must be in the same relative location as they were when you created the movie. To make sure you don't forget any linked media when you distribute a movie on a disk, place linked files in the same folder as the projector or in a folder inside the Projector folder.

If your movie includes Xtra extensions, you must include the Xtra extensions in the projector. If a movie distributed on a disk connects to the Internet in any way, be sure to click the Add Network button in the Movie Xtras dialog box.

Distributing movies on a local network

If you plan to place a movie on a local area network (LAN), all files must be set to read-only, and users must have read/write access to their system folders. Otherwise, the requirements are the same as for normal disk-based distribution.

Creating Shockwave content

You save your work as a Shockwave (DCR) movie to prepare it for playback in a Shockwave-enabled web browser, or to make disk-based movies smaller. Using Shockwave content also prevents your users from editing the movie if they own Director.

If the Shockwave content you're creating will be distributed on the Internet and requires any Xtra extensions, make sure the Xtra extensions are listed in the Movie Xtras dialog box and that Download If Needed is selected for each required Xtra extension. For more information, see "Managing Xtra extensions for distributed movies" on page 450.

Note: Use Update Movies to convert several movies at once to the Shockwave format. For more information, see "About cross-platform projectors" on page 463.

For more information about Shockwave content, see Chapter 24, "Using Shockwave Player," on page 469.

Using default Publish settings

To create Shockwave content, you use the File > Publish command. The default setting creates a Projector file with Preview enabled.

For more information about changing Publish settings, see the next section.

To create a Projector movie using default settings:

1 Select File > Publish.

2 Save your movie if prompted to do so.

Director creates and automatically starts a Projector version of your movie.

Director creates a Shockwave version of your movie, and an HTML file. Your default browser opens with the HTML page you just created.

Note: Your default browser is specified in your Network Preferences dialog box. To change your default browser, select Edit › Preferences › Network. (If you are using a Macintosh OS X operating system, select the Director menu, instead of the Edit menu, to access Preferences.)

If your movie needs Xtra extensions that fall beyond the range of the default publish settings, (for example, Windows Media) you will be prompted to add them. In that case, you can set up your movie with the correct settings by changing them.

Changing Publish Settings

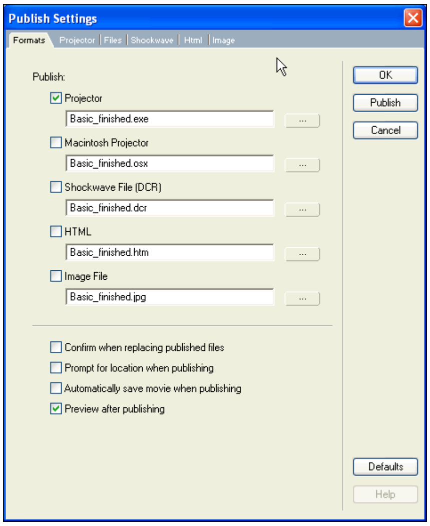
You can change Publish Settings by using the Publish Settings dialog box. Publish Settings lets you determine what type of Director movie you want to create, and what properties that movie shall have.

To access Publish settings:

- Select File > Publish Settings.

  The Publish Settings dialog box appears with the following tabs: Formats, Projector, Files, Shockwave, Html, and Image.

  Each Publish Settings tab contains the following buttons: OK, Publish, Cancel, Save As Defaults, Defaults, and Help.



Publish:

- [✓] Projector
  Basic_finished.exe
- [ ] Macintosh Projector
  Basic_finished.osx
- [ ] Shockwave File (DCR)
  Basic_finished.dcr
- [ ] HTML
  Basic_finished.htm
- [ ] Image File
  Basic_finished.jpg

- [ ] Confirm when replacing published files
- [ ] Prompt for location when publishing
- [ ] Automatically save movie when publishing
- [✓] Preview after publishing

- To save any changes you have made, click OK.
- To Publish your movie with the selected settings, click Publish.
- To Cancel any changes you have made, click Cancel.
- To save the current settings of the selected tab as default settings, click Save As Defaults.

  Note: Clicking the Save As Defaults button on one tab has no effect on any of the other tabs.

- To return settings of the currently selected tab to their saved default state, click Defaults. To return settings of the currently selected tab to their original default state, hold down Alt (Windows) or Option (Macintosh) and click Defaults. If you have not saved default settings, clicking Defaults without holding down the key returns the settings to their original default state.

  Note: Clicking the Defaults button on one tab has no effect on any of the other tabs.

- To access help, click Help.

Selecting a publishing format

Use the Formats tab to select the movie format you want and determine the types of files you wish to publish. It includes support for creating cross-platform projectors.

To set options using the Formats tab:

- To publish a Projector movie, click Projector. You can set more Projector options by using the Projector tab. For more information, see "Setting Projector options" on page 455.
- To publish a projector that will run on a different platform than the one in which you are authoring, select Macintosh projector (Windows) or Windows projector (Macintosh). For more information about creating cross-platform projectors, see "About cross-platform projectors" on page 463.
- To publish a Shockwave file (DCR) without an HTML file, select Shockwave File. You can set additional Shockwave options using the Shockwave tab. For more information, see "Setting Shockwave options" on page 456.
- To publish Shockwave content with an HTML file, select HTML; you can set additional Shockwave HTML properties using this tab. For more information, see "Setting HTML options" on page 456.
- To publish an Image file, select Image file. You can set additional Shockwave image options using this tab. For more information, see "Setting Image options" on page 458.
- To be prompted if you want to replace existing files, select Confirm when replacing files.
- To be prompted for the location you want to publish to, select Prompt for location when publishing.
- To automatically save your movie when publishing (instead of being prompted to save as is the case with default settings), select Automatically save when publishing.
- To change the default setting of automatically previewing your movie in a browser after publishing, deselect Preview after publishing.

Using the DCR and HTML file format

If you create a DCR file and an HTML file with all of the tags necessary to display your DCR movie, Director does the following:

- Creates a DCR and HTML file in the same directory as your Director (DIR) movie.

  Note: Director creates a CCT file for each external cast and, by default, saves the CCT file in the same folder as the DCR file. To specify a different file location, hold Alt (Windows) or Option (Macintosh) when you select File > Publish. Continue to hold the key for access to dialog boxes that let you specify new paths for both your DCR and CCT files.

- Gives both your DCR and HTML files the same name as your DIR file, with the appropriate extensions (for example, MyMovie.dcr and MyMovie.html).
- Sets the DCR movie's width and height to match the dimensions of the DIR movie.
- Configures the DCR movie and HTML file so that if your users resize their browsers, the DCR movie remains the same size as the original DIR movie.
- Compresses bitmap images and sound using JPEG compression. Note that if you've compressed images for individual cast members, those settings will override compression Publish settings are movie specific; that is, if you change the default Publish settings, Director saves those changes when you save your movie. A new movie uses the Default Publish settings, not those set for a specific movie.

Setting Projector options

The Projector tab lets you set options that include how the projector movie appears on stage or in a browser, and what type of player it is shown on.

To set options using the Projector tab:

1 Select File > Publish Settings and click the Projector tab.

2 Set the options you want and click OK to keep your settings and return to the movie or Publish to begin the projector publishing process.

For more information about setting Projector options, see "Creating projectors" on page 460.

Setting Files options

The Files tab lets you set file options for projectors. For example, by default a projector consists of just the current movie, but you can add external casts to the movie using this tab. You can also exclude Xtra extensions from your projector.

To set options using the Files tab:

1 Select File > Publish Settings and click the Files tab.

2 To set the Primary movie components for your projector, select from the following options:

Include linked cast files lets you include automatically all linked cast files to the primary movie.

Exclude all xtras lets you exclude some or all Xtra extensions from your projector. For more information, see "Excluding Xtra extensions from projectors" on page 462.

Compress files (Shockwave format) lets you compress files for a Shockwave projector as you would for standard Shockwave content.

Add Files opens a dialog box that lets you select and add Director Movie or Cast files to the current projector.

Remove All deletes all added files.

Play every movie in list ensures all movies in your movie list play.

Loop sets the movie to keep playing.

3 Click OK to save changes and return to the movie.

4 Click Cancel to reject changes and return to the movie.

5 Click Publish to publish the projector with these settings.

Setting Shockwave options

The Shockwave tab lets you set options that determine the Shockwave Player version, how images are compressed, how users interact with the movie once it's published, and more.

To set options using the Shockwave tab:

1 Select File > Publish Settings and select the Shockwave tab.

2 Select the options you want to provide for your users, as described in the following list:

Version:

- Select Shockwave Player 10 or Shockwave Player 8.5.

Image compression:

- **Standard:** Select Standard to apply compression techniques used by Director in versions 4 through 7, select Standard. This setting is suitable for graphics with few colors.
- **JPEG:** Select JPEG and specify the image quality setting by moving the slider to a value between 0 and 100 percent. The higher the percentage, the less the image is compressed.

Compression enabled lets you compress the sound in your movie. Select this option and select the level of compression from the kBits/second pop-up menu. For more information about sound compression, see "Compressing internal sounds with Shockwave Audio" on page 237.

Convert Stereo to Mono lets you convert stereo audio to monaural. This option is only available if Shockwave Audio Compression is enabled.

Enabled context menu items:

Include Cast Member Comments lets you include comments you might have entered in the Comments field of the Property inspector for your cast members. You can then use Lingo or JavaScript syntax to access the comments in the DCR file.

- **Volume Control** lets users adjust the volume of the movie's soundtrack.
- **Transport Control** provides controls for rewinding, stopping, and stepping through the movie.

Setting HTML options

If users view your movie in browsers, chances are they will resize their browsers. How your movie behaves when the browser size changes depends on how you set HTML options in the Publish Settings dialog box.

To set HTML options for browser resizing:

1 Select File > Publish Settings.

2 On the Html tab of the Publish Settings dialog box, select from the Dimensions pop-up menu. When you make a selection, the width and height values default to the movie size.

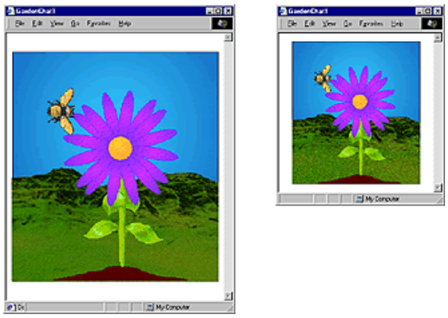
- To change the background color of your HTML file, either click the Page Background color box and select a color, or enter a value in the hexadecimal field.
- To make the DCR movie match the dimensions of your DIR movie, select Match Movie in the Dimensions field.

3 If you use the default Match Movie setting in the Dimensions field, values in the `OBJECT` and `EMBED` tags in the HTML file are set to the exact dimension of your movie. To change the dimensions, select either Pixels or Percent of Browser Window, and type the new dimensions in the Width and Height fields. Your movie will resize to fit the new rectangle only if you have not selected No Stretching in the Stretch Style pop-up menu on the Shockwave tab.

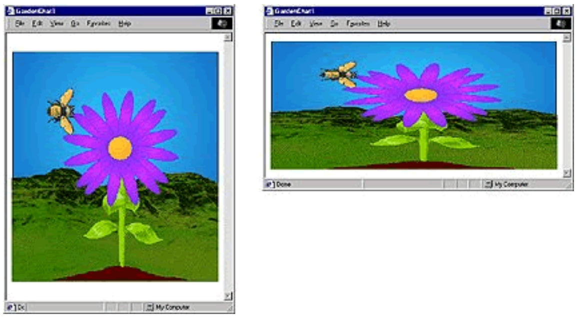
- To create an HTML file with parameters that match the height and width of the movie, select Match Movie.
- To specify height and width values in the HTML file in pixels, select Pixels.
- You can select Percentage of Browser Window, and specify a percentage in the Width and Height fields. (To make browser resizing affect the size of the DCR movie, you must specify percentages and select either Preserve Proportions, Stretch to Fill, or Expand Stage Size on the Shockwave tab of the Publish Settings dialog box.)

4 On the Html tab, select an option from the Stretch Style pop-up menu.

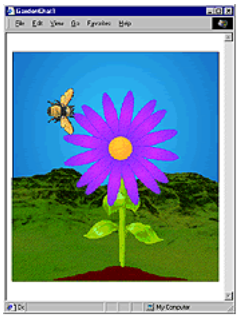
- To specify that your movie not resize at all, select No Stretching.
- To maintain the same aspect ratio of your original Director movie no matter what size the user makes the browser, select Preserve Proportions. The movie will fit within the width and height parameters, as much as possible, while preserving the movie's aspect ratio. The movie aligns within the window based on the align tags that you specify in step 4.

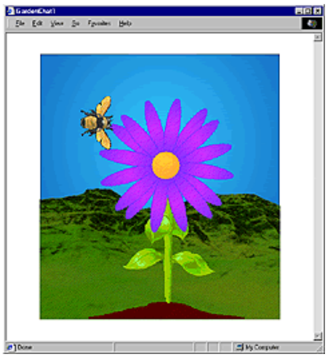


- To change the size of the movie to fit the size of the browser, select Stretch to Fill. Any browser resizing stretches the movie to fill the width and height parameters. Note, however, that if the aspect ratio of the movie changes, sprites on the Stage could appear distorted. If you select Stretch to Fill, Director ignores the align tags that you specify in step 4.



To let users resize the Stage without resizing the sprites, select Expand Stage Size. The movie is aligned within the browser based on the align tags that you specify in step 4.





5 To specify align tags for your movie, use the Horizontal Align and Vertical Align pop-up menus. You can select left, center, or right horizontal alignment, and top, center, or bottom vertical alignment.

Setting Image options

You can specify the images that should appear if the user doesn't have Shockwave or the ActiveX control.

To set options using the Image tab:

1 Select File > Publish Settings and select the Image tab.

2 Select from following Alternate Shockwave image options

- In the Poster Frame field, enter the frame number from your movie's Score that you want to appear as a JPEG image for users who are unable to view your movie.
- To specify compression for the image, move the Quality slider to the desired compression setting. The higher the percentage, the less the image is compressed.
- To specify that the image download as a progressive JPEG, select Progressive. The JPEG will then display at low resolution and increase in quality as it continues to download. Making a JPEG progressive also reduces its file size.

3 Select Publish to publish the movie with the chosen settings or click Cancel to cancel any changes made.

4 Select Default to return the image options to their default setting.

5 Click OK to save the settings and return to the movie.

Using dswmedia folders and the support folder to publish Shockwave content

Use a dswmedia folder to obtain data from a local source while a movie is playing in safe player mode. Director treats folders named dswmedia as exceptions to the safe player mode feature that normally stops movies from reading data from any local source. Any linked media or cast files in a dswmedia folder, or a subfolder of any depth, may be read by a movie running in safe player mode. You can use relative links or full file paths to files in dswmedia folders.

To test movies in a browser locally before uploading them to your web server, place the movie, linked casts, and linked media in a directory tree within a dswmedia folder and use relative links. When you move the movie and its media to a web server for testing, keep the directory structure intact. You must use file and folder names which do not have spaces or capital letters, and which have recognized file extensions like .dcr and .gif, in order for them to be accessible from your server.

To access local media while a movie is playing in safe player mode, install the media in a dswmedia folder in a known location on the user's computer. This location can be a CD that you provide.

Using linked media when testing Shockwave content

Linked media do not show up when testing Shockwave content from a local machine. However, when the movie is uploaded to a server, the media show up fine.

Shockwave content is played in what is called 'safe player mode,' which is a feature specific to Director movies. When a movie is played in safe player mode, it is restricted. The movie cannot read or write files on the user's local disk, except to get or set a movie's preferences. Therefore, linked files do not show up when played on a local machine.

There is one exception to the aforementioned restriction. A movie can read files that are stored in a folder named 'dswmedia' (or a sub folder).

To enable linked files to show on a local machine, follow the steps below:

1 Create a folder named 'dswmedia'.

2 Place all linked media in the 'dswmedia' folder or Place all files including the Director file in a dswmedia folder in order to maintain file and folder structure.

Note: Be sure to maintain the original file and folder structure of your project, so as not to break any links.

Converting movies created in previous versions of Director

Director MX 2004 can convert movies from Director 5 and later. It's not necessary to update movies created in Director 8 or Director 8.5 for use in Director MX 2004.

You can also update movies to Director MX by simply opening and saving them, but the Update Movies command is faster for converting large projects. It's also more effective for preserving links to external media. For more information, see "Processing movies with Update Movies" on page 464.

Note: Macromedia Shockwave Player 8.5 can play Shockwave content created with Director 5, 6, 7, 8, and Director MX.

To be included as a file in a Projector, a movie must be created with Director 8.5 or newer. However, using Lingo, movies created with earlier versions of Director can be played (e.g. go to movie, play movie) back to and including movies created with Director 5.

When you open a Director 7 movie in Director MX or convert it to the new format with Update Movies, the following changes occur to the movie:

- The data structure is changed to the latest file format.
- Shapes are not converted to the new Bézier shapes.
- Ink functionality is not updated unless you turn off Maintain Outdated Ink Mode Limitations in the Movie Properties dialog box.
- Old Score data, from versions of Director before Director 5, is converted to the new Score, combining adjacent frames in the old Score containing the same cast members into single sprites in the new Score. You might want to split or join sprites to make working in the Score more convenient.

About projectors

A projector is a stand-alone version of a movie. Projectors require certain Xtra extensions to use text, Flash content, connect to the Internet, and use certain other features. Director includes the most common required Xtra extensions by default. You can include or exclude Xtra extensions for each movie using the Include in Projector option in the Movie Xtras dialog box. You can also add Xtra extensions to a projector manually the same way you select movie files. You can create Macintosh projectors using a Windows application and create a Windows projector using a Macintosh application. For more information, see "About cross-platform projectors" on page 463.

In addition to the standard projector, you can create a fast-start projector, which typically starts faster. A fast-start projector doesn't include Xtra extensions inside the projector itself, so there's nothing to unpack.

Creating projectors

You can include only Director MX 2004 movies in projectors. You can use the Update Movies command to convert older movies to the latest version of Director. For more information, see "Converting movies created in previous versions of Director" on page 459.

To create a projector:

1 Select File > Publish Settings.

   The Publish Settings dialog appears.

2 From the Formats tab, select Projector in the Publish section of the dialog box.

3 From the Player type pop-up menu on the Projector tab, select Standard to create a standard Projector or select Shockwave to create a Shockwave projector.

   Standard includes the uncompressed player code in the projector file. This option starts the movie faster than other options but creates the largest projector file.

   Shockwave makes the projector use the Shockwave Player installed in a user's system instead of including the player code in the projector file. If the Shockwave Player is not available when the movie runs, the movie prompts the user to download it.

4 Select from the following options:

Animate in Background lets the movie continue playing if a user switches to another application. This is useful if you want the movie to continue running in the background when its window is not active. If this option is not selected, the movie pauses when the user switches to another application and resumes when the user switches back.

Full screen sets the movie to fill the entire screen when playing.

Center stage in monitor insures that the movie stays in the center of the screen when playing.

Escape key terminates lets the user press the ESC key to exit the movie.

Reset Monitor to Match Movie's Color Depth automatically changes the color depth of your monitor to the color depth of each movie in the projector play list. For example, if you are working on a color monitor set to 256 colors and a movie in the play list was created in thousands of colors, the monitor automatically switches to thousands of colors.

Extra main memory

System temporary memory (Macintosh only) makes Director use available system memory when its own partition is full.

5 Once all Projector options are set, click OK or click Default to return the settings to their defaults.

Note: To avoid problems with linked media, create the new projector in its final folder location and do not move it to a different folder. Director turns the movies, casts, and included Xtra extensions into a single projector.

You can also set options for the files that are associated with the Projector movie. By default, a projector consists of just the current movie. However, you can add external cast files and exclude Xtra extensions, among other options.

To set Files tab options:

1 Select File > Publish Settings.

2 On the Formats tab, click the Projector option.

3 Click the Files tab.

4 To set the Primary movie components, select from the following options:

Include linked cast files includes any linked cast files in the projector.

Exclude all xtras lets you exclude some or all Xtras from your projector. For more information, see "Excluding Xtra extensions from projectors" on page 462.

Compress files (Shockwave format)

Add Files opens a dialog box that lets you select additional Director Movie or Cast files to the current projector.

Remove All deletes all added files.

Play every movie in list ensures all movies in your movie list play.

Loop sets the movie to keep playing.

5 Click OK to save changes and return to the movie.

6 Click Cancel to reject changes and return to the movie.

7 Click Publish to publish the movie with these settings.

To create a fast-start projector:

1 Create a new folder on your computer desktop.

It does not matter what you name the folder.

2 In Director, select Modify > Movie > Xtras.

The Movies Xtras dialog box appears.

3 Select the name of each Xtra and deselect Include in Projector for each, then click OK.

4 Select File > Save and Compact.

If you are adding multiple movies to the package, repeat steps 2 through 4 for each of the movies.

5 Select File > Create Projector.

6 In the Create Projector dialog box, select the movies to include in the projector and click Add.

7 Click Options and select one of the following:

Shockwave (Windows) and click OK.

Standard (Macintosh) and click OK.

8 In the Create Projector dialog box, click Create.

9 In the dialog box that appears, type a name for the projector. If necessary, use the pop-up menu to browse to the desktop folder you created in step 1, and then click Save.

10 Exit Director and return to your computer desktop.

11 Open the folder you created in step 1. Create a subfolder within this folder and name it Xtras.

12 In your Director application folder, copy the Xtra extensions required to play your movie into the Xtras folder you just created.

You must also include external movies, external casts, and linked media with your projector. If the external files are in the folder that contains the projector, the projector can automatically link to the files.

13 (Windows only) In your Director application folder, copy the files dirapi.dll, iml32.dll, proj.dll, and msvcrt.dll into the Xtras folder.

14 Start your projector to see it open quickly and play your movies.

Excluding Xtra extensions from projectors

Sometimes it is useful to place any Xtra extensions needed by a projector in an external Xtra extensions directory. Some reasons for doing this include:

- Xtra extensions stored in a projector must be uncompressed to a separate file each time the projector starts up, which makes startup slower. If the Xtra extensions are stored in an external directory, the projector starts up more quickly.
- Developers are able to easily track exactly what Xtra extensions versions are provided for the projector.
- Developers can easily update an Xtra extension without having to create a new projector.
- Multiple projectors in the same folder can use the same external Xtra extensions folder.
- Sometimes you need to include other external files with a projector, for example an animated GIF. These files can just be put in the external Xtra extensions folder instead of having a separate location.

- Shockwave projectors can use the Xtra extensions that have been already installed with Shockwave instead of requiring their own copy.
- Because of the way the Xtra extensions list for the movie is maintained, it is possible to get Xtra extensions included in a projector that you don't need. For example, let's say that you can use an animated GIF member in a movie. In this case, the AnimGif Xtra is added to the Xtra list and marked for downloading. Later you delete the animated GIF member. The AnimGif Xtra is not removed from the Xtra list even though the movie is no longer using it. The reason the Xtra is not removed from the list is because the movie has no way of knowing whether an animated GIF member is loaded using Lingo or JavaScript syntax (either directly, or by branching to another movie that uses it). So developers have to remember to check the movie's Xtra extensions list to find out exactly what is included in a projector. Having the Xtra extensions always external eliminates this issue.

About cross-platform projectors

Cross-platform projectors are projectors created on one platform or operating system which can run on another: in Director's case, you can create projectors that will play on the Macintosh OS X system, but are created or authored on a Windows system and vice versa.

Note: You cannot create Macintosh Classic projectors on a Windows system.

To create a cross-platform projector:

1 Select File > Publish Settings.

2 From the Formats tab, select Macintosh Projector if you are working on a Windows system, or select Windows Projector if you are working on a Macintosh.

3 Make any other selections you want from the Publish Settings dialog box.

Note: You cannot create custom application icons for a cross platform projector

4 Click Publish.

Custom Xtra extensions and cross-platform projectors

To support custom Xtra extensions in cross-platform projectors, consider the following issues:

- Ensure that there is an entry for your Xtra extension in the xtrainfo.txt file that gives the proper filename for the Xtra extension on each platform. If this information is not provided, the code assumes your Xtra extension only exists for the current platform.
- To use your custom Windows Xtra extension in a Macintosh-created projector, put the Xtra extension in the /Cross Platform Resources/Windows/Xtras/ folder of the Macintosh Director installation.
- Macintosh Xtras must have both a data fork and a resource fork. These two fork files aren't supported on Windows. For each Macintosh Xtra extension, two files are created: one for the data fork and one for the resource fork. The names for these files must be the same as the original Xtra extension name with a suffix: ".data" for the data fork and ".rsrc" for the resource fork. Both of these files are placed in the /Cross Platform Resources/Macintosh/Xtras/ folder of the Windows Director installation.

  For example, the Macintosh Xtra called "MyXtra" would have two files on Windows, "MyXtra.data" and "MyXtra.rsrc."

Processing movies with Update Movies

You can use the Update Movies command on the Xtras menu to do the following:

- Update movies and casts from older version of Director to the latest file format.
- Compress movies for faster downloading from the Internet.
- Remove redundant and fragmented data in movie and cast files. The Save and Compact and Save As commands do this as well.
- Prevent users from opening movie and cast files.
- Batch-process movie and cast files in large projects.

When beginning a project, use Update Movies to convert Director 5 or later version files to the latest file format.

At the end of a project, use Update Movies to compress all your movies and casts at once.

To update and compress movies and casts:

1 Select Xtras > Update Movies.

The Update Movies dialog box appears.

2 Select one of the Action options:

Update converts movies from Director 5 or later versions to the latest file format. As it updates movies, Director consolidates and removes fragmented data, just as when you use Save As. (To update movies from older versions, you must first convert them to the Director 5 file format.)

Protect removes all the data required to edit the movie, but it does not compress the movies further. It adds the DXR extension to movies, and CXT to casts. Protect also flags the movie so it can't be opened in the authoring environment.

Convert to Shockwave Movie(s) rewrites movies and casts in the compressed Shockwave file format and adds the DCR extension to movies and CCT to casts. This options also prevents users from opening the movie or cast and making changes. Once a movie is compressed, there is no way to it to recover an editable file, so be sure to keep the original movie.

3 Select one of the Original Files options:

Back Up into Folder specifies that the original files go in a selected folder. Click Browse to select the folder for the original files. To avoid overwriting old backups, you should select a new folder each time you run Update Movies.

Delete specifies that the newly updated files overwrite the original files. Be very careful when using this option. Once a file is protected or compressed, you cannot open it again in Director.

4 Click OK.

A dialog box appears from which you select the files to change.

5 Select the movies and casts you want to change and click one of the following:

Add to add the selected files.

Add All to add all the movies in the current folder. The items you select appear in the file list at the bottom of the dialog box. You can update movies in different folders at the same time.

Add All Includes Folders before you click the Add All button to include any movies or casts inside folders appearing in the upper list. This option is useful for updating large projects with several levels of folders.

6 Click Update.

Director saves new versions of the selected movies with the same names and locations as the original movies. This ensures that all links and references to other files continue to work properly. Director copies the original movies to the folder you specified, re-creating their original folder structure. If you didn't specify a folder for the original movies, Director prompts you to select one.

Director adds the DCR extension to Shockwave content and the CCT extension to external casts in the Shockwave format. Protected movies have the DXR extension, and protected casts have the CXT extension.

Exporting digital video and frame-by-frame bitmaps

You can export all or part of a movie as a digital video. You can use this digital video in other applications or import it back into Director. Any interactivity in the movie is lost when it is exported as a digital video. You can also export a movie or a part of a movie as a series of bitmaps: BMP in Windows, and PICT on the Macintosh.

You can export QuickTime digital video from either the Windows or the Macintosh version of Director. QuickTime must be installed on the system to export as QuickTime (version 4 or later is required for Windows; version 3 or later is required for Macintosh). You can export the AVI (Audio-Video Interleaved) format only using the Windows version of Director. When you export to AVI, all sounds are lost.

When Director exports animation as a video or bitmaps, it takes snapshots of the Stage moment by moment and turns each snapshot into a frame in the video or a bitmap file. Sprites animated solely by Lingo or JavaScript syntax are not exported.

When Director exports video or bitmaps, it always uses the entire Stage.

To export to digital video or bitmaps:

1 Select File > Export.

The Export dialog box appears.

2 Select the range of frames you want from the Export options at the top of the dialog box:

Current Frame exports the current frame on the Stage. This is the default.

Selected Frames exports the selected frames in the Score.

All Frames exports all frames.

Frame Range exports only the range of frames that begin and end with the frame numbers you enter in the Begin and End boxes.

3 If you select Selected Frames, All Frames, or Frame Range as the Export option, select one of the following options. These options do not work with digital video.

Every Frame exports all frames in the selected range.

One in Every _ Frames exports only the frames at the interval you specify in the box.

Frames with Markers exports frames with markers set in the Score window.

Frames with Artwork Changes in Channel exports frames only when a cast member changes in the channel you specify in the box.

4. From the Format pop-up menu at the bottom of the dialog box, select a format.
   - Windows: Video for Windows (.AVI), DIB File Sequence (.BMP), or QuickTime Movie (.MOV)
   - Macintosh: PICT or QuickTime Movie

   BMP is the standard format for a Windows bitmap series. PICT is the standard format for a Macintosh bitmap file format.
5. If you are exporting video, click the Options button.

   The Video for Windows or QuickTime Options dialog box appears.
6. Select the options you want to use and then click OK.

   For AVI movies, enter a number of frames per second for Frame Rate.

   For information about the QuickTime options, see "Setting QuickTime export options" on page 466.

   The Export dialog box reappears when you click OK.
7. Click Export.

   A dialog box appears, prompting you to save the movie.
8. Name the file and then click Save.

   When you click Export, a dialog box appears allowing you to name the file. If you are saving in video format, only one file will be created. If you are saving in BMP or PICT format, 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 Myfile0001.

Setting QuickTime export options

You use the QuickTime Options dialog box to specify options for exporting a movie as a QuickTime digital video. This dialog box appears when you click the Options button in the Export dialog box and QuickTime is the specified format.

To set QuickTime export options:

1. Select File > Export.
2. Select QuickTime Movie from the Format pop-up menu.
3. Click Options.
4. To set the speed the video will play, select a Frame Rate option:

   Tempo Settings exports the settings in the tempo channel to the QuickTime movie. This setting lets you create a QuickTime movie at any tempo, even if Director is not capable of playing the movie at that tempo in real time.

   The size of an exported QuickTime movie is influenced by the tempo settings, transitions, and palette transitions in the Director movie. Fast tempos, certain transitions, and palette transitions all increase the size of the QuickTime movie. The tempo settings determine the number of QuickTime frames per second and the number of frames per transition. The faster the tempo, the more frames per second.

A movie that would work well with Tempo Settings as the Frame Rate option is one in which the tempos have been carefully timed. For instance, some frames could be set to a tempo of 10 frames per second, and their QuickTime frame durations would be exactly one-tenth of a second. Other frames later in the movie could be set to a tempo of 1 frame per second; when the movie is exported, these slower frames would each last precisely 1 second in the QuickTime movie.

Real Time lets you export a QuickTime movie that matches the performance of the Director movie as it plays on your system. (You should always play the entire movie with script disabled before using this feature.)

When you export a movie with Real Time selected, each Director frame becomes a QuickTime frame. Each frame in the QuickTime movie will match the duration of the same frame in the Director movie.

Director will generate as many frames as required to duplicate each transition, up to 30 frames per second. To increase the number of frames created for any transition, reduce the smoothness of the transition.

This option causes Director to use the actual durations that were stored the last time you played the entire movie, regardless of the actual tempo settings of the movie.

5 To reduce the file size of a QuickTime movie at the expense of quality, select an option from the Compressor pop-up menu. Different options appear on the Compressor pop-up menu depending on the video hardware and software available in your system. Consult your QuickTime documentation.

Animation compression is for simple animations.

Cinepak compresses 16-bit and 24-bit video for playback from CD-ROMs.

Component Video is usually used when capturing from a live video feed.

Graphics compression is for exporting single frames of computer graphics.

None exports with no compression.

Photo-JPEG compression is good for scanned or digitized continuous-tone still images.

Video compression is for exporting video clips.

6 To determine the compression quality and resulting file size when using the chosen compressor, use the Quality slider. A higher-quality setting preserves the appearance of the images and motion but increases the size of the file. A lower-quality setting results in poorer image quality but decreases the size of the file.

7 To determine the color depth (the number of colors) of your artwork, select a setting from the Color Depth pop-up menu. The compression method you select determines the color depth options available to you in this pop-up menu.

8 To determine the method by which the exported QuickTime movie is resized, select values for Scale. You can select a percentage from the Scale pop-up menu, or you can type pixel dimensions in the fields. By entering the number of pixels, you can stretch a movie so that it plays in a rectangle that does not adhere to the original aspect ratio.

9 To choose which soundtracks are exported with your movie, select Channel 1 or Channel 2. A checked box indicates that the associated sound channel in the score is exported with your QuickTime file.

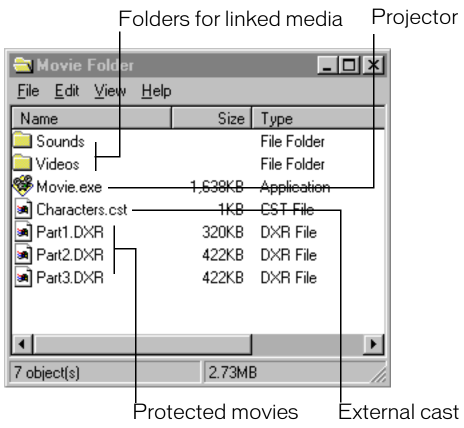
External sounds (sounds you imported as linked cast members) are not exported when you export a digital video. To include sound when you export a digital video movie, you must import the sounds as cast members instead of linking to them.

Looped sounds don't loop in a movie that you have exported as a digital video. To loop a sound in a movie that you plan to export as a digital video, you must trigger the sound by alternating it between the two sound channels.

About organizing movie files

In most cases, you should divide a large production into a series of smaller movies. You can combine as many movies as you want in a projector, but larger files take longer to save and are cumbersome to work with. Also, movies are easier to edit if they are organized in discrete sections.

The best way to organize a large production is to create a small projector file that starts the movie and then branches to Shockwave content or protected content. This saves you the trouble of re-creating the projector every time you change one part of a movie.



Name	Size	Type
Sounds		File Folder
Videos		File Folder
Movie.exe	1,638KB	Application
Characters.cst	1KB	CST File
Part1.DXR	320KB	DXR File
Part2.DXR	422KB	DXR File
Part3.DXR	422KB	DXR File

7 object(s) 2.73MB

A typical file organization for a distributed movie

This approach also makes sense for movies on the Internet, but for different reasons. If the first movie is small, users don't have to wait as long for something to happen. Branching to a series of smaller movies also enables users to avoid downloading time for parts of the movie they do not use.

The size of your movie may be less of an issue if you use streaming Shockwave content. For more information, see "Setting movie playback options" on page 471.

CHAPTER 24
Using Shockwave Player

Macromedia Director MX 2004 movies can use the Internet in various ways: hosting multiuser sessions such as chats and games, streaming movies and sounds, retrieving data from the network, and interacting with a browser. Whether it is distributed on disk or downloaded from the Internet, a movie can use an active network connection to retrieve linked files, send information, open web pages, and perform many other network activities.

To make a movie appear in a user's browser, you can save it as Macromedia Shockwave content and embed it in an HTML document. The movie can play from a local disk or an Internet server. When the user opens the HTML document stored on an Internet server, the movie often begins streaming to the user's system, and begins playing when either a specified number of frames has been downloaded, or waits until all of the movie is downloaded to the local disk. (The movie begins streaming if you have set the Shockwave streaming playback option to ON. If the default streaming option is left at OFF, the movie begins playing after the entire movie has been downloaded to the local drive. For more information, see "Setting movie playback options" on page 471.)

You can also distribute a movie over the Internet as a projector—a packaged movie that the user downloads and executes. A projector plays as a stand-alone application, not in a browser. For more information, see "About distribution formats" on page 451.

When you author a movie, consider how the movie is to be distributed and played on users' systems. If the movie streams from an Internet source, you might need to modify the movie for the best streaming performance and to use the behaviors that are built in to Director to make the movie wait while certain cast members download. Controls and script methods offer ways to send and retrieve media and other information, interacting with a browser, and monitoring downloading.

About streaming movies

When you distribute a movie on the Internet, streaming provides an immediate and satisfying experience for your users. If you don't enable streaming, your user must wait for the entire movie to download before it begins to play. A streaming movie begins playing as soon as a specified amount of content reaches the user's system. As the movie plays, the remaining content downloads in the background and appears when it is needed. Streaming can dramatically decrease the perceived downloading time.

When Director streams a movie over the Internet, it first downloads the Score data and other nonmedia information such as scripts and the size of each cast member's bounding rectangle. This data is usually quite small compared with the size of the movie's media—usually only a few kilobytes. Before starting the movie, Director then downloads the internal and linked cast members that are required for the first frame of the movie (or more frames if you have increased the number in the Movie Playback dialog box). After the movie starts, Director continues to download cast members (along with any associated linked media) in the background, in the order the cast members appear in the Score.

If the movie jumps ahead in the Score or uses cast members that are referenced only by scripts, the required cast member might not be available when necessary. If cast members are not available, the movie either ignores them or displays a placeholder, depending on how you set the streaming options in the Movie Playback Properties dialog box. In addition, if cast members are not fully downloaded and present in memory, you might get script errors, especially if you are working with 3D.

A challenge of authoring for Internet streaming is ensuring that all cast members have been downloaded by the time the movie needs them. To avoid missing cast members, make sure that all the cast members required for a particular scene have been downloaded before beginning the scene. You can use the Director behaviors to wait for media in certain frames or for particular cast members. For more information, see "About streaming with the Score and behaviors" on page 473. You can also write custom scripts to do this. See "Checking whether media elements are loaded with Lingo or JavaScript syntax" on page 474.

Director movies stream automatically if the Streaming playback option is set to ON (its default setting is OFF). In addition to turning streaming off and on, you can specify that the media elements for a certain number of frames must finish downloading before the movie starts playing.

You control streaming movies by arranging sprites in the Score and controlling the movement of the playhead either with the Director behaviors or with Lingo or JavaScript syntax. You can also use these scripting methods to specify when externally linked files are downloaded.

About network operations

Director lets a network operation begin even if a previous network operation is not complete. This capability, often referred to as background loading, lets Director perform multiple operations while loading files. Because something else is happening while files are loading, the user does not perceive the wait.

Note: Loading data from a network is different from loading cast members in Director. Loading from a network loads data to the local disk. Loading cast members in Director means loading cast members into memory.

It is a good idea to author Shockwave content so that it performs other tasks while data is loading in the background. Because Internet operations require background loading, Lingo or JavaScript intended to execute on the Internet behaves differently than script methods that run within one movie. For more information, see "Using Lingo or JavaScript syntax in different Internet environments" on page 477.

Setting movie playback options

To change basic streaming settings for a movie, you use the Movie Playback Properties dialog box. You can turn streaming on and off, specify a number of frames to download before playing the movie, and make Director display placeholders if cast members are not downloaded yet. The Movie Playback Properties dialog box also includes options for locking the current tempo.

Leaving streaming off (the default setting) makes sense for some types of movies. For example, a game that requires all cast members to be available at once might not be suitable for streaming. Other movies work best if the media for a certain number of frames downloads before the movie begins playing. This option is especially useful for streaming movies that were not originally designed for streaming.

Placeholders are rectangles that appear in place of media elements for cast members that have not yet been downloaded. Placeholders are useful when testing to indicate places where media is missing.

You can specify streaming options any time before saving a movie as Shockwave content.

Note: If you want to test how a movie streams from a server before you save the movie as a Shockwave movie, use File > Save and Compact to make sure the data in the movie is properly ordered and that redundant data is removed.

To set movie playback options:

1. Select Modify > Movie > Playback to define streaming options.
2. To let the movie stream automatically, select Play While Downloading Movie.
3. To make the movie wait for all media elements (internal and linked) for a specified range of frames, enter the number of frames in Download __ Frames Before Playing.

   By default, movies download the first frame only. Adjust this setting to the number of frames that is best for your movie.
4. To make the movie display placeholders for media elements that have not been downloaded, select Show Placeholders.

   The placeholders appear as rectangles when the movie plays.
5. To lock the movie to its current tempo settings, select Lock Frame Durations. For more information, see "Locking frame durations" on page 159.

To set Shockwave playback options, see the next section.

Setting Shockwave playback options

To view Shockwave content, your users must have the Macromedia Shockwave Player, which comes preinstalled on many computer systems. The player is also available for free downloading from the Macromedia website at www.macromedia.com/shockwave/download/.

The Shockwave Player includes a volume control and a standard context menu that appears when a user right-clicks (Windows) or Control-clicks (Macintosh) a movie. You can select specific playback options to include for your users when you save your movie as Shockwave content.

Shockwave content loops by default. To cause Shockwave content to play only once, add the `Hold on Current Frame` behavior to the last frame of the movie.

To set Shockwave playback options:

1 Select File > Publish Settings.

  The Publish Settings dialog box appears.

2 If not already selected, select Shockwave File (DCR) on the Formats tab.

  The Shockwave tab appears in the Publish Settings dialog box.

3 Select the Shockwave tab and select the options you want to provide for your users, as described in the following list:

  - Image Compression optimizes your content for download over the Internet by letting you manage the file size vs. the image quality a file. The default setting for all bitmap member types when creating a dcr. The default is set to JPEG, with a quality setting of 80. To further optimize the size of your images, you can adjust the Image Compression setting of each individual bitmap.

    Standard: Select Standard to apply compression techniques used by Director in versions 4 through 7, select Standard. This setting is suitable for simple graphics that contain few colors or non-photographic type images.

    JPEG: Select JPEG and specify the image quality setting by moving the slider to a value between 0 and 100 percent. A lower percentage results in increased compression, but produces a lower quality image.

  - Audio Compression:

    Compression Enabled lets you compress the sound in your movie. Select this option and select the level of compression from the kBits/second pop-up menu. For more information about sound compression, see "Compressing internal sounds with Shockwave Audio" on page 237.

    Convert Stereo to Mono lets you convert stereo audio to monaural. This option is only available if Shockwave Audio Compression is enabled.

  - **Include Cast Member Comments** lets you include comments you might have entered in the Comments text box of the Property inspector for your cast members. You can then use Lingo or JavaScript syntax to access the comments in the DCR file.

  - Enabled context menu items:

    Volume Control lets users adjust the volume of the movie's soundtrack.

    Transport Control provides controls for rewinding, stopping, and stepping through the movie.

About creating multiuser applications

If you want to create multiuser movies or applications with Director, you need to use Macromedia Flash Communication Server MX in your Director movie. This can be a Flash movie that has already been authored to communicate with the Flash Communication server or a simple Flash movie that can serve as a container for Flash script objects that you create with Lingo or JavaScript syntax. You can create applications that use the Flash Communications Server entirely in Director by using Lingo or JavaScript to create Flash script objects.

For more information about using Flash Communication Server MX in Director, see Chapter 9, "Using Flash, Flash Components, and Other Interactive Media Types," on page 181.

About streaming with the Score and behaviors

The easiest way to create a movie that streams well is to arrange the Score properly and use behaviors to control the playhead. Director downloads cast members in the order in which they appear in the Score. Try to arrange the Score so that events don't make the playhead jump far ahead in the Score, where cast members have not yet been downloaded. For example, if you place a menu in the first frame of a movie and a user selects an option that sends the playhead to frame 400, the cast members for frame 400 may not be available right away.

To avoid this problem, begin a movie with a simple introductory scene that contains a few small cast members, preferably vector shapes. You can use a streaming behavior from the Library palette to make the introduction loop until the cast members that are required for the next scene have been downloaded in the background.

Several behaviors that are included with Director control the playhead or a progress bar while media elements are downloading. These behaviors make it easy to allow enough time for downloading to catch up with action in the Score.

Looping behaviors

Looping behaviors make the playhead return (loop) to a frame or stay on the current frame until specified media elements have been downloaded and then continue to the next frame. Attach a looping behavior to a frame in the script channel, not to a sprite. The following looping behaviors are accessible by selecting Internet > Streaming from the Library List menu in the Library palette:

Loop Until Next Frame is Available loops the playhead to a specified frame until all the media elements that are required for the next frame have been downloaded.

Loop Until Member is Available loops the playhead to a specified frame until a certain cast member has been downloaded.

Loop Until Media in Marker is Available loops the playhead to a specified frame until all the media elements for the frame at the specified marker have been downloaded.

Loop Until Media in Frame is Available loops the playhead to a specified frame until all the media elements that are required for a certain frame have been downloaded.

Jumping behaviors

Jumping behaviors make the playhead skip to a specified frame or marker after certain media elements have been downloaded. Attach a jumping behavior to a frame in the script channel, not to a sprite.

Jump When Member is Available moves the playhead to the specified frame after a certain cast member has been downloaded.

Jump When Media in Frame is Available moves the playhead to the specified frame after the media elements for a particular frame have been downloaded.

Jump When Media in Marker is Available moves the playhead to the specified frame after the media elements for the frame at a particular marker have been downloaded.

Checking whether media elements are loaded with Lingo or JavaScript syntax

Director has several options that let an initial portion of a movie start playing as soon as the required data and cast members are available. You can use Lingo or JavaScript syntax to check whether media elements have been downloaded from a network by testing the following:

- Whether a specific cast member is loaded before the movie proceeds
- Whether the cast members used in a specific frame are loaded before the frame plays

Checking whether a cast member or sprite is loaded

To determine whether a specified cast member is available locally, you use the `mediaReady()` cast member or sprite property. You can check for a specific cast member or the cast member that is assigned to a specific sprite. When `mediaReady()` returns `TRUE`, the cast member is present on the local drive and ready for loading into memory. For more information, see the Scripting Reference topics in the Director Help Panel.

This property always returns `TRUE` for local files. It is useful only for movies that stream from a remote server. Because playback can begin before the entire movie has been downloaded, you must make sure that necessary media elements have been downloaded as the movie plays.

Checking whether a frame's contents are loaded

Use the `frameReady()` method to determine whether all the media elements that the specified frame requires are available locally. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

Downloading files from the Internet with Lingo or JavaScript syntax

Lingo or JavaScript syntax uses the Internet's resources by obtaining files from the Internet. The data is copied to the local disk or cache. After data is available on the local computer, use these scripts to retrieve the data for the movie. For more information, see "Retrieving network operation results with Lingo or JavaScript syntax" on page 476.

For a movie or projector that plays outside a browser, background loading is not required. However, preloading is a good idea because it improves playback performance.

All network Lingo or JavaScript syntax operations that obtain data from the network begin downloading the data and return a network ID. The data is not immediately available.

An unlimited number of network Lingo or JavaScript syntax operations can take place at once. However, depending on the browser chosen and the preferences set, there may be limitations set on the number of network operations taking place. When multiple network Lingo or JavaScript syntax operations run simultaneously, rely on the network ID that the method returns to distinguish which operation is complete. Be aware that running more than four operations at once usually adversely impacts performance.

When using network Lingo or JavaScript syntax, the current handler must finish before an operation's result can return. For best results, place script that initiates a network operation and script that uses the operation's result in different handlers. An `on exitFrame` handler is a good location for checking whether an operation is complete.

Note: You should typically use frame events rather than repeat loops for this process.

To execute a network script operation:

1 Start the operation.

   For example, the following statement initiates a text downloading operation and assigns the network ID returned by the `getNetText()` operation to the variable `theNetID`:

   ```
   theNetID = getNetText("http://www.thenews.com")
   ```

2 Make sure that the operation finishes.

   To check an operation's status regularly until the method indicates that the operation is complete, use the `netDone()` method. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

   For example, the following statement loops in the current frame until the download operation is complete:

   ```
   if not(netDone(theNetID)) then _movie.go(_movie.frame)
   ```

3 Check whether the operation was successful by using the `netError()` method. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

4 Obtain the results if the operation is complete.

Note: You can also use getStreamStatus() to determine the status of a network operation. You can use `netDone()` to check whether an operation is "done" or "not done". By using getStreamStatus(), you can check on an operation's progress, as it displays how many total bytes there are and how many have been downloaded so far.

To cancel a network operation in progress:

- Use the `netAbort()` method to cancel a network operation without waiting for a result. This frees up capacity for Internet access, which lets other network operations finish faster. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

To retrieve a file as text:

1 Use the `getNetText()` method or the `postNetText` method to start retrieving text. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

2 Use the `netTextResult()` method to return the text you retrieved with `getNetText` or `postNetText`. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

To retrieve and play a new Shockwave movie from the network:

- Use the `gotoNetMovie()` method. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

The current movie continues to run until the new movie is ready to play. After the new movie is ready, the player quits the current movie and plays the new movie in the same display area as the calling movie.

To open a URL in the user's browser:

- Use the `gotoNetPage()` method. This method works whether the URL refers to Shockwave content, HTML, or another MIME type. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

You can specify that this method replaces a page's content or opens a new page. If the browser is not open, the method launches the browser. If the `gotoNetPage` method replaces the page in which the movie is playing, the movie keeps playing until the browser replaces the page.

The `gotoNetPage` method is similar to the Director `open` method. It does not return a value.

To preload a file from the server into the browser's cache:

- Use the `preloadNetThing()` method. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

The `preloadNetThing()` method initiates downloading a linked movie asset into the cache, where it is available for later use. Director can later preload the asset into memory without a download delay.

The current movie continues playing while preloading occurs.

To test whether getNetText(), preloadNetThing, or gotoNetMovie operations are complete:

- Use the `netDone()` method.
- Use the `getStreamStatus()` method.

For more information about these methods, see the Scripting Reference topics in the Director Help Panel.

To post information using HTTP post/get abilities to a server and retrieve a response:

- Use the `getNetText()` or `postNetText()` methods. You can use either method to post data to a server, but HTTP `"get"` commands can only post a limited amount of information compared with HTTP `"post"` operations. If you are posting a great deal of information, the `postNetText()` method is recommended.

For more information about these methods, see the Scripting Reference topics in the Director Help Panel.

Retrieving network operation results with Lingo or JavaScript syntax

Lingo or JavaScript syntax can retrieve network operation results, such as a text result, a unique identifier for a network operation, a file's MIME type, and the date an HTTP item was last modified.

To retrieve the text result of a network operation:

- Use the `netTextResult()` method. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

To retrieve the "date last modified" string from the HTTP header for a specific item:

- Use the `netLastModDate()` method. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

To obtain the MIME type of the HTTP item:

- Use the `netMIME()` method. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

The results of the `netTextResult()`, `netDone()`, `netError()`, `netMIME()`, and `netLastModDate()` methods are retained until normal memory clean-up is needed.

To determine the state of a network operation that retrieves data:

- Use the `Lingo on streamStatus` event handler or the `JavaScript streamStatus` function. For more information, see the Scripting Reference topics in the Director Help Panel.

Using Lingo or JavaScript syntax in different Internet environments

Some Lingo or JavaScript syntax features behave differently, depending on whether the movie is playing back in a browser, as a projector, or within the authoring environment.

Using Lingo or JavaScript syntax with Internet security restrictions

Because of security issues for movies that play back in browsers, the following Lingo or JavaScript syntax features are unsupported for Shockwave movies playing in a browser.

In general, the following Lingo or JavaScript syntax features are unsupported because of Internet security concerns:

- Setting `colorDepth()` for the user's monitor
- Saving a movie by using the `saveMovie()` method
- Printing by using the `printFrom()` method
- Opening an application by using the `open()` method
- Stopping an application or the user's computer by using the `quit()`, `restart()`, or `shutDown()` method
- Opening a local file that is not in the dswmedia folder or a subfolder of the dswmedia folder
- Pasting content from the Clipboard by using the `pasteClipBoardInto()` or the `copyToClipboard` method prompts a security dialog asking for permission to perform the operation.
- Searching for files on a user's system with `getNthFileNameInFolder()`, `searchCurrentFolder()`, or `searchPath()`.
- The permanent downloading of files from a URL to the user's local drive using `downloadNetThing()` is disabled in Shockwave due to security concerns. This function works in authoring and projectors.

Using URLs with Lingo or JavaScript syntax

In addition to the Lingo or JavaScript syntax that is explicitly intended for use with network operations, some script elements can use URLs as references to external files.

The following Lingo or JavaScript syntax elements can use URLs as file references in all circumstances:

- `moviePath`
- `pathName`
- `unloadMovie`

Note: You can also set the fileName of cast members to URLs and link to external media assets.

The following Lingo or JavaScript syntax supports URLs as references to external files. If you use this script in projectors or during authoring, you can avoid pauses while the file is being downloaded by first using `preloadNetThing()` to download the file. After the file has been downloaded, you can use these Lingo or JavaScript syntax elements with the file's URL without a delay.

When the following Lingo or JavaScript syntax is used in a browser, however, you must first download the file by using the `preloadNetThing` method. If you don't, the script fails.

- Using a `gotoMovie` statement
- Using an `importFileInto()` method
- Using a `preLoadMovie()` method
- Using a `playMovie()` method
- Using an `openWindow` method`()` (disabled in browsers)

The following Lingo or JavaScript syntax elements can use URLs to Shockwave Audio (SWA) sound files as file references:

- `streamName`
- `URL` cast member property

The following elements don't work in Shockwave content because Shockwave does not support movies in windows (MIAWs):

- `open window`
- `forget window`
- `close window`

Differences in scripting for browsers

The following list discusses some general differences in the way to script for a movie that plays over the Internet, depending on whether the movie is in a browser.

- For a movie playing in a browser, it's best to use `preloadNetThing` to load media elements, external casts, or other movies you might navigate to into the browser's cache first. If the media elements are not preloaded using `preloadNetThing`, linked media elements might not be present when they are needed.
- Avoid using long repeat loops in browsers; such repeat loops can make the computer appear unresponsive. As an alternative, you can split long operations into sections and execute them over a series of frames or check for user actions in an `on exitFrame` handler. Also remember that the status of any active network operation does not update while long repeat loops are running.
- Do not use a `repeat while` loop to check whether a network operation is complete.

Lingo or JavaScript syntax that is unsupported in browsers

The following Lingo or JavaScript syntax features are unsupported for movies that play back in browsers:

- Creating and managing a MIAW
- Installing and managing custom menus

Interacting with browsers

You can use `setPref()` or `getPref()` to read or write simple text *preference* files to the Prefs folder found in the Shockwave Player folder.

To write to a preferences file on a local disk:

- Use the `setPref` method. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

  After the method runs, a folder named Prefs is created inside the Shockwave Player folder (in the same location as the Xtras folder). The `setPref` method can write only to that folder. The default folder locations for Windows and Macintosh are described in the following list:

  Windows The \Macromed\Shockwave 8 subfolder of the system folder; the system folder is typically c:\winnt\system32 or c:\windows\system

  Macintosh The OS 9 path is The System Folder:Extensions:Macromedia:Shockwave 8 folder. The OS X path is /my Volume/Users/<me>/Library/Application Support/Macromedia/Shockwave/prefs.

  The `setPref` method cannot write to a file that is on a CD.

To return the content of a file that was written by a previous setPref method:

- Use the `getPref()` method. If no `setPref` method has already written such a file, the `getPref()` method returns `VOID` if called from Lingo or `null` if called from JavaScript syntax. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

To specify text in a browser's status area:

- Use the `netStatus` method. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

  Note: Some browsers do not support this method.

Testing your movie

However you decide to create your movie, test it thoroughly before releasing it to the public. Make sure you test on systems with all common types of Internet connections, especially on slow modems and at busy times of day. The following list describes things you might want to check before distributing your movie over the Internet; remember, however, that each movie has its own special needs:

- Compare a streaming version of the movie to a nonstreaming version to see if the performance is different. Some smaller movies might work better without streaming playback.
- Verify that all linked media elements appear correctly. To see if the movie correctly handles an error, try forcing the linked media elements to fail.
- Run the movie on all systems your users are likely to have. For the general public, these include Windows 98, 2000 or XP. For Macintosh OS X users, these include Power Mac G3 running OS X 10.2.6 or 10.3 or later. For Macintosh Classic users, a Power Macintosh G3 running System 9.2. Browsers include Netscape 7.1 or later; Microsoft Internet Explorer 5.2 or later and a color monitor.
- Test your movie on a machine that was not used to develop the content: this is critical.

- Run the movie on slow and fast connections; problems can arise from fast as well as slow connections.
- Check for display problems on systems set to 8-, 16-, 24-, and 32-bit color. Also test as many types of monitors and display adapters as you can.
- Check for font mapping problems in your movie. If your movie uses nonstandard fonts, use embedded fonts. For more information, see "Embedding fonts in movies" on page 164.
- Check for sound problems, particularly if you stream sounds with SWA.

About downloading speed

Developers distributing multimedia over the Internet usually limit file size, primarily because most users connect at relatively slow speeds. At 28,800 bps, it takes 30 seconds to 1 minute to download a 60K file. Using streaming playback can help you avoid some of the delays caused by downloading large files.

Movies and streaming SWA sounds always compete for control of the network, which can cause a noticeable problem on slower connections.

The following chart shows theoretical throughput times for various types of network access technology. The rates calculated for this chart are based on the average throughput rate shown—actual download times vary, and are highly dependent on the network conditions that exist at the time of download.

Content	28.8 Kbps	56K modem	DSL	T1
Small graphics and animation, 30K	15 sec	7 sec	1 sec or less	1 sec or less
Small complete movie, 100K	45 sec	21 sec	3 sec	1 sec or less
500K movie	220 sec	107 sec	15 sec	4 sec
1 MB movie	--	213 sec	30 sec	8 sec

INDEX