CHAPTER 7

The Power of Behaviors

In This Chapter

Introducing Director's Library Palette window

Accessing prepackaged behaviors

Creating interactive Director movies without scripting

Using the Behavior Inspector to attach, create, and edit behaviors

Attaching external casts to use third-party behaviors

Sending messages from one behavior to another

Interactivity is critical to multimedia — without it, a multimedia application is little more than a self-running slideshow. Director 6.0 introduced a powerful new tool for adding interactivity to your movies: the behavior. Whether encapsulated in the Library Palette, or implemented via Lingo using the Behavior Inspector, behaviors make your buttons click crisply, control media within your movie, and even keep time for you.

What Are Behaviors?

With behaviors, you can add interactivity to a movie without Lingo scripting. Later, in Chapter 11, you learn to create a few simple scripts in Lingo to:

- Jump to a target frame
- Loop on the current frame
- Move the playback head forward one frame
- Move the playback head back one frame
- Jump to a target marker (creating a marker is covered later in this chapter)

When you create these scripts in Chapter 11, you actually create behaviors — because a behavior is nothing more than a script or a page of code that includes *event handlers* (or subroutines) for events that occur in your movie. Director's standard events are listed later in this chapter and are discussed in detail in Chapter 11.

The scripted actions in the bulleted list that opens this chapter are simple examples of what you can accomplish with behaviors. Fortunately, you can add powerful scripts to your movies without programming in Lingo, using the dozens of prescripted behaviors that are shipped with Director. Director's built-in behaviors are contained as a series of external cast libraries (such as interactive.cst, navigation.cst, QuickTime.cst, text.cst, and so on) that are stored in the Libs folder within the Director 8 folder on your hard drive. These prepackaged behaviors enable you to add interactivity to objects in your movie, just by clicking, dragging, and responding to a few simple prompts.

Note

A well-written behavior doesn't require modifications to its code in order to work in your program. That means you can attach behaviors to objects without writing a single line of code.

Accessing Director's predefined behaviors

All the predefined behaviors included with Director appear in the Library Palette window. You can display the Library Palette by choosing Window ➪ Library Palette. The purpose of this palette is to enable you to view and attach predefined behaviors to sprites in your movies. Using the Library Palette window, you can click, drag, and attach behaviors to objects in your movies, but you *cannot* edit or delete the original behaviors. When you drop a behavior from the Library Palette window onto an object, Director copies the behavior to the movie's Cast window. That's where you can view and edit the behavior, as you learn to do later in this chapter and throughout the Lingo sections of the book.

Behaviors are grouped into several categories (Animation, Controls, Internet, Java Behaviors, Media, Navigation, and Text). Each category contains behaviors that are related to the action that the behavior was built to perform. Several categories — media, for example — are divided into subcategories that group the behaviors according to the aspects of the main category with which the behaviors were designed to work. The categories found on the Library Palette and the types of behaviors that are contained in them are:

✦ **Animation.** These behaviors are applied to sprites to perform numerous animation effects and can enable the user to manipulate sprites while the movie is playing. There are three types of animation behaviors: automatic, interactive, and transitions.

  • You use automatic animation behaviors to animate sprites automatically; although they can be triggered by another event, they don't require any user interaction. Some of these behaviors are frame-based — the action happens over a series of frames in the Score. Others are time-based — the action happens over a specified period, regardless of the number of frames the sprite occupies in the Score.

- Interactive animation behaviors require some type of user interaction to trigger them. You use most of the behaviors in this section to manipulate sprites on the Stage while the movie is playing. For example, you can make a sprite "draggable," enabling a user to drag the sprite to another location on the Stage. There are also several behaviors that detect the location of a sprite while it is being moved on the Stage and then determine whether it's within the bounding box (referred to as the *rect*) of another sprite.
- Sprite transition behaviors create transitions from one screen to another, using the new imaging Lingo feature introduced in Director 8. Many of these effects mimic ones found in programs such as Adobe Premiere, such as barn doors and soft-edge wipes.

Cross-Reference

Chapter 26 goes into great detail about the new imaging Lingo commands introduced in Director 8.

✦ **Controls.** Controls contain a series of behaviors designed to add interactivity and user feedback to your movies. Most of the basic behaviors that are applied to buttons are found in this category. Also, several behaviors enable you to create a status bar and ToolTips to display custom messages. Last, but not least, one behavior is for building an analog clock that uses the internal clock on the user's computer to correctly display the time.

✦ **Internet.** The Internet category (you guessed it) contains behaviors designed for building movies that primarily will be played over the Web. Three subcategories of behaviors are located here: Forms, Multi-user, and Streaming. The first two subcategories contain behaviors that are designed to enable your movie to interact with a Web server. The Streaming category contains behaviors that enable you to control the way Shockwave movies will download and play back over the Web.

- The Forms subcategory contains behaviors for creating forms in your movies that the user can fill out. The data can then be sent to a Web server and used by other applications, such as a database. These behaviors are for advanced users and require a working knowledge of Web server applications.
- The Multi-user subcategory contains behaviors for connecting to and disconnecting from a server. Also, several behaviors in the subcategory are for creating Shockwave "chat rooms," which enable multiple users to communicate with each other over the Web.
- The Streaming subcategory contains behaviors that control the way the media elements in a Shockwave movie download from the Web. You can use behaviors to display a progress bar showing the status of the movie as it is being downloaded. In addition, several behaviors enable you to loop a movie until the media elements that you defined have been downloaded.

✦ **Java Behaviors.** These are a collection of "Java-safe" behaviors that should be used if you will be outputting your Director movies as Java applets to be played over the Web. The Java behaviors are for advanced users and require a strong knowledge of Lingo.

✦ **Media.** You use the Media behaviors to control and add interactivity to Flash movies, QuickTime digital video, and audio that has been imported into your movies. The subcategories are divided according to the type of media that the behavior is designed to work with: Flash, QuickTime, or Sound.

- You use Flash behaviors to add interactivity and control the playback size and quality of an entire Flash movie or individual Flash elements.
- Use the QuickTime behaviors to create custom controls that will be used to play back a QuickTime movie. This subcategory contains a very useful behavior that you can use to create a custom VCR-like control panel that enables a user to control the way a QuickTime move is played. Another behavior enables you to create a custom slider bar that works the same way as the built-in controller that can be displayed with QuickTime video or audio movies.
- Two Sound behaviors in this subcategory enable you play a linked or internal sound file.

✦ **Navigation.** You use the Navigation behaviors primarily in conjunction with the Control behaviors to determine where the movie will advance to when a user clicks on a sprite that contains a control behavior. Also, other behaviors control how long a movie loops on a particular frame.

New Feature

Director now has a set of behaviors designed to create painting tools that enable users to paint on the screen while the movie is running.

✦ **Paintbox.** This new set of behaviors introduced in Director 8 is designed to create a painting application using Director. These behaviors control the basic functions needed to paint images on the screen, including selecting colors and brushes, erasing, and undoing an action.

✦ **Text.** Use these behaviors to format text as well as to create special effects with text, such as having the text appear to type automatically on the screen. Several behaviors in this category are for generating and displaying hypertext links for multimedia applications that will interact with the Web.

Tip

If you momentarily hold the mouse over any behavior in the Library Palette, a ToolTip appears, giving a brief description of the behavior.

A detailed listing of all of the behaviors that ship with Director, as well as their uses, is in Appendix B, "Director's Built-in Behaviors," located at the back of this book.

You work with several of these behaviors in this chapter. In each case, the action listed in the Action column occurs when a specified event occurs. When you set up the use of a behavior, you get to specify the event that triggers it.

Attaching predefined behaviors to objects

To use a behavior from the Behavior Library requires just a few steps.

1. First, be sure that the Library Palette window is visible by choosing Window ➪ Library Palette. The Library Palette window appears (shown in Figure 7-1) and displays the behaviors stored in a single predefined external cast library stored in the Libs folder.

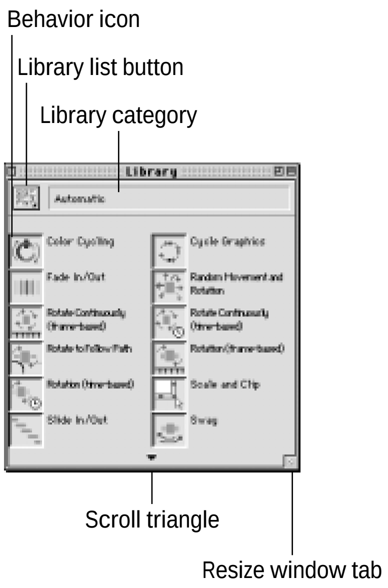


Figure 7-1: All of Director's predefined behaviors are contained in the Library Palette.

2. If the desired behavior is not visible, you can view other behaviors by using the Library List button, which is identified in Figure 7-1. Simply click the button, drag until the desired behavior library name is highlighted (such as Animation, Navigation, Media, and so on), and then release the mouse button. The behaviors contained in the selected category are displayed in the window. If more behaviors exist than can be displayed simultaneously in the Library Palette window, a scroll triangle appears at either the bottom or the top of the palette. You can use the scroll triangle to scroll toward the beginning or end of the list of behaviors in the current behavior library. Resize the entire window by clicking on the tab in the lower right corner and dragging the window to the desired size.

Note

In Director 6, a single Behavior Library .cst file contained all the prepackaged behaviors. Once opened, you had to scroll through cast member after cast member to locate the specific behavior that you wanted. Director 8's Library Palette window organizes behaviors by their function, making it easier to locate the desired behavior.

3. After you have located and displayed the desired behavior in the Library Palette, click the behavior's icon (identified in Figure 7-1) to select it.
4. Drag and drop the behavior by its icon onto a sprite (on the Stage) or onto the target cell (in the Score window's script channel).
5. Some behaviors require parameters, called *properties*, which define the action or designate a target for the behavior's action. For example, you can attach a behavior to a button that causes the playback head to jump to a specific frame. To make the jump, the behavior must be told the name or number of the target frame. If additional information is required, a Parameters dialog box in which you must enter the properties for the behavior appears. After you enter the requested parameters, click OK to close the dialog box.

A behavior can be attached only to a sprite on the Stage or to a frame in the Score window. Behaviors that are dropped onto a sprite are called *sprite scripts*. You can attach multiple behaviors to a single *sprite* on the Stage to create a more complex action. Behaviors dropped onto a cell in the Score window are considered *frame scripts*. You can attach only a single behavior to a frame's script channel.

Note

In their most essential form, behaviors are Score scripts. Score scripts, as well as other types of scripts, are discussed throughout the Lingo section of this book.

Building a Better Movie

Director is the hands-down favorite among graphic designers for creating portfolios, resumes, and promotional pieces. Version 8 of the program is sure to increase its popularity, because it offers so many code-free options for creating highly interactive presentations.

Using the built-in behaviors that come with Director in your movies enables you to add sophisticated interactive and animation components, easily control how a movie downloads over the Internet, and even create chat rooms and multiuser games. You can add all of these components to your multimedia applications with little or no Lingo programming experience.

Pushing all the right buttons

When you get right down to it, creating an interactive interface boils down to creating buttons and then telling the program what to do when someone clicks them. It is these buttons, in fact, that make the interaction between user and application possible. With buttons, the graphical user interface (GUI) becomes more than simply a picture on the screen — it becomes a tactile experience. A good button invites the user to click it and serves to take the program's illusion of reality one notch higher (see Figure 7-2).

Figure 7-2: A well-designed button engages the user.

The exact definition of a button can vary widely, but for the most part includes these characteristics:

- ✦ A *button* is an object or spot on the screen that, when clicked, causes something to happen.
- ✦ A button typically simulates a real-life physical button, like those on an electronic appliance or computer. When you click the mouse on it, the button changes its appears (looks "raised" or "pressed"); when the mouse button is released again, the button returns to its former state (pressed or not pressed). This kind of button essentially transfers the physical action of the mouse click to the virtual action of a button click.
- ✦ A button may become highlighted (become brighter on the screen) or animated when the cursor moves over it, with or without the pressed appearance.
- ✦ A button can be rendered inactive so that it does not react at all to the mouse, and then be made active again when some external event takes place.

Even though buttons are a staple of multimedia, they have always been notoriously difficult to pull off well in Director. Beginning with Director 6.*x* this has changed (and how!). Now you can implement buttons in three different ways: by employing standard system buttons on the Tool Palette, by using prepackaged behaviors (as shown in the next section), or by writing a navigational handler using Lingo.

Cross-Reference Chapter 14 discusses using Lingo to control a button's behavior.

Using navigation behaviors to build buttons

In the next few sections, you use Director's predefined behaviors to accomplish some of the tasks that required Lingo scripting in prior versions of Director. To provide the user control over the flow of a movie, you must be able to put the playback head in a holding pattern on a specific frame and attach behaviors to buttons that, when clicked, forward the playback head to the next frame or a designated frame. Finally, you apply a behavior that sends a message to another sprite to trigger an animation. The Library Palette includes behaviors to accomplish these tasks, which you can drag and drop onto frames and sprites.

To demonstrate this, you use the engine.dir movie for the exercises in this chapter. The engine.dir file is a partially built Director movie that simulates an interactive learning module that could be created as part of a multimedia application. Most of the cast members that will be used in this movie are already on the Stage. You will concentrate on adding behaviors to the sprites that will add the interactivity to the movie.

When you look at the Score window, it is difficult to verify the names of cast members upon which each sprite is based, because almost all sprites in the movie are one frame in duration. To display the name for each sprite, you should make the Score the active window and move the mouse pointer over the sprite in question. If you leave the pointer there momentarily, a ToolTip appears, showing the sprite's name. You can also determine the name and appearance of a sprite by clicking on its cell in the Score window, which causes a thumbnail of the cast member to appear in the upper-left corner of the Score window along with the sprite's name, as shown in Figure 7-3.

Tip You can also assign a color to cells in the Score to make it easier to identify sprites. Simply select the cells in the Score to which you want to apply a color, and then click on a color swatch located in the lower-left corner of the Score window.

In the following exercise, you initially test the movie to determine whether it plays from the first frame to the last without pausing. Next, you open the Library Palette window with prescripted behaviors and add a behavior to the movie that causes it to pause and loop on frame 1.

Note When you apply a behavior to a sprite from the Library Palette, the behavior is copied to the current cast window.

On the CD-ROM For this exercise, you use a partially complete director movie (engine.dir) available on the companion CD-ROM in the EXERCISE:CH07 (EXERCISE\CH07) folder. You will be asked to save this movie to your hard drive at the end of this exercise so that you can add features to it in the other exercises in this chapter.

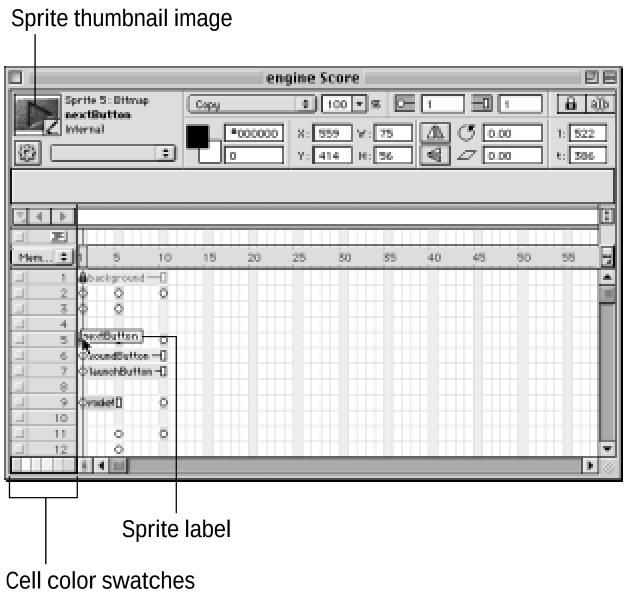
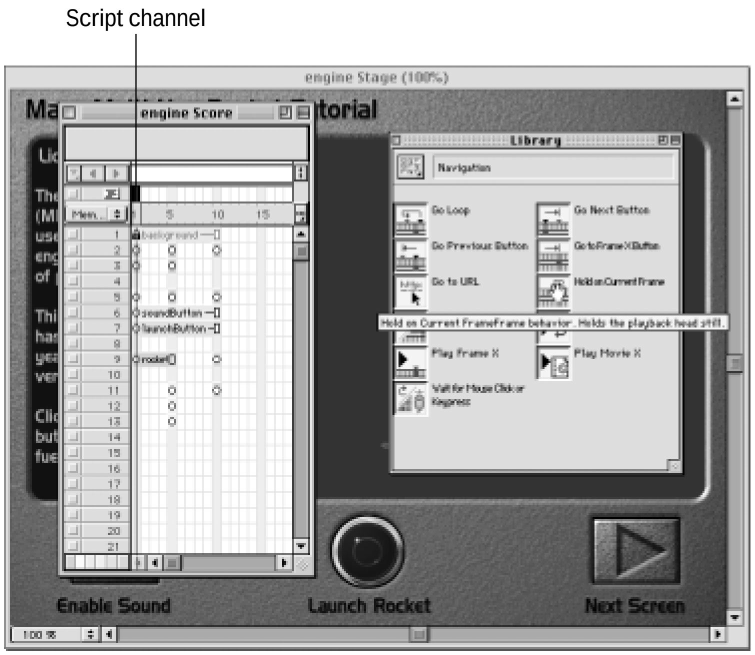


Figure 7-3: The Score window with a thumbnail of a sprite displayed.

Using a Behavior to Loop on the Current Frame

1. Open the engine.dir movie.
2. Be sure that the Score and Stage windows are visible. The Score will appear similar to Figure 7-3.
3. Rewind and play the movie. It plays from start to finish without stopping (and may loop if the Control Panel Loop Playback button is selected. You see certain elements flashing on and off, which happens because some sprites are used only on single frames, while others (like the background sprite) are used in all of the frames of the movie.
4. Stop the movie and rewind it.
5. If the Library Palette window is not visible, click the Library Palette button on the toolbar or choose Window ➪ Library Palette.
6. When the Library Palette window appears, click the Library List button. When the pop-up list appears, select the Navigation category.
7. Locate the Hold on Current Frame behavior in the list of available behaviors, and then drag and drop it onto the Script channel in frame 1 of the Score window, as shown in Figure 7-4.



Hold on Current Frame behavior. Holds the playback head still.

Figure 7-4: Drag the Hold on Current Frame behavior to the script channel in frame 1.

8. Save the movie as **engine1.dir**.

9. Rewind the movie, close all open windows, and then play the movie.

Tip You can temporarily close all windows and play a movie if you hold down the Shift key and press Enter on the numeric keypad.

The engine1.dir movie now holds on the first frame of the movie. You can click the Next Screen, Enable Sound, and Launch Rocket buttons, but nothing happens in your movie . . . yet!

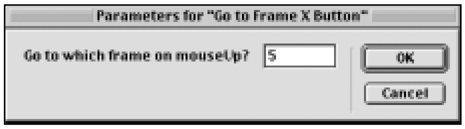
10. Stop the movie and rewind it.

The Hold on Current Frame behavior accomplishes a single task: It causes the playback head to loop on the current frame until another event instructs the playback head to move on. It can be instructive to examine the script that causes this behavior to function. You'll do that later in this chapter.

At this point, the engine1.dir movie is waiting on the first frame for some kind of instruction that will tell it to proceed to the next frames of the movie. In the following steps, you attach the Go to Frame behavior to the Next Screen button and thus enable the playback head to escape from the frame 1 loop.

Attaching a Behavior to Jump to a Specific Frame

1. Be sure that the engine1.dir movie and the Library Palette window are both open.
2. Locate the Go to Frame X behavior in the Library Palette window (it's in Navigation group of behaviors).
3. Drag and drop the Go to Frame X behavior onto the Next Screen button sprite (colored yellow in the Score window) located in sprite channel 5. Because this behavior (unlike the Hold on Current Frame behavior) includes properties, the Parameters dialog box appears, as shown in Figure 7-5. You must specify the destination frame.



Go to which frame on mouseUp? 5

Figure 7-5: Enter the frame number to which you want the movie to advance when a user clicks the button.

4. Enter **5** next to the Go to which frame on mouseUp prompt. Then click the OK button to return to Director's main window.
5. Save the movie as **engine2.dir**, and then rewind and play the movie.
6. Click the Next Screen button to test the Go To Frame behavior. If the Loop Playback button on the Control Panel is turned off (so that the movie does not loop), the movie will jump to frame 5 and continue playing until the end. You need to add another Hold on Current Frame script to have the movie loop on frame 5.
7. To cause the movie to loop on frame 5 rather than automatically terminating, open the Cast window and drag the Hold on Current Frame behavior to the script channel in frame 5. This creates a copy of the behavior in frame 5.

Caution

It's possible to copy and paste a script; however, it's generally not a good idea. If the behavior that you are copying has parameters applied to it, the same parameters will be applied to the new sprite or Score channel onto which it has been pasted. It's usually safer to reapply a behavior that has parameters connected with it to the new sprite so that new parameters can be set for it. It is far safer to drag and drop a behavior from either the Cast window or the Library Palette into the Score window's Script channel.

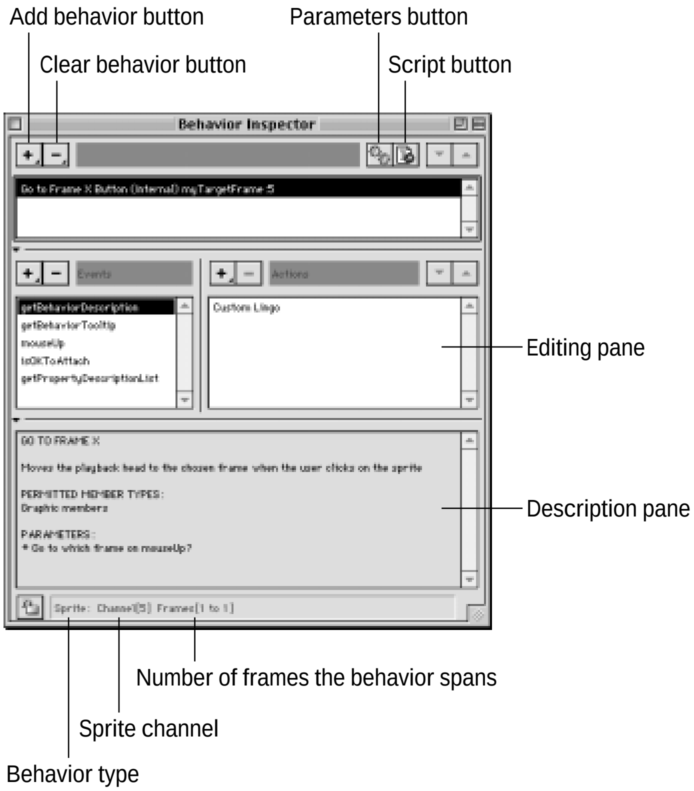
8. Save the movie again as engine2.dir, and then rewind and play the movie. After testing the Next button, stop the movie.

The movie now loops on frame 1 until the Next Screen button is clicked, and then it advances to frame 5 and loops on that frame. The Next Screen button appears to be dimmed out. Because this button is no longer needed, a blend was applied to it to make it appear to be inactive.

Using the Behavior Inspector

The Behavior Inspector, introduced with Director 6, provides a means for you to view and edit existing predefined behaviors and to create new ones in a cast. You cannot modify or delete behaviors in the Library Palette window, but you can view, edit, delete, or create a behavior in the Cast window.

The Behavior Inspector, shown in Figure 7-6, provides easy access to information about the behavior, including the behavior's name, type, its script, its description, the names and values of any parameters (properties) associated with the behavior, and the span of frames to which the Behavior is applied. You open the Behavior Inspector by choosing Window ➪ Inspectors ➪ Behavior Inspector or by pressing Command+Option+; (Ctrl+Alt+;).



Go to Frame X Button (Internal) myTargetFrame 5

getBehaviorDescription
getBehaviorTooltip
mouseUp
isOKToAttach
getPropertyDescriptionList

Custom Lingo

GO TO FRAME X

Moves the playback head to the chosen frame when the user clicks on the sprite

PERMITTED MEMBER TYPES:
Graphic members

PARAMETERS:
* Go to which frame on mouseUp?

Sprite: Channel[5] Frames[1 to 1]

Figure 7-6: The Behavior Inspector with both the editing and description panes expanded.

The Behavior Inspector is built on a series of panes that can be collapsed or expanded as needed. In its default state, these panes are hidden. You click the two triangles on the left side of the Inspector to expand or collapse the panes.

You'll get a good look at these panes as you work through these sections. The Editing Pane contains the Events and Actions pop-up button. The Description Pane displays a description of what the behavior does; this may include information about properties and their roles, events that the behavior generates, and copyright information.

At the top of the Behavior Inspector is the gear-shaped Parameters button, which opens the behavior's Parameters dialog box. From it, you can change the behavior's properties. The Script button, located next to the Parameters button, enables you to view and directly modify the Lingo script that is used for the behavior. At the bottom of the Behavior Inspector, you see the type of behavior, the sprite channel in which it's located, and the span of frames to which the behavior has been applied. A button located in the lower-left corner of the window locks the currently selected behavior so that it is displayed no matter what sprite is chosen. This is useful if you need to view all of the properties that are applied to a series of sprites, although it can cause confusion, because there is no indication of what the displayed behaviors are applied to.

Viewing the script behind a behavior

In Chapter 11, you explore the Script window and begin learning the basics of programming using Lingo, Director's proprietary scripting language. For now, it's helpful to glimpse behind the façade of the behavior and simply be aware that it's just a series of instructions.

By default, when you double-click on a behavior in the Cast window (either an internal or external cast), the Behavior Inspector opens so that you can edit it. You can change the default setting so that the Script window pops up rather than the Behavior Inspector. As you become proficient in Lingo scripting, you may prefer to have the Script window appear instead of the Behavior Inspector — although for now, the Behavior Inspector is more useful as you learn and experiment with codeless programming.

To change which window appears for viewing or editing a behavior, you can choose File ➪ Preferences ➪ Editors. After the Editors Preferences dialog box appears, select Behavior from the list and click the Edit button. When the Select Editor for Behaviors dialog box appears, you can choose between the Script Window radio button or the Behavior Inspector radio button, and then click the OK button. Finally, click the OK button to close the Editors Preferences dialog box. For the duration of this lesson, we'll assume that you have established the Behavior Inspector as the default editor for behaviors.

You can open the Behavior Inspector by using one of several methods:

✦ You can use the shortcut keys Command+Option+; (semicolon) on the Macintosh or Ctrl+Alt+; (semicolon) for Windows.

✦ You can click the Behavior button in the top-left part of the Score window (see Figure 7-7) or on the Director toolbar.

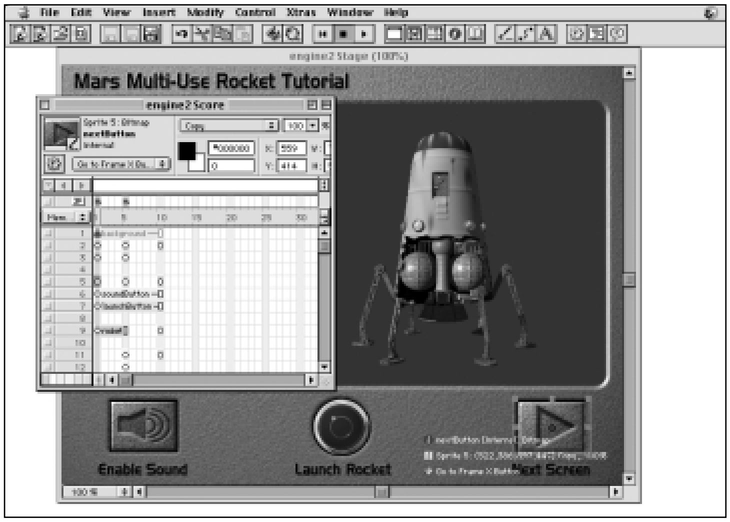
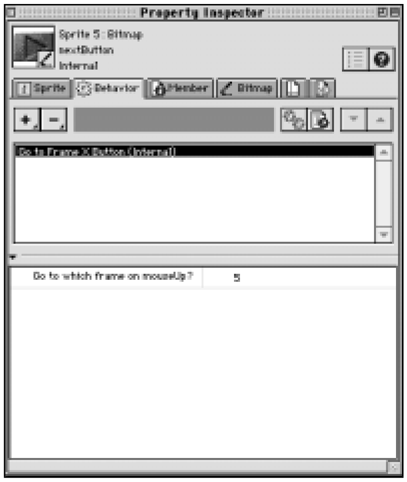

File Edit View Insert Modify Control Xtras Window Help
engine2 Stage (100%)
Mars Multi-Use Rocket Tutorial
engine2 Score
Enable Sound
Launch Rocket
Next Screen

Figure 7-7: Open the Behavior Inspector by clicking on the Behavior icons located in the Score window or on the toolbar.

Director 8 introduces a stripped down version of the Behavior Inspector, shown in Figure 7-8, that is included in the new Property Inspector window. The primary purpose of the Behavior tab is to view and modify the parameters of the behavior that are applied to a sprite. The version contained in the Property Inspector does not contain the description window or the tools needed to create a new behavior. You can open the Behavior tab located in the Property Inspector by using one of two methods.

✦ If the Sprite Overlay is visible, you can click the Behavior icon (identified in Figure 7-7) to display the Behavior tab in the Property Inspector.

✦ Click on the Behavior tab in the Property Inspector when editing other properties of a sprite that has behaviors attached to it.



Go to Frame X Button (Internal)

Go to which frame on mouseUp? 5

Figure 7-8: The Property Inspector contains a "stripped down" version of the Behavior Inspector.

Tip

What's the Sprite Overlay? When you click on a sprite to make it active, the Sprite Overlay appears with three icons: the Cast Member Properties icon, the Sprite Properties icon, and the Behavior Inspector icon. Clicking the Behavior Inspector icon opens the Behavior Inspector. You can toggle the Sprite Overlay between being visible and invisible by choosing View ➪ Sprite Overlay or by using these keystrokes: Ctrl+Shift+Alt+O (Windows) or Command+Shift+Option+O (Mac). This feature provides a summary of information on the currently selected sprite.

Caution

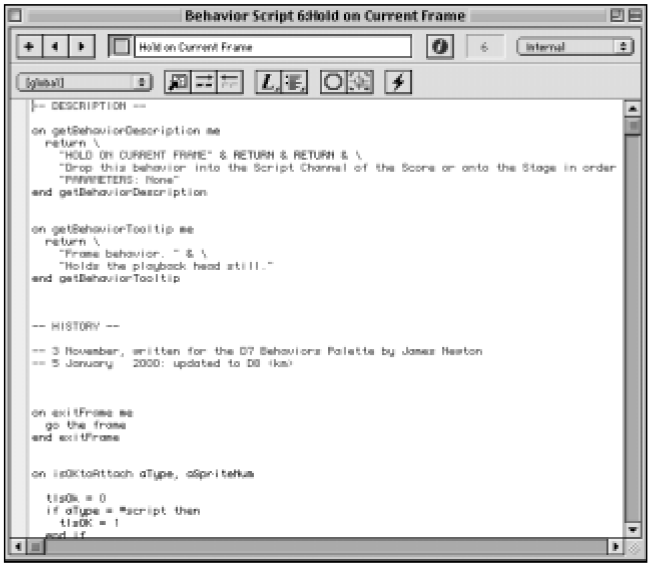
Dissecting someone else's script or behavior can help you learn Lingo, but don't change the behaviors until you understand what you are doing. A small change can cause a behavior not to function as intended.

In the next exercise, you open a behavior and display its "guts" using the Script window.

Examining the Hold on Current Frame Behavior

1. Open the engine2.dir movie that you modified in the preceding exercise.
2. Double-click the Hold on Current Frame behavior in the Cast window, and either the Script window or the Behavior Inspector appears. If the Behavior Inspector appears, click the Script button, identified in Figure 7-6, to display the Script window.
3. Use the scroll bar to scroll to the top of the script.

4. Review the script — and don't panic over its initial complexity. Some of the script is composed of comments, that is, lines of text preceded by the double dash (--), as shown in Figure 7-9. When you examine the essence of this behavior (that is, the *uncommented* lines), it is composed of three lines, with its "heart" being a `go the frame` instruction).



```
-- DESCRIPTION --

on getBehaviorDescription me
  return \
    "HOLD ON CURRENT FRAME" & RETURN & RETURN & \
    "Drop this behavior into the Script Channel of the Score or onto the Stage in order
    "PARAMETERS: None"
end getBehaviorDescription


on getBehaviorTooltip me
  return \
    "Frame behavior. " & \
    "Holds the playback head still."
end getBehaviorTooltip



-- HISTORY --

-- 3 November, written for the D7 Behaviors Palette by James Newton
-- 5 January   2000: updated to D8 (km)



on exitFrame me
  go the frame
end exitFrame


on isOKtoAttach aType, aSpriteNum

  tIsOk = 0
  if aType = #script then
    tIsOK = 1
  end if
```

Figure 7-9: The text of the Hold on Current Frame behavior script with the getBehaviorDescription instruction indicating that no parameters are required for this behavior.

5. Close the Script window and the Behavior Inspector.

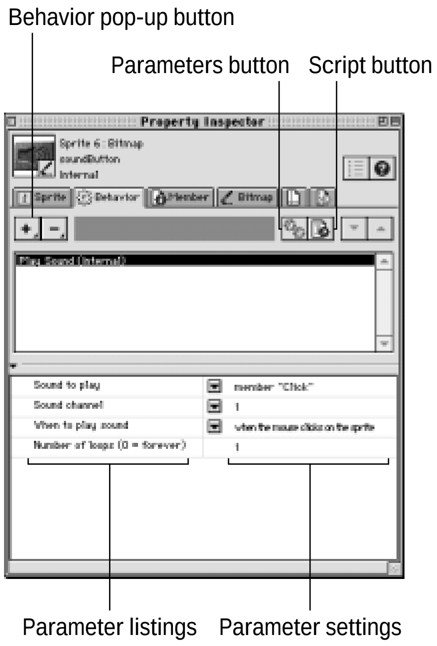
Adding behaviors by using the Property Inspector

You already know how to drop and drag a behavior onto a sprite, but you can also add a behavior to sprites by means of the Property Inspector. Unless you are creating a new Behavior — you do that later in this chapter — it saves time to add behaviors using the Property Inspector because you probably already have it open in order to adjust other properties in your movie. To do so, you need to:

1. Open the Library Palette by choosing Windows ➪ Library Palette. Select the behaviors that you want to use in your movie, and then drag them into the Cast window.

2. On the Stage, select a sprite to which you want to apply a behavior.

3. Click the Behavior button on the Sprite Overlay. This launches the Property Inspector with the behavior tab active.
4. Choose a Behavior from the Behavior Pop-up button, as shown in Figure 7-10.



Sound to play	member "Click"
Sound channel	1
When to play sound	when the mouse clicks on the sprite
Number of loops (0 = forever)	1

Figure 7-10: The Property Inspector enables you to apply behaviors to sprites. You can also easily modify the parameters of the applied behavior.

5. If the behavior you applied has parameters that need to be set, a menu appears, prompting you to choose them. After they have been set, the parameters are listed in the lower pane of the window. The parameters can be adjusted at any point during the creation process either by clicking the small arrow next to the parameter listing and choosing a new parameter setting from the pop-up menu, or by clicking on the Parameters button to display the Parameters dialog box.

Note

While the Property Inspector is open, you can repeat the process and attach additional behaviors to the selected sprite on the Stage.

6. When you have finished adding properties to the selected sprite, you select another sprite on the Stage to edit, and then add behaviors to it. When you are finished adding behaviors, either close the Property Inspector or leave it “floating” open on your desktop.

Adding behaviors in the Score window

In most situations, the fastest way to add behaviors to sprites is through the Score window. Usually when you are creating a movie in Director, you place all of the cast members on the Stage and position them, and then switch to the Score window to adjust the sprite span, add keyframes, and so on. The next logical step in the process is to add the behaviors to the sprites. Director enables you to do this without having to open the Property or Behavior Inspectors. Adding behaviors in the Score window doesn't enable you to see the parameters that you have set; however, you can always view and adjust them at any time by using the Property or Behavior Inspectors. Follow these simple steps to add behaviors in the Score window:

1. Open the Score window and select a sprite.
2. Click on the Behaviors pop-up menu, shown in Figure 7-11, and choose from one of the available behaviors displayed in the list that are in your Cast windows.
3. If any parameters need to be set, a menu appears, prompting you to set them.

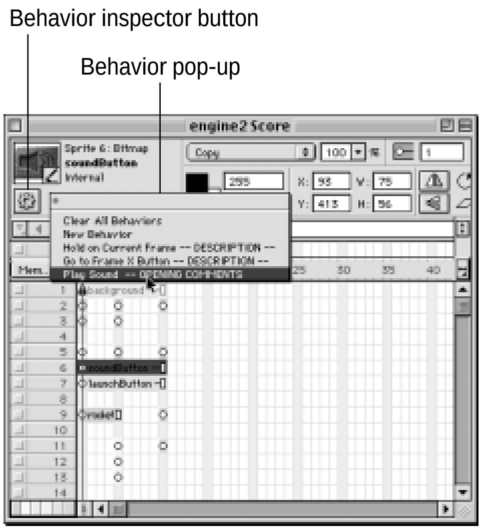


Figure 7-11: Applying a behavior to a sprite in the Score window

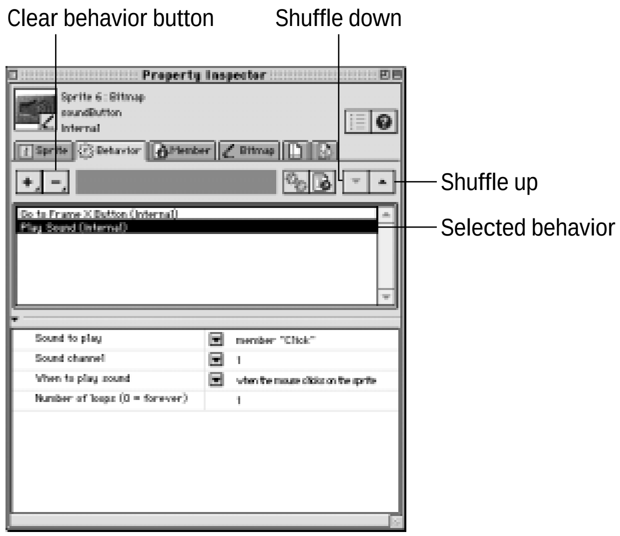
You can also clear all the behaviors applied to a sprite or create a new behavior. You can also click on the Behavior Inspector button in the Score window to launch the Behavior Inspector.

Caution Clicking the New Behavior option in the Score window pop-up menu launches the Script window, not the Behavior Inspector. This occurs even if you have chosen the Behavior Inspector as your default editor for behaviors.

Removing behaviors from a sprite

If you change your mind, you can also use the Property Inspector to remove one or more behaviors. To delete a behavior, you must:

1. Locate and select the sprite on the Stage (from which you want to remove a behavior).
2. Click on the Behavior button located in the Sprite Overlay.
3. In the Behavior tab of the Property Inspector, click to highlight the behavior in the Behavior List pane, as shown in Figure 7-12.



Sprite 6 : Bitmap
soundButton
Internal

Go to Frame X Button (Internal)
Play Sound (Internal)

Sound to play	member "Click"
Sound channel	1
When to play sound	when the mouse clicks on the sprite
Number of loops (0 = forever)	1

Figure 7-12: The Property Inspector with multiple behaviors listed

4. Press the Clear Behavior button and choose the Remove Behavior option. You can also delete a selected behavior by pressing the Backspace or Delete key.

Tip

You can also delete all of the behaviors applied to a sprite by choosing the Remove All Behaviors option in the Clear Behavior pop-up menu.

Reordering behaviors attached to a sprite

Behaviors are executed in the order in which they were originally attached to the sprite, unless you specifically change the order. To modify the execution order, you need to change the order in which behaviors are listed in the Behavior List pane in the Property Inspector or Behavior Inspector.

To change the order of execution, using the Property Inspector, complete these steps:

1. Open the Property Inspector and click on the Behavior tab.
2. Locate and select the sprite on the Stage to which the behaviors are applied that you want to reorder.
3. In the Behavior List pane of the Property Inspector, highlight the behavior that you want to reorder.
4. After selecting it, click either the Shuffle Up or Shuffle Down arrow, which are identified in Figure 7-12, until the behaviors are listed in the desired execution order.

In the following exercise, you add a second behavior to the Next button in the engine2.dir movie that you saved to your hard drive, and then you reorder the sequence in which the behaviors are executed.

On the CD-ROM

In the next exercise, you use the engine2.dir movie you saved earlier in this chapter. You can also use the engine2.dir movie found on the CD-ROM in the EXERCISE:CH07 (EXERCISE\CH07) folder.

Adding Media Behaviors to a Movie

1. Open the engine2.dir movie.
2. Make sure the Cast window is open. Open the Library Palette window.
3. From the Library List pop-up menu, select the Sound subcategory under the Media category.

Note

You need to drag behaviors located in the Library Palette into the Cast window before you can use the Behavior or Property Inspectors to apply them to a sprite.

4. Drag the Sound Beep behavior into the Cast window (see Figure 7-13).
5. Open the Score window and set the playback head to frame 1, and then click on the Next Screen button on the Stage.

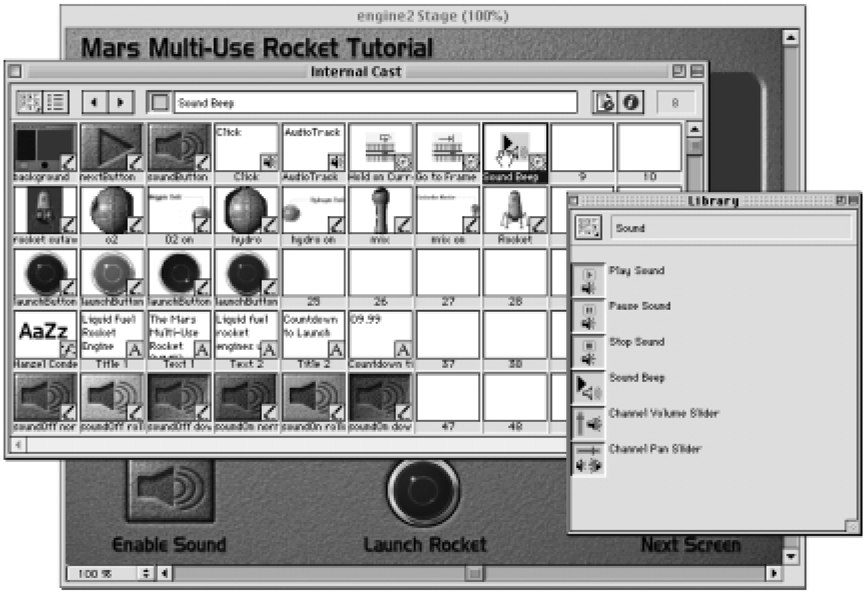


Figure 7-13: Drag the Sound Beep behavior from the Library Palette to the Cast window.

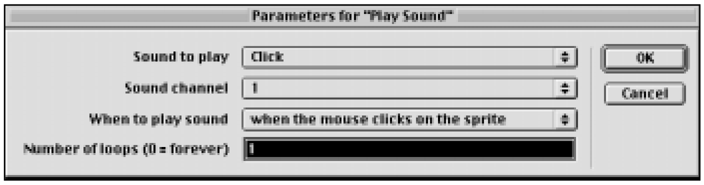
6. Click the Behavior button on the Sprite Overlay, and then select the Sound Beep behavior from the Behavior pop-up button in the Property Inspector.
7. Save the movie as **engine3.dir**. Rewind the movie and play it back. Click the Next Screen button. You will not hear the system beep, because the first behavior (go to frame 5) is executed before the second behavior's actions can occur (Sound Beep).
8. Stop and rewind the movie.
9. Select the Next Screen button on the Stage, open the Property Inspector, and click on the Behavior tab if it is not already open. The Inspector should display the two behaviors attached to the Next button sprite.
10. In the Property Inspector, select (highlight) the Sound Beep behavior, and then click the Shuffle Up arrow.
11. Save the movie again. Rewind the movie and play it.
12. Click the Next Screen button; you will hear a system beep before the movie advances to frame 5.
13. Stop the movie.

In the next few steps, you remove the behavior that plays a system beep and replace it with a behavior that plays an internal sound cast member (Click). You use the Property Inspector to accomplish these tasks.

Cross-Reference Chapters 5 and 18 provide detail on importing sound, using sound channels, and controlling sound using Lingo. For now, enjoy the simplicity of using a behavior to add a sound to your movie.

Attaching and Deleting Media-Control Behaviors Using the Behavior Inspector

1. Open the engine3.dir movie that you saved in the preceding exercise and rewind it to frame 1, if necessary. Make sure the Cast window is open.
2. Open the Library Palette and select the Sound subcategory to display the Sound behaviors. Drag the Play Sound Member behavior into the Cast window.
3. Click the Next Screen button on the Stage and then click the Behavior button on the Sprite Overlay to open the Behavior tab in the Property Inspector. The two behaviors previously attached to the Next button sprite are displayed in the window.
4. Activate (click) the Sound Beep behavior.
5. Press Delete or Backspace to remove the highlighted behavior.
6. Click the Behavior pop-up button and select the Play Sound Member behavior.
7. The Parameters dialog box for the behavior appears, as shown in Figure 7-14. You must specify (from a list of possible settings) the sound to be played, the sound channel to use, the initializing event, and how many times the sound should loop. You set these options in the following steps.



Sound to play: Click
Sound channel: 1
When to play sound: when the mouse clicks on the sprite
Number of loops (0 = forever): 1

Figure 7-14: The Parameters dialog box for the Play Sound Member behavior

8. The movie's internal cast includes two sounds (Click and Audio Track). By default, the Sound to Play field (shown in Figure 7-14) lists the Click cast member, which is the first sound cast member in your movie. If you click on the Sound pop-up menu button, you'll see all the sound cast members available in the current movie.
9. Leave the Sound Channel field set to 1 and the When to Play Sound field set to when the mouse clicks on the sprite (a friendly term for mouseDown). Leave the Number of Loops parameter set to 1 so that the sound will play only one time.

10. Click OK to accept the current parameters and attach the behavior to the Next button sprite.
11. In the Property Inspector, make sure the Play Sound Member behavior is listed as the first behavior. If not, select the behavior and click the Shuffle Up arrow to cause it to be listed (and executed) first.
12. Save the movie as **engine4.dir**.
13. Close all windows. Rewind and play the movie.
14. Click Next, and you hear the Click sound cast member play; then the movie advances to frame 5.
15. Stop the movie and rewind it.

In the following exercise, you use the drag-and-drop method to attach the Play Sound Member behavior to the Enable Sound button on the Stage. You can use this method to attach any of the behaviors in the Behavior Library Cast.

Dragging and Dropping a Media Behavior onto a Sprite

1. Open the engine4.dir movie that you created in the last exercise. Rewind the movie if needed.
2. With the Cast window open, locate the Play Sound Member behavior.
3. Drag and drop the Play Sound Member behavior onto the Enable Sound button on the Stage.
4. When the Parameters dialog box appears, which is similar to the one shown previously in Figure 7-14, you must specify the desired settings.
5. Choose the AudioTrack sound from the Sound to Play? pop-up menu.
6. Set the Sound Channel field to 2 and the When to Play Sound field to when the mouse is released over the sprite (a friendly term for mouseUp). You want the AudioTrack to play in sound channel 2 because the Click sound that you selected for the Next Screen button will play in sound channel 1. Set the Number of loops parameter to 0, which enables the member to loop indefinitely.

Caution Only one sound at a time can play in each sound channel. If you had specified sound channel 1 for the AudioTrack member, it would stop playing when the Next Screen button is clicked.

7. Click OK to finish setting the parameters for the behavior attached to Enable Sound sprite.
8. Save the movie as **engine5.dir**.

9. Close all windows. Then rewind and play the movie. Click the Enable Sound button, and the AudioTrack cast member plays when the mouse button is released. The click sound also plays if you click the Next Screen button.

10. Stop the movie and rewind it.

You could have set the AudioTrack member property to loop instead of setting it with the parameter in the Play Sound Behavior. Setting the looping parameter in the behavior gives you more flexibility than setting the cast member property to loop, because you can use it in other places in the movie without having the sound loop.

Caution

Setting a loop in the Play Sound Parameter can have very unexpected results if the member's looping property has also been set. Use one or the other.

Using the interactive behaviors

In addition to navigational and media behaviors, Director includes prescripted interactive animation behaviors. These behaviors alter the appearance of sprites on the Stage.

In the following exercise, you add the Rollover Member Change behavior to the engine5.dir movie, which changes the appearance of a specific sprite when the mouse pointer rolls over it. In this example, three parts of the rocket engine (the Oxygen Tank, the Mixing Chamber, and the Hydrogen Tank) will highlight and display a label describing the specific engine part. The sprites are located in channels 11, 12, and 13 of frame 5 of the Score, and are named O2, mix, and hydro.

On the CD-ROM

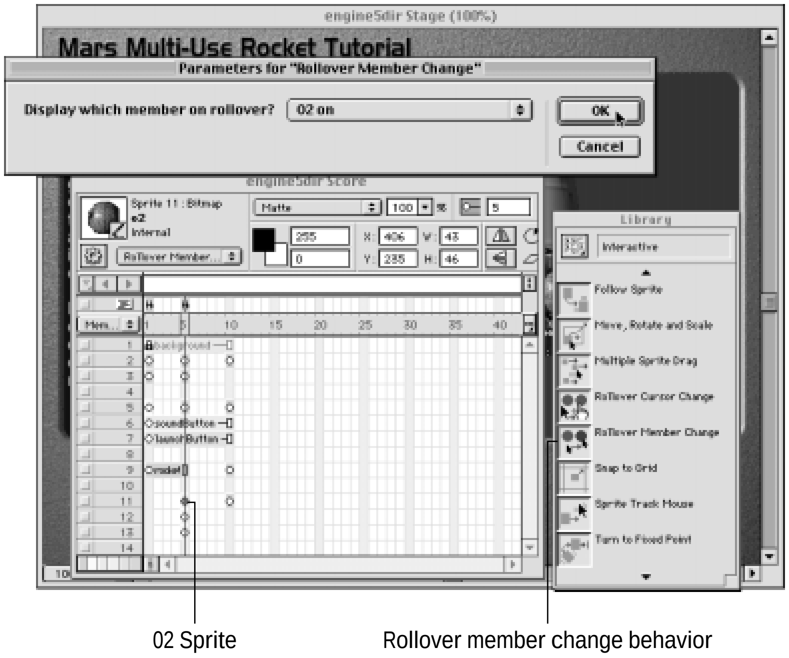
You can find the file engine5.dir on the companion CD-ROM in the EXERCISE:CH07 (EXERCISE\CH07) folder. You also can use the engine5.dir movie that you created in the preceding exercise in this chapter. You also use the Rollover Member Change behavior, which is in the Library Palette.

Adding a Behavior That Swaps One Sprite for Another

1. Open the engine5.dir movie. In the Score window, move the playback head to frame 5. Note that there are three sprites (in channels 11, 12, and 13) that are colored light blue. These are the sprites to which you will apply the Rollover Member Change behavior.

2. Be sure the Library Palette is open. Then choose Animation ➪ Interactive from the Library List pop-up menu. Locate the Rollover Member Change behavior.

3. Drag and drop the Rollover Member Change behavior from the Library Palette onto the O2 sprite, located on frame 5 in Sprite channel 11 of the Score, as shown in Figure 7-15.



Parameters for "Rollover Member Change"

Display which member on rollover? O2 on

Figure 7-15: You can drag a behavior from the Library Palette directly onto a sprite in the Score window.

4. When the Parameters dialog box appears, the cast member that is displayed in the cast member selection pop-up menu is called O2 on. This parameter always displays the next available cast member in the Cast window. Because the O2 on cast member is the one you want, click OK to return to Director's main window.

5. Select the sprite in channel 12 (mix) and apply the Rollover Member Change behavior, using the behavior pop-up menu in the Score window. Select mix on as the rollover member. Repeat this step for the hydro sprite in channel 13, using hydro on as the rollover member.

6. Save the movie as **engine6.dir**.

7. Close all windows, and play the movie. If you rewound the movie, click on the Next Screen button to advance to frame 5.

8. Move the mouse pointer over the engine parts on the rocket. Each part becomes highlighted and a label describing the part appears.

9. Stop the movie.

Creating complex rollover cast members

The effect created in the preceding exercise required some planning before the elements were brought into Director. By default, Director sets the registration point to the center of a cast member's bounding box when it is imported into the program. Because the rollover cast members are larger than the ones that are on the Stage, they would have appeared to jump when you rolled over them with the cursor. I could have reset the registration point of the rollover cast members in the Paint window, but this would have been very time-consuming. The easier solution was to add the connecting lines and text labels to the cast members after importing them into Director using the Paint window. This technique assured that the registration point would be exactly the same for both the original and rollover cast members because they were the same exact size when they were imported into Director. The labels had to be created at some point in the process, so adding them in the Paint window after importing them into Director added no time to the process of creating the graphics, yet saved quite a bit time in Director by not having to manually re-register them.

Creating Your Own Basic Behaviors

To this point, you have used the Behavior and Property Inspectors to add, remove, and view the behaviors attached to a specific sprite. You can also use the Behavior Inspector to create simple behaviors and still avoid the need to script or program interactivity from scratch.

Tip

The Behavior Inspector operates similarly to a macro editor, where specific choices you make are converted to the required Lingo code. As you use the Behavior Inspector to create new behaviors, you may be tempted to examine the scripts that are generated—which is a good learning technique—but you probably want to read Chapter 11 first. You can learn a great deal by observing how the Behavior Inspector translates your choices into basic Lingo functions.

The process required to create a new behavior varies depending on the nature of the behavior. In general, you must complete these steps:

1. Open the Behavior Inspector by choosing Window ➪ Inspectors ➪ Behavior Inspector, or by clicking the Behavior Inspector button in the toolbar or Score window.
2. Click the Behavior pop-up button and select the New Behavior option.
3. When the Name Behavior dialog box appears, as shown in Figure 7-16, enter an appropriate, descriptive name for the behavior, and click OK.

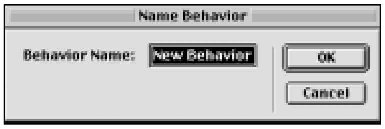
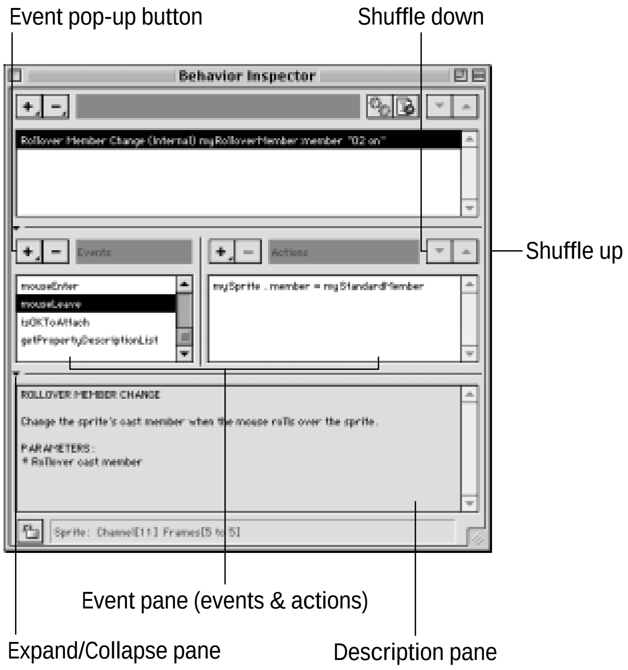


Figure 7-16: The Name Behavior dialog box

Note When you create a new behavior, it is stored either in the currently selected empty cast slot or in the next empty cast position, if the current slot is occupied.

4. If the editing pane is not visible, click the upper arrow to expand the Editing pane for the Behavior Inspector (identified in Figure 7-17).

Note The top arrow alternately contracts and expands the Editing Pane. The bottom arrow, identified in Figure 7-17, alternately expands and contracts the Description pane. The Editing pane displays the events and actions included in the current behavior. No events or actions appear when you first begin to create a new behavior.



Rollover Member Change (Internal) myRolloverMember:member "02 on"

mouseEnter
mouseLeave
isOKToAttach
getPropertyDescriptionList

mySprite . member = myStandardMember

ROLLOVER MEMBER CHANGE

Change the sprite's cast member when the mouse rolls over the sprite.

PARAMETERS:
* Rollover cast member

Sprite: Channel[11] Frames[5 to 5]

Figure 7-17: The Behavior Inspector with the Editing and Description panes expanded

5. At this point, you can add events to the behavior by using the Event pop-up button and add actions by using the Action pop-up button. Each time you add an event, you must add a corresponding action (that occurs in response to the event). *You can assign more than one action to each event.*

Note The Actions available from the Action pop-up menu are grouped into six categories: Navigation, Wait, Sound, Frame, Sprite, and Cursor. A seventh option, New Event, enables you to create a new, user-defined event. This feature requires that you know how to create scripts using Lingo.

Tip

You can use a similar process to edit events and actions assigned to an existing behavior. Simply highlight the behavior, select the event from the list, and then add new actions or delete any existing actions from the actions list. To delete an event or action, select it and then press the Delete or Backspace key.

6. You can alter the sequence of actions within a behavior by using the Shuffle Up and Shuffle Down arrows just above the Editing pane.

7. When you have finished adding events and their corresponding actions, you can close the Behavior Inspector (or leave it open) and save your movie — which now includes the newly defined behavior. Unfortunately, there is no way to add a description to a behavior without having to resort to Lingo.

Note

As you become familiar with Lingo (starting in Chapter 11), you'll discover that the Behavior Inspector provides a transparent way of creating Lingo code. The syntax of event names in the Behavior Inspector is similar to the Lingo code used for these behaviors (Mouse Up = mouseUp, for example).

To get good at building your own behaviors, you need to understand what types of actions generate each event. The Event found in the pop-up menu and the actions that generate them are listed in Table 7-1.

Table 7-1
Standard Events

Event	***Action That Generates the Event***
`MouseUp`	Occurs when the left mouse button is released after being pressed (clicked). It is called only once, and only if the mouse button is released while within the active area of a sprite (see `MouseWithin`).
`MouseUp`	Occurs when the left mouse button is pressed (clicked) but not released. Generally, it's better to assume that the user has not made a selection (such as clicking on a menu item or button) until the mouse button is released. This gives users a chance to move the mouse pointer off an object if they have inadvertently pressed the mouse button over the wrong object.
`MouseEnter`	Occurs when the cursor moves into the active area of a sprite. This event is generated only once upon each entry into the sprite, after which `MouseWithin` events are generated. This is a good point at which to display highlight states for a button, write out status messages, call pop-up menus, and so forth.

Event	***Action That Generates the Event***
`MouseWithin`	Occurs repeatedly while the cursor is within the active area of a sprite (that is, in the sprite's bounding rectangle, unless the sprite is displayed with Matte ink). The actions generated by this event should be short and to the point, because when you write a script that traps (responds to) this event, it can slow system performance if the mouse stays within a sprite's area for a prolonged period of time.
`MouseLeave`	The cursor moves out of the active area of a sprite; this is the only time this event is called. It's a good point at which to set highlight states for buttons back to their base states, revert status messages to the background message, hide pop-up menus, and so forth.
`KeyUp`	Occurs each time a key is pressed and released. Works only for controls that have text field components, and only if the control currently has the focus. See Chapter 14 for more information about text fields.
`KeyDown`	Occurs each time a key is pressed but not released. Only works for controls that have text field components and only if the control currently has the focus. See Chapter 14 for more information about text fields.
`RightMouseUp`	Occurs when the right mouse button is released. This obviously applies primarily to Windows-based machines, because Macintosh mice have only one button. To simulate a mouse right-click on the Macintosh, Control+click.
`RightMouseDown`	Occurs when the right mouse button is pressed. This obviously applies primarily to Windows-based machines, because Macintosh mice have only one button. To simulate a mouse right-click on the Macintosh, Control+click.
`PrepareFrame`	Occurs after the playback head exits a frame and prior to the next frame being painted on the screen. This is a good place to check to whether a visible sprite needs to be invisible or vice versa.
`ExitFrame`	Called after the frame has been drawn and all sprites have been tested for events. This is a good point for navigating to different frames.
`BeginSprite`	Occurs when the playback head moves to a frame that contains the first instance of the sprite. This event occurs only once, even if the playback head is looping on the frame.

Continued

Table 7-1 *(continued)*

Event	***Action That Generates the Event***
`EndSprite`	Occurs when the playback head leaves the last frame containing the sprite and moves to a frame where the sprite no longer exists. The event is generated after exitFrame.
`NewEvent`	Gives you the ability to create a customized event handler for the behavior. Using this feature requires that you know how to script using Lingo and how to create a user-defined handler, which is explained later, in the Lingo section of this book.

Creating a New Behavior

In the following exercises, you create a new behavior and then attach it to the Enable Sound button in the engine6.dir movie (after deleting a prepackaged behavior that plays a click sound when you click on the button). The new behavior applies the Add Pin ink to the sprite, making it appear to be highlighted when the mouse clicks on it and then reverses the ink back to the copy ink when the mouse button is released. The new behavior also plays the Click sound. You then add a mouse leave event that resets the sprite if the mouse rolls off the button before it is released.

On the CD-ROM

The file engine6.dir is on the companion CD-ROM in the EXERCISE:CH07 (EXERCISE\CH07) folder. Alternatively, you can use the version that you saved in the preceding exercise.

Building the mouseDown event

1. Open the engine6.dir movie and rewind it.

2. Click on an empty cell in frame 1 to deselect all sprites in the movie, or click an empty slot in the Cast window. Now open the Behavior Inspector.

Caution

If a sprite is selected when you create a new behavior, the behavior will be applied to the sprite automatically.

3. If the Editing pane is not expanded, click the top arrow to expand it. The Behavior Inspector should appear empty, like the one shown in Figure 7-18.

Tip

You can also resize the panes of the Behavior Inspector. To do so, move the mouse pointer to the horizontal lines (pane dividers) that divide the panes. When the cursor changes to a double-headed arrow (a resizing cursor), drag up or down to resize the pane.

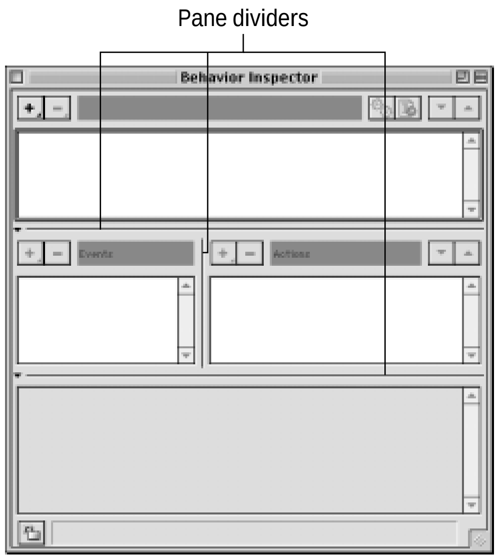


Figure 7-18: You can drag the pane dividers to resize the panes of the Behavior Inspector.

4. Click the Behavior pop-up button and select the New Behavior option.

5. When the Name Behavior dialog box appears, change the default behavior name to **Better Button** and click OK.

6. Click the Events pop-up button and select the Mouse Down option so that the Events pane matches Figure 7-19.

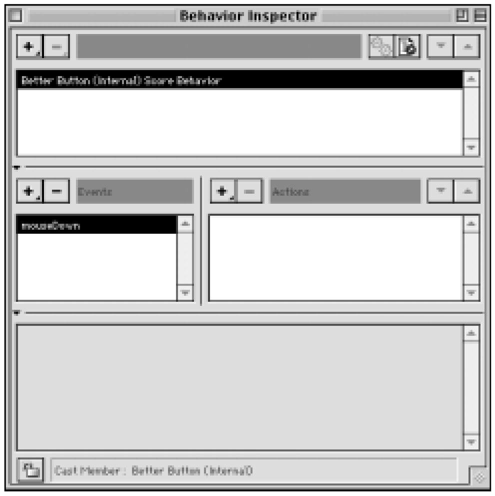

Behavior Inspector
Better Button (Internal) Score Behavior
Events
Actions
mouseDown
Cast Member : Better Button (Internal)

Figure 7-19: Attaching the mouseDown event to the Better Button behavior

7. Click the Action pop-up button and then choose Sprite ➪ Change Ink.
8. When the Specify Ink dialog box appears, select the Add Pin ink effect from the pop-up menu, as shown in Figure 7-20. You'll need to scroll down the list to find the Add Pin option. When the option is set, click OK to close the dialog box.

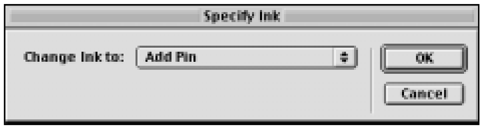


Figure 7-20: Choose Add Pin for the Change Ink option.

9. To add a second action to the `mouseDown` action, click the Action pop-up button and choose Sound ➪ Play Cast Member.
10. When the Specify Sound Cast Member dialog box appears, set the Play Sound setting to Click, as shown in Figure 7-21. Next, click OK to close the dialog box.
11. Save the movie as **engine7.dir**.

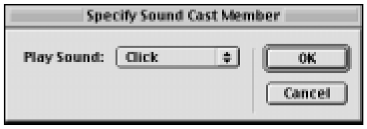


Figure 7-21: Choose Click for the sound cast member.

In the next exercise, you add a new event/action to the behavior, which causes the ink for the button to be reset to Copy on the Mouse Up action (when the user releases the mouse button).

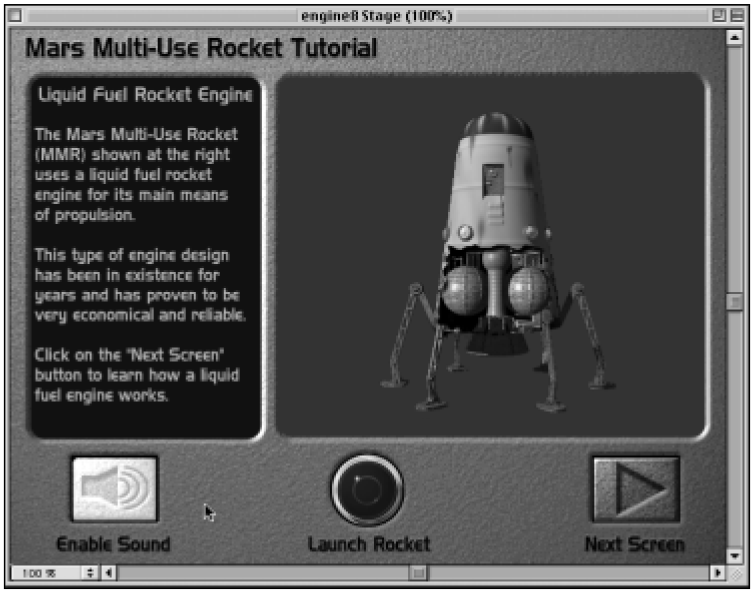
Creating the mouseUp Event and Action

1. Open the engine7.dir movie, and double-click on the Better Button behavior in the Cast window to activate the Behavior Inspector.
2. Click the Events pop-up button, and select the mouseUp option.
3. Click the Action pop-up button, and choose Sprite ➪ Change Ink.
4. When the Specify Ink dialog box appears, make sure the Ink effect is set to Copy, and then click OK to close the dialog box.
5. With the Behavior Inspector still open, select the Enable Sound button sprite on the Stage. Click the Behavior pop-up button, and select the Better Button property.

6. Click the Shuffle Up button so that the Better Button behavior will execute before the Play Sound Member behavior.

7. Save the movie as **engine8.dir**, rewind it, and play it. Click the Enable Sound button; you'll notice the change in ink and you'll hear the Click sound, and then the AudioTrack sound will play when the mouse is released over the button.

8. Now stop the movie, rewind it, and play it. This time, click the Enable Sound button, and with mouse button still down, drag the mouse off the sprite. Note that the button is still highlighted, as shown in Figure 7-22.



The Mars Multi-Use Rocket (MMR) shown at the right uses a liquid fuel rocket engine for its main means of propulsion.

This type of engine design has been in existence for years and has proven to be very economical and reliable.

Click on the "Next Screen" button to learn how a liquid fuel engine works.

Figure 7-22: The Enable Sound button remains highlighted after the cursor is dragged off the sprite before releasing the mouse button.

If users drag the cursor off the Enable Sound sprite before letting go of the mouse button, they can become confused, because the button is still highlighted and might appear to a user as if it should be doing something. To correct this problem, add a mouseLeave event that resets the ink of the Enable Sound sprite back to Copy.

Creating the mouseLeave Event and Action

1. Open the engine8.dir from the preceding exercise, if it's not already open.

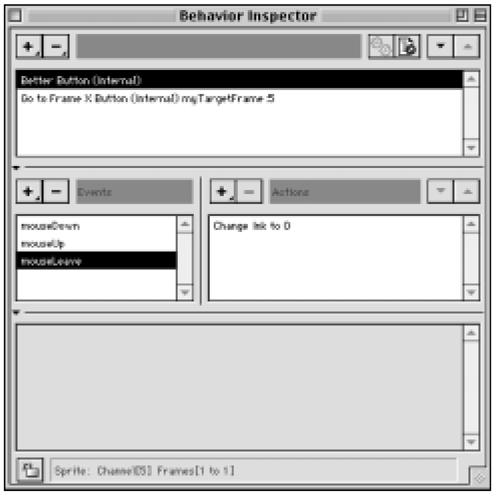
2. Open the Better Button behavior in the Behavior Inspector.

3. Click the Add Event pop-up button, and select the `mouseLeave` event. In the Actions pane, choose the Change Ink option and set the ink to Copy.
4. Save the movie as **engine9.dir**, and then rewind and play it. Click the Enable Sound button and drag the cursor off the button without letting go of the mouse button. The ink changes back to Copy, restoring the sprite to its original state.

Note

At this point, it's a good idea to replace the behavior applied to the Next Screen with your new Better Button behavior to ensure that both buttons behave in similar fashion.

5. Rewind the movie and stop it, making sure that the playback head is in frame 1. With the Behavior Inspector still open, select the Next Screen button sprite on the Stage or in the Score window.
6. In the Behavior List pane, select the Play Sound Member behavior and delete it.
7. Click the Behavior pop-up button, and select the Better Button behavior to attach it, as shown in Figure 7-23.



Better Button (Internal)
Go to Frame X Button (Internal) myTargetFrame 5

mouseDown
mouseUp
mouseLeave

Change Ink to 0

Sprite: Channel[5] Frames[1 to 1]

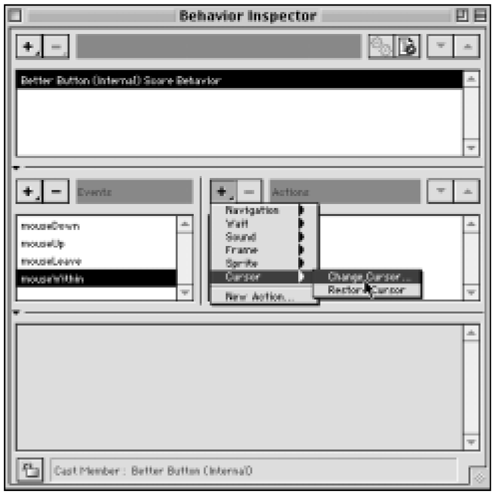
Figure 7-23: The modified behavior list

8. Shuffle the Better Button behavior so that it is first in the list of behaviors attached to the Next button.
9. Save the movie again as engine9.dir, and then rewind and play it. Click the Next Screen button and note the change in ink, the click sound, and that the playback head advances to frame 5.

The Better Button behavior still lacks some behaviors commonly associated with a "well-behaved button." In the next exercise, you add two more events/actions to the behavior that provide a visual cue to users when the mouse pointer moves over an active area on the Stage. You add an action to the new behavior that changes the cursor (mouse pointer) to a pointing finger when it moves over a button to which the behavior is attached. In Director, as in life, whatever is done must eventually be undone, so you also add a second behavior that restores the cursor to its normal state when the mouse pointer is not within the bounding box of the sprite (to which the behavior is attached).

Adding a Cursor Change to Your User-defined Behavior

1. With the engine9.dir movie and the Behavior Inspector still open from the preceding exercise, select the Better Button behavior in the Behavior List pane.
2. Click the Events pop-up button, and select the `mouseWithin` event.
3. Click the Action pop-up button, and then choose Cursor ➪ Change Cursor, as shown in Figure 7-24.



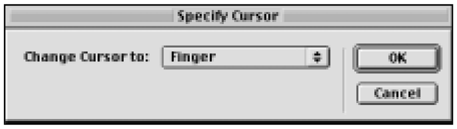
Better Button (Internal) Score Behavior

mouseDown
mouseUp
mouseLeave
mouseWithin

Cast Member : Better Button (Internal)

Figure 7-24: Select the Cursor option from the Actions pop-up menu.

4. When the Specify Cursor dialog box appears, select the Finger option (see Figure 7-25). Click OK to close the dialog box.



Change Cursor to: Finger

Figure 7-25: Choose the Finger cursor in the Specify Cursor dialog box.

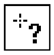
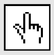
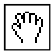
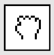
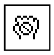
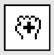
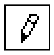
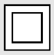
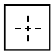
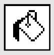
5. To add a fourth event/action to the Better Button behavior, one that sets the cursor back to a pointer after the behavior is exited, be sure the Better Button behavior is still selected in the Behavior List pane of the Behavior Inspector.
6. Click the `mouseLeave` event that you added earlier.
7. Click the Action pop-up button and choose Cursor ➪ Restore Cursor.
8. Save the movie as **engine10.dir**. Rewind the movie, and then play it. Move the mouse pointer over the Sound and Next buttons (both have the Better Button behavior with the modified events attached). Note the change in the cursor when it is over the two buttons as compared to when it is not. Try out the Sound and Next buttons and note that all the previous actions assigned to this behavior still work.

Director has almost any cursor type that you might need for your movies. Table 7-2 Shows each of the built-in cursors, as well as a description of the situation in which you should use them. Each cursor also has a number to which you can refer when creating Lingo scripts that call a particular cursor. Many programs use these cursors to indicate their tools and the cursors have come to mean specific kinds of operations. Note that switching to a specific cursor in your movie does not cause that cursor to perform that operation. You still have to develop the behaviors to do that.

Note The graphics of the cursors listed in Table 7-2 were taken from a Macintosh. Some cursors will look slightly different on Windows computers.

Table 7-2
Director's Built-in Cursors

Cursor	*Name*	*Use*	*Cursor Number*
↖	Arrow	Standard Arrow pointer used to point on the screen. Usually the default cursor setting.	0
I	I-Beam	Used to indicate a text insertion point.	1
+	Crosshair	Used to indicate the starting point of a selection.	2
✜	Crossbar	Denotes a selection that will be added to.	3
⌚	Watch	Usually used to indicate when the movie is paused in order to perform a read or write file activity.	4

Cursor	*Name*	*Use*	*Cursor Number*
	Blank	Hides the cursor. Used mainly for touch-screen applications.	200
	Help	Used to point to an element on the screen that the user needs help using.	254
	Finger	Indicates that some action can be taken if the user clicks on the element over which the cursor is positioned.	280
	Hand	Enables the user to move the Canvas in much the same way that a piece of paper can be moved around a desktop.	260
	Closed Hand	Used for when a user clicks on an element that can be dragged.	290
	No Drop Hand	Indicates the element cannot be placed in the current position.	291
	Copy Close Hand	Usually used with an Option (Alt) modifier key to drag a copy of an element.	292
	Pencil	Used to draw a single pixel on the Stage. Often used with the Shift modifier key to constrain the selection horizontally, vertically, or to a 45-degree angle.	256
	Eraser	Erases pixels that are under the cursor. Usually replaces them with the current background color. Often used with the Shift key to constrain the eraser horizontally, vertically, or to a 45-degree angle.	257
	Select	Indicates that a rectangular bounding box can be created to make a selection. Often used with the Shift modifier key to constrain the selection to a square.	258
	Bucket	Fills any connected pixels containing the same color that the icon is over.	259

Continued

Table 7-2 *(continued)*

Cursor	***Name***	***Use***	***Cursor Number***
	Lasso	Indicates that the user can create a free-form selection. Often used with the Shift modifier key to constrain the selection horizontally, vertically, or to a 45-degree angle.	272
	Dropper	Used to indicate that the user can “suck” the color that is currently used under the cursor.	281
	Air Brush	Used to indicate the user can spray pixels onto the screen, using a predetermined size and flow setting.	301
	Zoom In	Indicates the screen will Zoom in by a predetermined increment from the chosen point.	302
	Zoom Out	Indicates the screen will Zoom out by a predetermined increment from the chosen point.	303
	Vertical Size	Usually used to indicate that a dialog box or window element can be sized vertically.	284
	Horizontal Size	Usually used to indicate that a dialog box or window element can be sized horizontally.	285
	Diagonal Size	Usually used to indicate that a dialog box or window element can be sized diagonally.	286

Creating Multi-State Buttons

Until now, the button behaviors that you have created in this chapter have all used a single cast member. To enhance the look and feel of a button, you can create a *multi-state* button that consists of four different cast members that are assigned to the various states of the button. An effective multi-state button changes according to the event that is occurring in the life of the button. It should highlight when the mouse rolls over it, appear to be pressed while the mouse is being clicked on it, and then pop back up to its original state when the mouse button is released. In some situations, there should also be an inactive state that informs the user that the button is not available in the current context of your presentation. Adding a sound that plays when the button is clicked completes the user experience.

Director has a couple of very useful behaviors that make it easier to create multi-state buttons: the Push Button and the Multi-State Toggle behaviors. Although both behaviors enable you to assign a different cast member for each event, they each perform a different action.

- ✦ You use the Push Button behavior to create multi-state buttons that will either revert to their original state or that will appear to be inactive after they have been clicked.
- ✦ The Multi-State behavior acts like a toggle. You can vary the states of a single button between an on state and an off state. This behavior can also be applied to a group of buttons that will act like radio buttons. When one button in the group is set to its on state, the other buttons in the group are set to their off state.

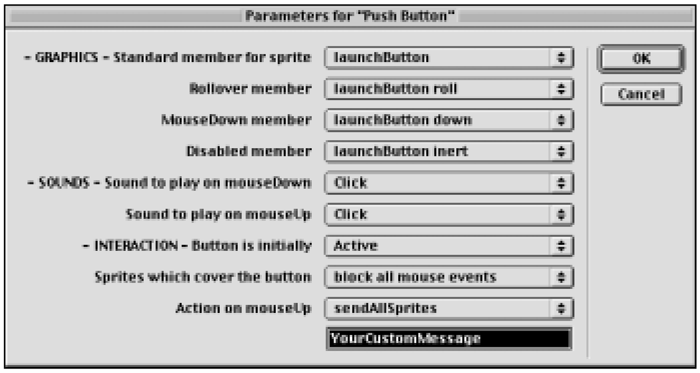
Creating a push button

In the next exercise, you apply the Push Button behavior to the Launch Rocket button and set the parameters of the behavior so that the button appears to light up when the mouse is rolled over it, and then appears to be pressed when it's clicked on. You also add a click sound to the `mouseDown` and `mouseUp` states. You then apply the Rollover Cursor Change behavior so that the cursor changes to a finger when it is rolled over the sprite. This parameter works the same way as the change cursor event that you added to the Better Button behavior earlier in this chapter.

On the CD-ROM For the following exercise, you need to use the engine10.dir movie that you saved in the preceding exercise, or you can use the engine10.dir file found on the CD-ROM in the EXERCISE:CH07 (EXERCISE\CH07) folder.

Applying the Push Button Behavior

1. Open the engine10.dir movie, and then open the Library Palette.
2. Locate the Push Button behavior in the Controls category, and drag it onto the Launch Rocket sprite on the Stage.
3. The Parameters dialog box for this behavior appears. Note that the cast members were automatically assigned in the graphics options for this behavior. Like some of the other behaviors that you applied to other sprites, this behavior applies the cast members that follow the one to which the sprite is referring. Unless you have rearranged the cast members used for the Launch Rocket button, they should be in the correct order. The cast members that are applied to each event in the graphics options should be the ones shown in Figure 7-26.


Parameters for "Push Button"

– GRAPHICS – Standard member for sprite	launchButton
Rollover member	launchButton roll
MouseDown member	launchButton down
Disabled member	launchButton inert
– SOUNDS – Sound to play on mouseDown	Click
Sound to play on mouseUp	Click
– INTERACTION – Button is initially	Active
Sprites which cover the button	block all mouse events
Action on mouseUp	sendAllSprites
	YourCustomMessage

Figure 7-26: A cast member is applied to each event in the graphics options parameters.

Tip

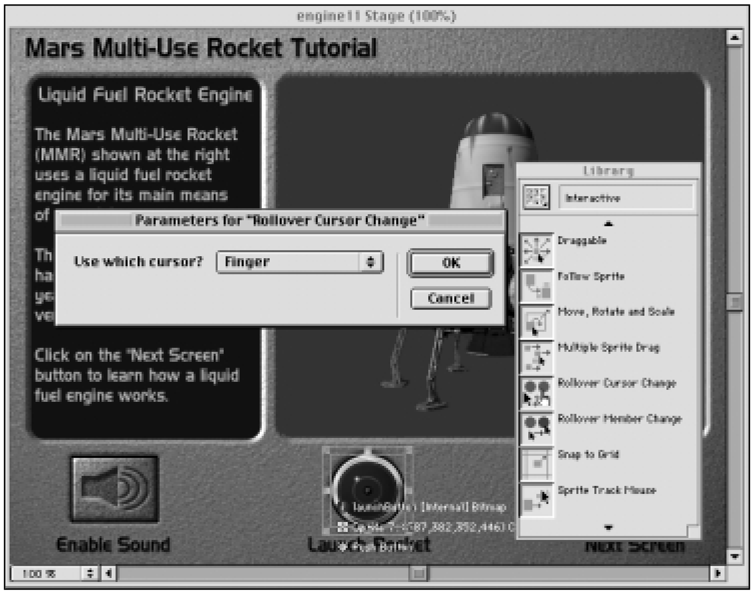
In the Cast window, arrange the cast members in the order that they will appear in the behavior Parameters dialog box. This will save a lot of time if you are applying the behavior to several buttons. It's also a good idea to add the name that is used for each state of the button ("roll," "down," "inert") as the second word in the cast member's name. This makes it much easier to find the correct cast member for each state of the button.

4. With the Parameters dialog box still open, select the Click sound cast member for both the Sound to play on mouseUp and the Sound to play on mouseDown options.

Caution

If you choose to add a sound option to this behavior, make sure that you specify a sound cast member for both the mouseUp and mouseDown options; otherwise, this behavior will not function correctly.

5. Use the default settings for the last four options, because you want the button to be active on the Stage, and no sprites are covering the sprite to which this behavior is being applied.
6. Save the movie as **engine11.dir**. Rewind the movie and play it. Roll the mouse over the Launch Rocket button and click on it to see the different states that have been applied.
7. Stop the movie, and drag the Rollover Cursor Change behavior (choose Animation ➪ Interactive in the Library Palette) onto the Launch Button sprite on the Stage.
8. Make sure that the Finger cursor is selected in the Rollover Cursor dialog box, and then click OK to return to Director's main window (see Figure 7-27).



engine11 Stage (100%)

Mars Multi-Use Rocket Tutorial

Liquid Fuel Rocket Engine

The Mars Multi-Use Rocket (MMR) shown at the right uses a liquid fuel rocket engine for its main means of

Click on the "Next Screen" button to learn how a liquid fuel engine works.

Parameters for "Rollover Cursor Change"

Use which cursor? Finger

Figure 7-27: The Rollover Cursor change behavior enables you to change the cursor to a finger when it rolls over a sprite.

9. Save the movie again as engine11.dir, and then rewind the movie and play it. The cursor now changes to a finger when it is rolled over the Launch Rocket button.

Later in this chapter, you add the navigation component to this button, which will advance the playback head to the frame in the Director movie where the sequence that launches the rocket occurs.

In the next exercise, you rebuild the Enable Sound button, using the Multi-State button behavior, so that it acts as a toggle to turn the sound on or off.

Creating a multi-state toggle button

You may have noticed that there is no way to turn off the AudioTrack sound other than by stopping the movie after it has started playing. You could create another button that turns off the sound, but this takes extra screen real estate, which is often at a premium in multimedia applications. A more effective way to turn the sound on and off is to have the Enable Sound button act as toggle — when the button is first clicked, the sound begins playing; if the button is clicked again, it stops the sound. The Enable Sound button should also have a visual cue informing the user that the button is either on or off.

The Multi-State Button behavior works in a way similar to the Push Button behavior in that you can specify different cast members for each event that occurs. The big difference is that you can specify cast members that respond to events for both the on and off states of the button. You can also assign a Lingo command that tells the button what operation to perform in the on and off states.

Adding the Multi-State Button Behavior

1. Open the Score window in the engine11.dir movie. Select the Sound button sprite in channel 6.
2. Open the Cast window, and expand the window so that the six cast members in slots 41 through 46 are visible. These are the cast members that you will use to build the new Enable Sound button (see Figure 7-28).

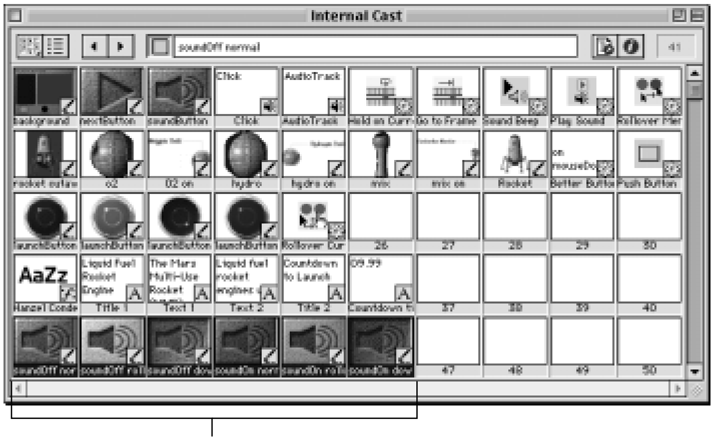


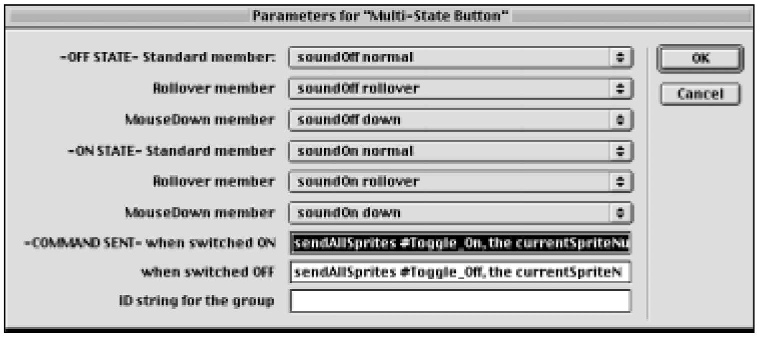
Figure 7-28: The six cast members used for the Enable Sound button.

3. Select the cast member named soundOff normal in the Cast window, and choose Edit ➪ Exchange Cast Members or press Command+E (Ctrl+E) to switch the current cast member in sprite channel 6 with the soundOff normal cast member. Look at the Sprite Overlay for the Enable Button sprite on the Stage, and make sure that the soundOff normal cast member is listed in the cast member description.
4. Click on the Behavior button located on the Sprite Overlay for the soundOff normal sprite to open the Property Inspector. Select the behaviors that have been applied to the sprite, and then delete them.
5. Open the Library Palette and locate the Multi-State Button behavior in the Controls category, and then drag it onto the Enable Sound button sprite, either on the Stage or in the Score window.

6. When the Parameters dialog box appears, note that the cast members have been automatically applied to the Standard, Rollover, and `mouseDown` events for both the off state and the on state options. The members that have been applied should match the ones shown in Figure 7-29.

Note

If the cast members have been moved from their original order in the Cast window, you may need to select the correct cast members from the pop-up menus located next to each event option. The second word of each cast member name is the same as the state to which the cast member should be applied.


Parameters for "Multi-State Button"

-OFF STATE- Standard member:	soundOff normal
Rollover member	soundOff rollover
MouseDown member	soundOff down
-ON STATE- Standard member	soundOn normal
Rollover member	soundOn rollover
MouseDown member	soundOn down
-COMMAND SENT- when switched ON	sendAllSprites #Toggle_On, the currentSpriteNu
when switched OFF	sendAllSprites #Toggle_Off, the currentSpriteN
ID string for the group	

Figure 7-29: Make sure that the cast members that are shown here are the ones that are applied to the off state and on state events for the behavior.

7. After making sure that the correct cast members are assigned to the Parameters listed in the dialog box (see Figure 7-29), click OK or press Enter (or Return, if you're using a Macintosh) to set the options and return to Director's main window.

Note

For now, you leave the COMMAND SENT and ID string parameters as they are. A Lingo command can be added to both the COMMAND SENT when switched ON and the COMMAND SENT when switched OFF parameters, shown in Figure 7-29, that can cause the button to perform a specific operation when the button is toggled to an on or off state. The default setting for these two parameters sends a message to all of the other sprites in the frame, notifying them of the current state (on or off) of the sprite that can be sent to other sprites on the Stage or to a Lingo movie script. For another sprite to take advantage of this command, it needs a behavior applied to it that contains a Lingo handler that can intercept the message. The ID string for the group parameter enables you to assign a group name to a series of buttons that will use this same behavior. This is useful for creating radio buttons. Because you are applying this behavior to only a single button, there is no need to enter a group name in this parameter.

8. Click on the behavior button of Enable Sound sprite to open the Property Inspector. Add the Rollover Cursor Change behavior to the sprite from the Behaviors pop-up menu.

9. Save the movie as **engine12.dir**. Then rewind the movie and play it back. Roll the cursor over the Enable Sound button and note that it becomes highlighted. When you click on the button, it changes to a different color, indicating that it's active. When you click the button again, it reverts to its original state.

When the Multi-State button behavior is applied to a group of buttons, the button that is clicked on is set to the ON state and all the other buttons are set to their OFF state. This behavior can enable you to create custom multi-state radio buttons that are more visually appealing than Director's built-in radio buttons.

In the next exercise, you add a very basic Lingo command that plays the AudioTrack sound cast member when the Enable Sound button is in its ON state, and then turns the sound off when the button is in its OFF state.

Adding a Lingo command to a behavior

Several of the behaviors in the Library Palette enable you to add simple Lingo commands that perform specific operations, such as triggering a behavior that's applied to a different sprite in the frame. You can also send messages to custom-built Lingo handlers that can perform a variety of operations. You learn a great deal about Lingo commands and creating handlers beginning in Chapter 11 of this book. You don't have to be a "power programmer" to use Lingo commands that increase the capabilities of a behavior.

You use the `puppetSound` Lingo command to play the AudioTrack sound cast member when the Enable Sound button is toggled to its ON state, and then turn the sound off when the button is toggled to its OFF state. Playing sounds with Lingo is discussed in detail in Chapter 17. For now, you only need to know how to specify the sound channel that the sound will play in, specify the sound cast member that will play, then turn it off again.

Adding the puppetSound Lingo Command

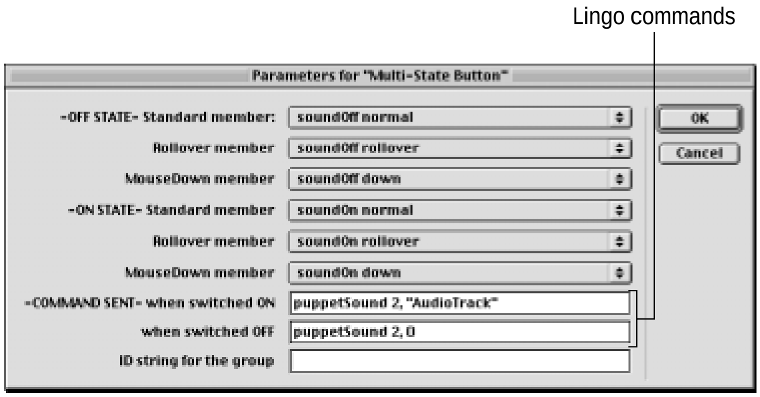
1. Make sure the engine12.dir movie that you saved in the preceding exercise is open in Director. In the Cast window, select the Audio Track member, and click on the Properties button to open the Property Inspector. Click on the Loop check box to enable the sound file to loop whenever it is played.

2. With the Property Inspector window still open, select the Enable Audio sprite on the Stage or Score. You may need to click on the Behavior tab to make the Behavior window active. Click on the Parameters button in the Property Inspector to display the Parameters dialog box for the behavior.

3. In the Command Sent When Switched On field, type **puppetSound 2, “AudioTrack”**. Make sure that you type this command exactly as shown in Figure 7-30, including the comma and quotation marks. This command tells Director to play the sound cast member called AudioTrack in sound channel 2 when the button is toggled to its ON state.
4. In the COMMAND SENT when switched OFF field, type **puppetSound 2, 0** (see Figure 7-30). This command turns off the sound that’s playing in channel 2 when the button is toggled to its OFF state. The 0 in this Lingo statement is actually the equivalent of a `FALSE` command that acts as toggle to turn off the sound that is currently playing. You learn about true/false statements in Chapter 11.

Note

True and false statements are Lingo’s way of turning a property on or off. They act like the on/off switch found on most electronic components. In fact, the symbol used for the on/off switch on most electronic equipment is a 1 or 0.



Parameters for "Multi-State Button"

-OFF STATE- Standard member:	soundOff normal
Rollover member	soundOff rollover
MouseDown member	soundOff down
-ON STATE- Standard member	soundOn normal
Rollover member	soundOn rollover
MouseDown member	soundOn down
-COMMAND SENT- when switched ON	puppetSound 2, "AudioTrack"
when switched OFF	puppetSound 2, 0
ID string for the group	

Figure 7-30: Add the puppetSound Lingo command with statements that define the cast member and the sound channel, and turn off the sound.

5. Save the movie as **engine13.dir**. Rewind the movie and play it. Click on the Enable Sound button. The button toggles to its ON state and the AudioTrack sound cast member plays. Click the button again, and the sound stops playing and the button toggles to its OFF state.

Adding simple Lingo commands is one way to greatly expand the capabilities of a behavior. As you learn more about Lingo in the second half of this book, you will be able to combine custom-built Lingo scripts with Director’s built-in behaviors to create very powerful multimedia applications.

Navigating with Markers

Earlier we mentioned that multi-state buttons are far more useful if they actually do something when you click them. Fortunately, Director gives you an abundant collection of options as to what these buttons can do. One of the simplest and most useful roles for a multi-state button is providing the capability to jump to a different location in the Score.

In earlier chapters, we briefly discussed markers, showing you how to use them to organize the Score. Markers are also indispensable for creating Director movies that will play back nonlinearly. After you begin learning scripting behaviors (in Chapter 11), you'll discover that a simple instruction can direct the playback head to a frame number or a marker.

Adding markers to your movie

Most of Director's basic navigational commands and navigational behaviors make use of markers within a movie. As you learned earlier, markers are labels that you attach to specific frames.

The steps to adding a marker are:

1. In the Score window, locate the Markers channel. As shown in Figure 7-31, the Markers Channel is immediately above the Effects channels.

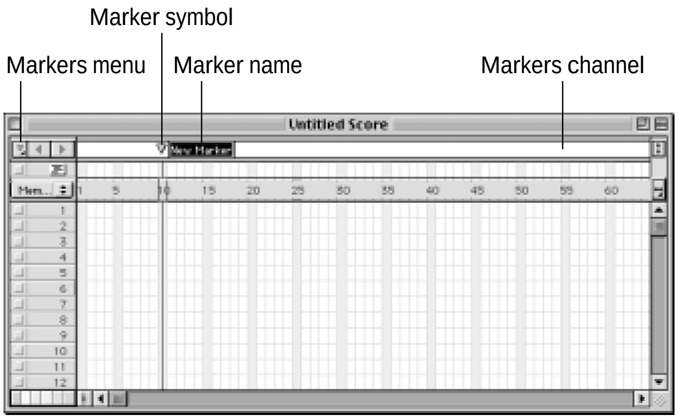


Figure 7-31: The Markers channel in the Score window

2. Click in the Markers channel, above the frame where you want to add a marker. A symbol appears for the marker, as shown in Figure 7-31, along with a default marker name (New Marker).
3. You typically want to use a more specific marker name, such as Start. Just type it in, and the name you type replaces the default New Marker name.

Tip You can move a marker to a new location by dragging the marker symbol to a different frame.

To delete a marker, click on the triangular marker symbol and drag up or down off the Markers channel.

You can choose from three different methods to display a list of all the markers in your movie:

✦ Choose Window ➪ Markers. In the Markers window, double-clicking on a marker name jumps the playback head to that marker.

✦ Click the Markers menu (see Figure 7-31) to display a pop-up menu of all markers. On the Markers menu, click the desired marker name to jump to that marker in your movie.

✦ Press Command+Shift+M (Ctrl+Shift+M) or choose Window ➪ Markers to display the Markers window shown in Figure 7-32.

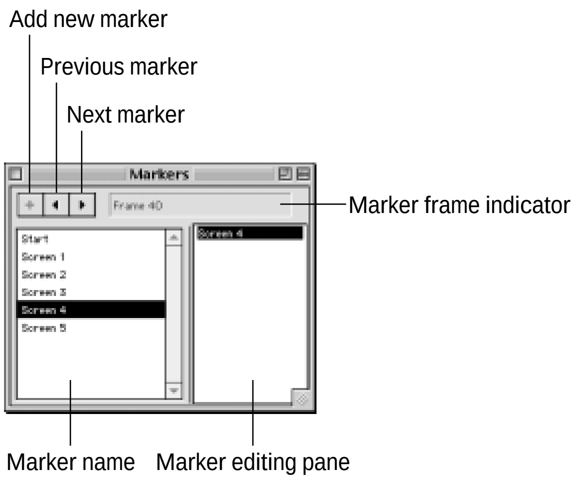


Figure 7-32: Use the Markers window to view all of the markers in your movie.

Now that you can add markers to your movies, you can use them to control the flow of your movie. You can combine button behaviors with frame and marker navigation to really enliven your interactive presentations.

In the next series of exercises, you add markers to the engine13.dir movie, and then add a jump to marker behavior to the Launch Rocket button.

On the CD-ROM

You use the engine13.dir movie that you saved in the previous exercise, or you can use the engine13.dir movie on the CD-ROM in the EXERCISE:CH07 (EXERCISE\CH07) folder.

Adding Markers for Navigation

1. Open the engine13.dir movie in Director; then make sure the Score window is active.
2. Click in frame 1 of the Marker channel. Type **Start.**
3. Click anywhere in the Marker channel. Type **Rollovers**, and then drag the marker to frame 10.
4. Click anywhere in the marker channel and add a marker named **Launch rocket**. Then drag it to frame 15. The Score should look like Figure 7-33.

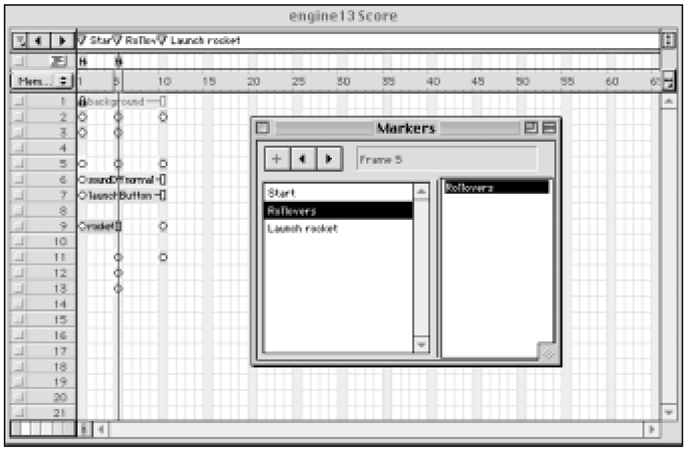


Figure 7-33: Click on a marker in the Marker window to move the movie’s playback head to the frame that contains the selected marker.

5. Open the Markers window by choosing Window ➪ Markers or by pressing Command+Shift+M (Ctrl+Shift+M). Make sure that frames that have sprites in them are visible in the Score window, as shown in Figure 7-33. Click on one of the markers in the Marker window and the playback head jumps to the frame that contains the marker you selected.
6. Save the movie as **engine14.dir**.

Now you need to add a behavior to the Launch Rocket button that jumps to the marker called Launch Rocket. The behavior you will add is called the Jump to Marker Button and is in the Controls category in the Library Palette.

Using the Jump to Marker behavior

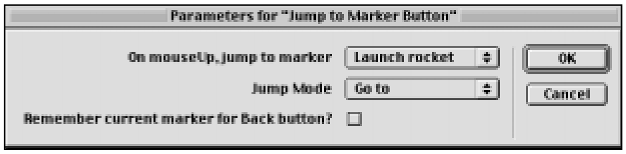
The Jump to Marker Button is a very flexible behavior that makes it easy to add navigation elements to your buttons. Like most other behaviors, it can be added to other behaviors that have already been applied to the sprite. This behavior automatically creates a list of the markers that are contained in the movie so that you can quickly choose any marker from the pop-menu that's presented in the Parameter dialog box.

Tip

It's a good idea to add markers to your movie as early as possible in the production process. You can always delete them, add new ones, or move them around later. Not only does this make the movie easier to navigate while you're building it, but it takes advantage of the Jump to Marker Button behavior's capability to create a list of the markers contained in the movie.

Adding the Jump to Marker Button Behavior

1. Make sure that the engine14.dir movie that you saved in the preceding exercise is open in Director. Open the Library Palette, and choose the Controls category from the Library List button.
2. Locate the Jump to Marker Button behavior, and drag it into the Launch Rocket Button on the Stage.
3. The Parameters dialog box appears, as shown in Figure 7-34. For the On mouseUp, jump to marker parameter, six items appear in the list. The first three, Previous, Loop, and Next, enable you to jump to the next marker, the previous marker, or loop on the current marker. The last three items (Start, Rollovers, and Launch rocket) are the markers that are contained in the movie. Select the Launch rocket item from this list.


Parameters for "Jump to Marker Button"

On mouseUp, jump to marker: Launch rocket
Jump Mode: Go to
Remember current marker for Back button? ☐

Figure 7-34: The dialog box for the Jump to Marker Button behavior enables you to specify a marker, determine how the movie goes to the marker, and set a parameter that remembers the frame from which the movie jumped.

4. Use the default Go to option for the Jump Mode parameter. You can uncheck the Remember current marker for Back button, although it won't affect the behavior in any way if you leave it checked. Click OK to return to Director's main window.
5. Click the Parameters button to display the Parameters dialog box for the behavior (see Figure 7-34).

Note

There are two options in the Jump Mode pop-up menu. They are Go to, which jumps to the specified marker, and Play and Return, which is used when you want to go to a marker to perform some activity and then jump back to the marker that you came from. If you choose the Play and Return option, you will probably want to make sure that the check box located next to the Remember current marker for Back button is checked. This enables the behavior to remember which markers have been visited.

6. Now you need to add a Hold on Current Frame behavior to frame 10 (under the Launch rocket marker) so that the movie loops on that frame. Open the Score window and click in frame 10 of the script channel.
7. Select the Hold on Current Frame behavior from the Behavior pop-up list in the Score window, as shown in Figure 7-35.

Note

To perform the operation in Step 7, you may have to enable the Sprite toolbar in the Score window by Control+clicking (Alt+clicking) the Frame bar and choosing the Sprite Toolbar option.

8. Save the movie as **engine15.dir**. Rewind the movie and play it back. Click on the Launch Rocket button. It now jumps to the marker called Launch rocket.

Caution

Try to avoid having two markers with the same name. The Jump to Marker Button behavior jumps to the first marker that contains the name specified. If you have two markers with the same name, you might have unexpected results.

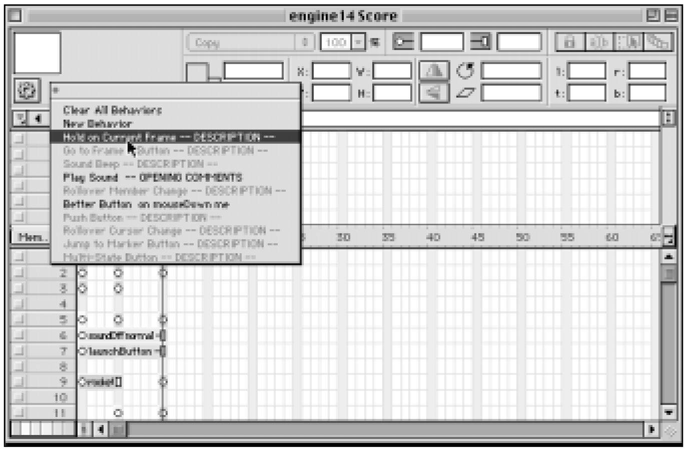

engine14 Score
Clear All Behaviors
New Behavior
Hold on Current Frame -- DESCRIPTION --
Go to Frame Button -- DESCRIPTION --
Sound Beep -- DESCRIPTION --
Play Sound -- OPENING COMMENTS
Rollover Member Change -- DESCRIPTION --
Better Button on mouseDown me
Push Button -- DESCRIPTION --
Rollover Cursor Change -- DESCRIPTION --
Jump to Marker Button -- DESCRIPTION --
Multi-State Button -- DESCRIPTION --

Figure 7-35: Apply a behavior to the script channel or a sprite in the Score window from the Behaviors pop-up menu.

When you use the Jump to Marker Button behavior in combination with the Jump Back behavior, you can have a button jump to a specific marker and then jump to frame that it came from. Most of the other button navigation behaviors found in the controls category act similarly to the Jump to Marker Button behavior. You can use the other button behaviors in the Controls category to jump to a different movie, and then return to the previous movie.

By now you have learned that Director's built-in behaviors are very powerful, enabling you to add sophisticated interactivity to your movies. You can add multiple behaviors to a sprite that add layers of interactivity and good user interface design to the navigation elements in your movies.

The movie to which you have been adding these behaviors is pretty simple; there aren't too many buttons that need to have multiple behaviors applied to them. In a large project, applying many individual behaviors to each sprite can be a time-consuming task. This is where Lingo comes to the rescue: You can create custom behaviors that combine the events and parameters of several different behaviors.

Where Did that Behavior Come From?

As mentioned earlier, behaviors are *scripts* (and you'll learn a great deal more about scripts in Chapter 11). Unlike many other development languages, Director's scripts are actually kept as a resource. Specifically, the script is a cast member that contains the text of the script, as well as the internal code that stores the script when played at run-time. From the standpoint of the developer, a script is an autonomous entity, and behaviors are especially independent.

The independent nature of scripts and behaviors means that they can be collected in a cast and used in a variety of projects. An *external cast* can really include any kind of cast member (bitmap, text, script, video, and so on). The Library Palette is really just a collection of external casts that are stored in the Libs folder located in the Director Application folder. Although the Library Palette as shipped with Director contains only behaviors, you can add to the Library Palette any kind of cast member that can be contained in a Cast window. For instance, if you're using a logo that appears in all of your movies, you can put the logo into an external cast and then move it into the Libs folder; it will appear in the Library Palette.

A custom multi-state button behavior we call the MultiPurpose Button behavior was developed specifically for this book, although it might be useful to you in a variety of circumstances. Along with the MultiPurpose Button behavior, we also include several types of multi-state button cast members, and several sounds that were optimized for use with buttons. These elements are contained in a Cast Library called coolbuttons.cst. You can link them to your own movies and use them royalty-free, although you should read the comments (included as a text cast member in the Library) for more details.

On the CD-ROM

The coolbuttons.cst Cast Library is on the CD-ROM in the EXERCISE:CH07 (EXERCISE/CH07) folder. There is also a folder in the Goodies section of the CD-ROM called COOLBUTTONS, which contains all of the original artwork created for the buttons found in the coolbuttons.cst Cast Library. They are supplied in both Photoshop 5.0 and Fireworks 2.0 formats, which you can modify using either of these two programs. Last but not least, is a folder called Buttons that you can drag into your Libs folder; it includes the buttons arranged by convenient categories in your Library Palette.

Cool button factories

The artwork that was created for all of the buttons in the coolbuttons.cst Cast Library started life as Photoshop 5.0 artwork. Both Photoshop and Fireworks have the capability to create graphic documents that contain many layers. You can take advantage of this capability by creating what this author calls factories, which have layers containing all of the states of a button in a single document.

There are several advantages to this technique. The first is that you need to manage fewer documents for the original artwork that you create for your movies. A second advantage is that you can view how the different states of the button look by turning on and off each layer that contains the different button state graphic. After you are satisfied with each state of your button, you can create the individual cast members by turning on the appropriate layers and performing a Save As operation in Photoshop or Exporting in Fireworks.

Spend some time looking at the button factories to get a better idea of the power of this technique. In fact, after you become familiar with the factories, feel free to modify them (by adding text labels, for example) for your own movies.

Cast libraries

In traditional programming languages such as C and C++, creating an application is sometimes a challenge. Many developers manage the overall project by breaking it up into distinct tasks — that is, by separating the application into modules. Director originally didn't support this method — everything had to be contained in a single movie, and that typically meant a lot of duplication if one movie needed to have the same resources as another.

Cast Libraries were Director 5.0's answer to the demands of developers for a more efficient container of resources. A Cast Library is a separate and discrete entity. It has its own cast members and can be used to store any cast members that the normal *internal* cast can use. There's no real limit to the number of Cast Libraries a movie can open (although certainly system limitations will limit the number).

Cast Libraries come with some drawbacks. Because they are linked as an external source, you need to be careful not to move either the director movie that is using the library or the Cast Library itself to another location. You also need to include any external Cast Libraries that are used in a movie when outputting a Director movie as a Projector, Shockwave movie, or Java applet. It also possible to accidentally modify an element in a linked cast and then save over the original Cast Library, thus losing the original element forever. If you have elements such as scripts, behaviors, or graphics that you want to preserve in their original state, you might want to put the casts into the Libs folder so that they will appear in Library Palette. As you know by now, when you modify a behavior (or any other element that can be contained in the Library Palette), you are actually modifying a copy of the element that has been placed into an internal Cast window.

Adding a custom-built behavior to your movie

We have created an external Cast Library called coolbuttons.cst that contains a very useful, all-purpose button behavior that combines the button states, cursor change, and navigation parameters into a single behavior. There are also several multi-state button graphics that you can use royalty-free in your Director movies.

You will link the coolbuttons.cst Cast Library to your movie, and then replace the three behaviors that have been applied to the Launch Rocket button with the custom-built behavior called the MultiPurpose Button behavior.

On the CD-ROM The coolbuttons.cst is on the CD-ROM that came with the book in the EXERCISE: CH07 (EXERCISE\CH07) folder.

Linking Cast Libraries to Movies

1. Copy the coolbuttons.cst external Cast Library to a personal folder (where you saved your other Director movies) on your local hard drive.
2. Open the engine15.dir movie.
3. Choose Modify ➪ Movie ➪ Casts, or press Command+Shift+C (Ctrl+Shift+C) to display the Movie Casts dialog box (see Figure 7-36).

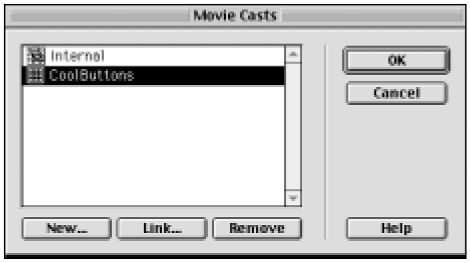


Figure 7-36: The Movie Casts dialog box enables you to link Cast Libraries to your current movie.

4. Click the Link button. This displays an Open (file) dialog box, where you can select a Cast Library (any file with the .cst or .cxt extension). Locate the coolbuttons.cst external cast and double-click on it to link it to the current movie. After the cast is linked, you can access it through the Cast window.

Note

If you have the CD-ROM in your CD-ROM drive, you can link to coolbuttons.cst on the CD-ROM located in the EXERCISE:CH07 (EXERCISE\CH07) folder, or if you've copied it to your local drive, you can locate it there.

5. Click OK to close the dialog box.
6. If the Cast window is not visible, open it by pressing Command+3 (Ctrl+3).
7. Click the Choose Cast button (the button in the upper-left corner of the Cast window), and a list of the active Cast Libraries for the current movie appears (see Figure 7-37).
8. Select CoolButtons to make the library available to the current Director movie.
9. Click the Cast Library button again and set the cast back to Internal, which is the default cast for a movie. This setting ensures that any new cast members you add will be added to the Internal cast rather than to the imported (CoolButtons) cast.
10. Save the movie as **engine16.dir**.

You can also incorporate the coolbuttons Cast Library into the Library Palette. Just drag the coolbuttons.cst into the LIB folder found in the application folder for Director 8. This will enable you to access the MultiPurpose Button behavior and all of the button bitmap and sound cast members from the Library Palette. The advantage to this approach is that the behavior, button bitmap, and sound cast members will automatically be included in your movies.

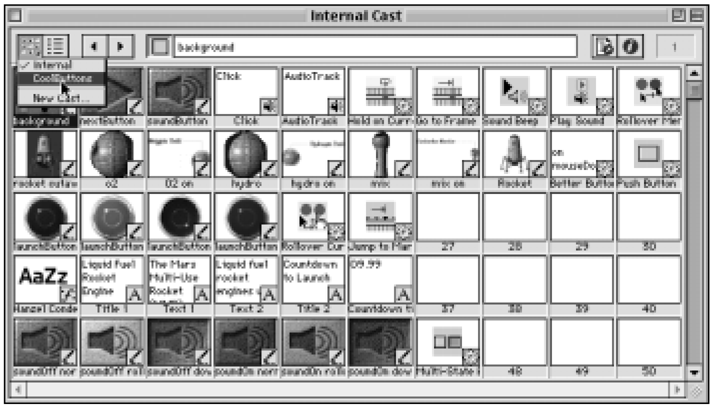


Figure 7-37: Clicking the Cast Library button brings up a list of all the current Cast Libraries.

Still another way of accessing the cast members of a library is to place the Cast Library in the Xtras folder located in the application folder containing your copy of Director. This technique makes the cast available from the Xtras menu. For example, if you drag the coolbuttons.cst into the Xtras folder, you'll see a CoolButtons entry in the Xtras menu. Note, however, that the cast is *not* actually linked to the file. You have to physically drag one or more cast members from the cast into your movie's internal cast to make them work. If you use this method, Director prompts you whether to link the external cast or just transfer the selected cast members to the internal cast.

A cast library case study

Two of the authors of this book were building a Director-based engine for an interactive news journal that would be updated on a daily basis. This was a huge undertaking that required several programmers and multimedia authors working simultaneously to meet the ever tightening deadline for the release of the product. The Director templates and interface elements being built for the application were being produced at the same time that the code was being written, so we had to have a way to update all of the different components. Cast Libraries came to our rescue. We divided up the components into several shared Cast Libraries that were stored in a specific location on the network, and then linked to the Director the movies that the multimedia authors were building. Because the source code was stored as several components in a linked Cast Library, new functionality and bug fixes were globally updated in all of the individual Director movies. It was an amazing sight to see a bug magically fixed on four different Director movies running on four different computers at the same time. We also used this approach to create custom authoring tools that were developed for the application.

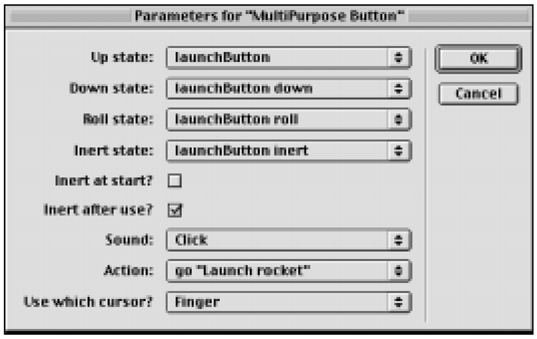
Now that you have the Coolbuttons Cast Library attached to your movie, it's time to replace the behaviors attached to the Launch Rocket button with the MultiPurpose Button behavior that was custom-built using Lingo. The advantage to the MultiPurpose Button behavior is that it replaces the Push Button, Rollover Cursor Change, and Jump to Marker Button behaviors with a single behavior that enables you to set all of the parameters used for the three behaviors that were previously applied to the sprite.

Adding a Custom Built Behavior

1. Open the engine16.dir movie in Director. Click on the Launch Rocket sprite on the Stage, and then click on the Behavior button in the Sprite Overlay to open the Property Inspector.
2. Delete the three behaviors that are applied to the sprite.
3. Add the MultiPurpose Button behavior from the Behavior List button.
4. When the Parameters dialog box appears, make sure the cast members selected for Up State, Down State, Roll State, and Inert State are the launchButton cast members shown in Figure 7-38.

Tip

If you add the name of the state to which the cast member will be applied, it makes it much easier to locate them when it comes time to apply the cast member's parameters to a multi-state button.



Up state:	launchButton
Down state:	launchButton down
Roll state:	launchButton roll
Inert state:	launchButton inert
Inert at start?	☐
Inert after use?	☑
Sound:	Click
Action:	go "Launch rocket"
Use which cursor?	Finger

Figure 7-38: All of the cast members, except for the Up state, have the name of the state they are designed for as the second word in the cast member name.

5. Click on the Inert after use? check box so that the button will be inactive after it has been clicked.
6. Select the Click sound cast member for the sound parameter.
7. Click on the Action pop-up menus, and choose the go "Launch rocket" option. This behavior creates a list of the markers that are contained in the movie and adds them as options for the Action parameter.

8. Make sure that the Finger option is set for the Use which cursor? parameter. If you click on the pop-up menu for this parameter, you will see a list of all the built-in cursors for Director.

9. Save the movie as **engine17.dir**. Rewind the movie and play it, and then click on the Launch Rocket button. The Launch Rocket button acts the same way as it did when it had the three different behaviors applied to it.

Note

You can activate the button after it has been set to an inert (inactive) state by double-clicking on it. You can also activate the button by sending a custom message called the #enable message to the sprite with Lingo.

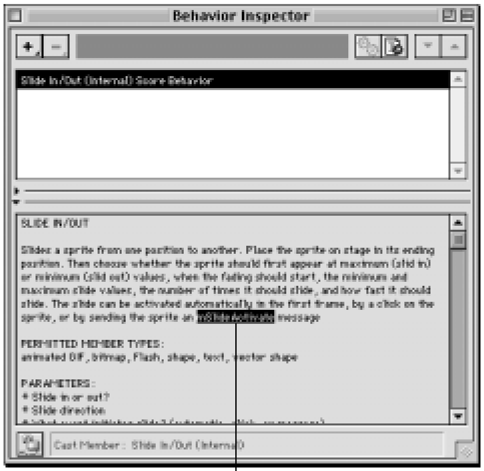
The MultiPurpose Button behavior was completely written in Director using Lingo. If you had to add the three original behaviors that were applied to the Launch Rocket button to 50 or 60 other buttons, it would be very time-consuming and would probably require a lot of debugging. This is a situation in which a custom-built behavior such as the MultiPurpose Button can save you a lot of creation time as well as prevent a swollen mouse finger.

There are many situations where a well-designed custom behavior will save a tremendous amount of time building a Director movie. Suppose that you are building an application in which you want to test the user's comprehension of the subject matter at different points in the Director movie. To accomplish this goal, you want to build a quiz in which the user drags pictures next to the words that define them. You could design a behavior in Lingo where you assign the matching graphic and text cast members, and add a parameter to apply sound cast members for positive and negative feedback sounds. There could also be a field where you could type in a custom message that will appear as a custom alert message. Combining all of these events into one single behavior would enable you to spend most of your time creating high-quality visuals for the quizzes instead of linking all of the pieces together with a series of individual Lingo scripts that would have to be modified for every quiz.

Sending a Message from One Behavior to Another

Many of Director's built-in behaviors can send a message to other sprites, behaviors, frame scripts, and cast member scripts. These messages are usually sent to a Lingo *handler* — a set of instructions that tells Director what to do when an event occurs. A message can be sent to a specific sprite, to all the sprites in the frame, to a specific behavior, to a frame script, or to a movie script. You learn how to send messages with Lingo starting in Chapter 11. In this chapter, we look at sending simple messages from one behavior to another.

You can find out whether a behavior can send or receive a message by looking at the behavior's description in the Behavior Inspector. Director's built-in behaviors will describe the message that can be sent to or from the behavior. Figure 7-39 shows the message that can be used to activate the SlideIn/Out behavior. There is a parameter setting for this behavior that can activate the behavior when the playback head enters the frame, by clicking on the sprite to which the behavior is attached, or by sending the mSlideActivate message from another behavior or Lingo script.



SLIDE IN/OUT

Slides a sprite from one position to another. Place the sprite on stage in its ending position. Then choose whether the sprite should first appear at maximum (slid in) or minimum (slid out) values, when the fading should start, the minimum and maximum slide values, the number of times it should slide, and how fast it should slide. The slide can be activated automatically in the first frame, by a click on the sprite, or by sending the sprite an mSlideActivate message

PERMITTED MEMBER TYPES:
animated GIF, bitmap, Flash, shape, text, vector shape

PARAMETERS:
* Slide in or out?
* Slide direction

Cast Member : Slide In/Out (Internal)

Message to activate the behavior

Figure 7-39: The descriptions for Director's built-in behaviors will list any specific messages that can be sent to and from the behavior.

Note You have to drag a behavior from the Library Palette into the Cast window before you can view the behavior's description in the Behavior Inspector.

The mSlideActivate message shown in Figure 7-39 activates the Slide In/Out behavior. When the behavior receives the message, it causes the rest of the events in the behavior to execute.

Sending a message from a behavior

One of Director's built-in behaviors that can be activated by a message sent from another behavior is the Countdown Timer behavior. This behavior is located in the text category of the Library Palette. The Countdown Timer behavior can be applied to a text or field cast member to create a digital clock that counts from a predetermined time down to zero. A message or command can be sent from this behavior to activate a behavior applied to another sprite when the timer reaches zero.

In the next two exercises, you apply the Countdown Timer behavior to a text cast member that will execute when the movie enters the Launch Rocket frame. When the timer sprite reaches zero, it sends a message that initiates the Slide In/Out behavior that has been applied to the Rocket sprite, causing the rocket sprite to fly off the top of the Stage.

On the CD-ROM

You need the engine17.dir Director movie for the next exercise. You can use the one that you saved in the preceding exercise or use the engine17.dir file on the CD-ROM in the EXERCISE:CH07 (EXERCISE\CH07) folder.

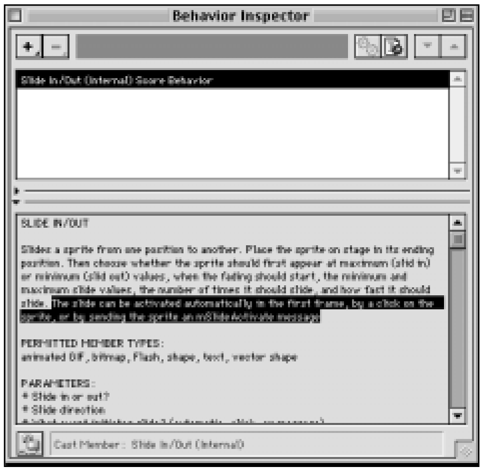
Creating a Countdown Timer

1. Open the engine17.dir movie in Director, and make sure the Cast window is active.
2. Open the Library Palette (choose Window ➪ Library Palette). Click on the Library List button, and select the Text category. Locate the Countdown Timer behavior and drag it into the Cast window.
3. Choose Animation ➪ Automatic in the Library Palette, locate the Slide In/Out behavior, and drag it into the Cast window. You can close the Library Palette to get it out of the way.

Note

You will be sending a message from the Countdown Timer behavior to the Slide In/Out behavior, so you need to find out what message the Slide In/Out behavior requires to initialize it.

4. Locate the Slide In/Out behavior in the Cast window and double-click on the cast member to open it in the Behavior Inspector. In the Description pane is a sentence that describes how the behavior can be activated (see Figure 7-40). One of the options listed informs you that this behavior can be sent a message called mSlideActiviate. Double-click on the mSlideActiviate text in the Description pane to select it, and copy it to the Clipboard.



Slide In/Out (Internal) Score Behavior

SLIDE IN/OUT

Slides a sprite from one position to another. Place the sprite on stage in its ending position. Then choose whether the sprite should first appear at maximum (slid in) or minimum (slid out) values, when the fading should start, the minimum and maximum slide values, the number of times it should slide, and how fast it should slide. The slide can be activated automatically in the first frame, by a click on the sprite, or by sending the sprite an mSlideActivate message.

PERMITTED MEMBER TYPES:
animated GIF, bitmap, Flash, shape, text, vector shape

PARAMETERS:
* Slide in or out?
* Slide direction

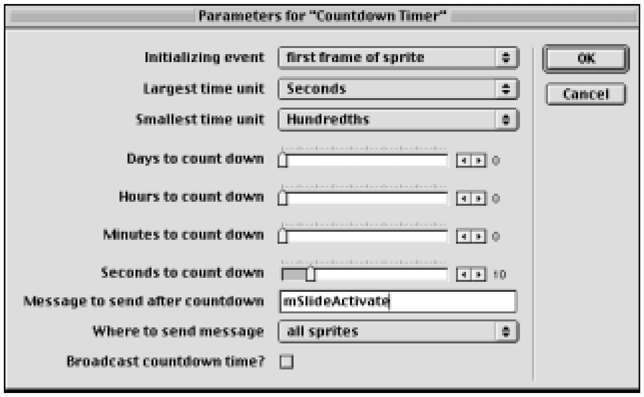
Cast Member: Slide In/Out (Internal)

Figure 7-40: The sentence describing the options that activate this behavior is highlighted.

Note

If the behavior opens in the script window, you should adjust the preferences so that behaviors open in the Behavior Inspector. Choose File ➪ Preferences ➪ Editors and then select the Behavior Inspector as the editor for behaviors.

5. Close the Behavior Inspector. In the Score window, advance the playback head to frame 10, and then on the Stage, select the text sprite called Countdown Timer. Drag the Countdown Timer behavior onto the sprite. The Parameters dialog box for the Countdown Timer behavior appears, as shown in Figure 7-41.



Parameters for "Countdown Timer"

Initializing event	first frame of sprite
Largest time unit	Seconds
Smallest time unit	Hundredths
Days to count down	0
Hours to count down	0
Minutes to count down	0
Seconds to count down	10
Message to send after countdown	mSlideActivate
Where to send message	all sprites
Broadcast countdown time?	☐

Figure 7-41: The parameters for the Countdown Timer behavior should be set as they are shown in this figure.

6. Note that this behavior can also be initialized by a message that is sent to it; however, in this situation, you will want the behavior to be activated when the playback head enters the frame where the sprite is located. Use the default setting for the Initializing event parameter.

7. Choose Hundredths from the Smallest time unit parameter pop-up menu. This choice displays the timer in hundredths of a second.

8. Move down to the Seconds to count down slider, and click on the advance arrow until the slider displays 10 seconds. You can also drag the slider to 10 seconds if you prefer.

9. Double-click the text in the field located next to the Message to send after countdown parameter to select it. Paste the mSlideActivate message, which you copied to the Clipboard earlier in this exercise, into the field. You can also type the mSlideActivate message into this field, but be careful that you type the message correctly.

Tip

Whenever possible, it is better to select the actual message, copy it to the Clipboard, and paste the message into the field where you want the message to appear. This avoids the possibility of introducing a typing error that could cause the behavior to work incorrectly.

10. Use the default setting of all sprites for the Where to send message parameter, because you will be sending the message to another sprite. There is no need to check the Broadcast countdown time parameter in this situation.

Note

The Broadcast countdown time parameter constantly sends the current time to all of the other sprites on the Stage. You are concerned only about sending a message when the timer reaches zero, so enabling this parameter will have no effect.

11. Click OK to set the parameters and return to Director's main window. Save the movie as **engine18.dir**. Rewind the movie and play it. Click on the Launch Rocket button, and you will see the countdown sprite ticking off the seconds until it reaches zero.

Even though the mSlideActivate message is being sent from the Countdown Timer behavior, nothing happens when the timer reaches zero seconds. The Slide In/Out behavior that this message will activate needs to be added to the rocket sprite before anything can happen. In the next exercise, you add the Slide In/Out behavior to the rocket sprite and set the parameter to activate the behavior when the behavior receives the mSlideActivate message.

Receiving a message from another behavior

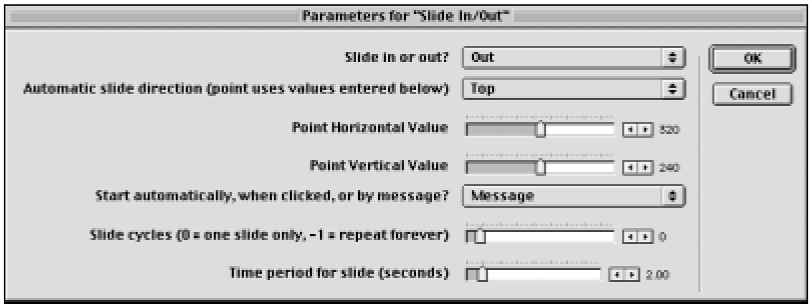
As you know, you can activate several of Director's built-in behaviors by means of a custom message sent from a behavior applied to a sprite or to a Lingo script contained in the movie. Having a behavior be activated when a message is sent to it enables you to create very dynamic, event-driven animations that can happen in a single frame of your movie. This can save a lot of space in the Score window as well as make your movies more compact.

You can create very complex Lingo movie scripts that can globally intercept messages that have been sent from behaviors, and then send the messages to appropriate sprites on the Stage to activate the behaviors that are applied to them. Combining Director's built-in behaviors with custom Lingo scripts can save quite a bit of development time, because you would only create Lingo scripts that would interpret and distribute the messages. You can learn more about sending and receiving messages to and from sprites beginning in Chapter 11; this concept is used throughout the Lingo section of the book.

You can still take advantage of the messaging capabilities of a behavior without having any Lingo programming knowledge to execute them. There are many situations in which simply selecting the Activate with Message parameter for a behavior will accomplish the task that you need the sprite to perform. In the next exercise, you apply Director's Slide In/Out behavior to the rocket sprite and set a parameter that initializes the behavior when the it receives the mSlideActivate message that is sent from the Countdown Timer behavior. When the behavior is activated, the rocket sprite automatically flies off the top of the frame.

Using a Message to Activate a Behavior

1. Open the engine18.dir movie that you saved in the preceding exercise in Director. In the Score window, advance the playback head to frame 10.
2. On the Stage, select the rocket sprite, and then drag the Slide In/Out behavior from the Cast window onto the sprite. The Parameters dialog box for the behavior appears (see Figure 7-42). The parameters for this behavior can be set to determine whether the sprite will slide in or out of the frame, and the direction that the sprite will slide. You can set a point to which you want the sprite to slide, determine how the behavior will be activated, select how many times the behavior will execute, and define the time period over which the sprite will slide.



Parameters for "Slide In/Out"

Slide in or out?	Out
Automatic slide direction (point uses values entered below)	Top
Point Horizontal Value	320
Point Vertical Value	240
Start automatically, when clicked, or by message?	Message
Slide cycles (0 = one slide only, -1 = repeat forever)	0
Time period for slide (seconds)	2.00

Figure 7-42: The Slide In/Out behavior contains several parameters that can be set to control how the animation occurs.

3. Select the Out option from the Slide In or Out pop-up menu. This choice causes the sprite to move off the frame when the behavior is activated.
4. Select the Top option from the Automatic Slide Direction parameter's pop-up menu so that the Sprite slides off the top of the frame.

Note

You use the Horizontal and Vertical point values if you select the Point option for the Automatic Slide Direction parameter. You can set the Horizontal and Vertical points on the Stage to (or from) which you want the sprite to slide, depending on the Slide In or Out option that you selected.

5. Choose the Message option from the pop-up menu next to the Start automatically, when clicked, or by message parameter. This behavior contains a Lingo handler that intercepts the mSlideActivate message sent from the Countdown Timer behavior and activates the rest of the behavior.

6. Use the default setting (0) for the Slide cycles parameter, because you want the sprite to animate only one time.

7. You can use the slider controls next to the Time period for slide parameter to set the amount of time that the animation will use to slide the sprite to its end position.

8. Click OK when you have selected the parameters that you want for the behavior to return to Director's main window.

9. Save the movie as **engine19.dir**. Rewind and play it. Click on the Launch Rocket button. The Countdown Timer activates and begins counting down to zero. When the timer reaches zero, it sends the mSlideActivate message to the Slide In/Out behavior. This activates the behavior and causes the rocket to appear to fly off the top of the frame.

As you can now see, Director's built-in behaviors provide a very powerful way to add interactivity to your Director movies using very little or no Lingo. Although you've worked with only a few of Director's built-in behaviors (so many behaviors are shipped with the application that a whole book could be written on this subject alone), you should have a good idea of how to use behaviors in your movies.

It's worth taking some time on your own to familiarize yourself with the behaviors found in the Library Palette. It's an investment that could save you untold hours of movie-creation time. We've provided more information about Director's built-in behaviors in Appendix B. This appendix has descriptions of all the behaviors found in the Library Palette.

Summary

This chapter taught you how to add interactivity to your Director movies using behaviors that enable you to create sophisticated buttons to navigate your movies, as well as to create animated special effects. Before moving on to the next chapter to learn about the various ways the movies that you created in Director can be output, review some of the important features about behaviors that were discussed in this chapter:

- ✦ Director ships with a variety of useful behaviors that are contained in the Library Palette.
- ✦ Behaviors are a collection of event handlers, and each handler contains one or more actions. The Behavior Inspector translates these behaviors, events, and actions into scripts, handlers, and statements.

- Behaviors contain parameters that you can set to control the way a behavior will work, as well as how a behavior is activated.
- Messages can be sent to some behaviors that then can activate them. Several behaviors are also capable of sending messages that can be interpreted by Lingo scripts and other behaviors.

In Chapter 8, you learn how to fine-tune and then output your movies so that they can be played back without having to use Director.

✥ ✥ ✥