Chapter 3: Behaviors

Introduction to behaviors

This section describes how you can create LiveMotion 1.0 behaviors in LiveMotion 2.0. It is meant to help you move on to a new way of looking at what behaviors really are.

In LiveMotion 1.0, behaviors did everything from playing or stopping a composition to downloading a movie clip. Traditionally, behaviors executed when either a movie clip reached a certain point on its timeline or when a movie clip entered a certain state. In LiveMotion 2.0, behaviors have evolved into Player scripting code. To assist you in your transition to writing scripts, this section explains where you can add Player scripts and the implications of adding the scripts in these locations. It provides an overview of how to add, open, and remove scripts. Then for each LiveMotion 1.0 behavior, the section provides a procedure for implementing it in LiveMotion 2.0. As additional help, you are provided guidance using the Scripting Syntax Helper to access the LiveMotion 1.0 behaviors and the LiveMotion 2.0 code to which each behavior maps.

Even if you are new to LiveMotion, it will benefit you to read this section to learn how LiveMotion 1.0 behaviors are implemented in Player scripting, because you can incorporate their functionality into any Player scripts that you write. You are not required to know anything about LiveMotion 1.0 behaviors to create the examples in this chapter, which can instead serve as simple examples to start you down the road to Player scripting

Working with Player scripts that replace behaviors

This section provides procedures for adding, opening, and deleting scripts.

Note: *In LiveMotion 2.0, you also can write scripts to handle events. Event handling was made possible in LiveMotion 2.0 because of its support for Player scripting. For details on creating event handlers, see* *"Movie Clip Events and Event Handlers" on page 79.*

The effect of writing scripts to movie clip timelines versus movie clip states

You can write Player scripts to movie clip timelines or to movie clip states, depending on the effect that you are after. To prepare you for working with Player scripts, you should understand these concepts:

- Timelines have *script keyframes* (that is, script icons on timeline frames)
- States have timelines

When you write a script to a movie clip timeline, you write that script to a specific timeline frame. The frame is called a script keyframe. During execution in of the `.liv` file in Preview mode or on export of the SWF file to a browser, the script keyframe executes at a predictable point in the lifetime of the move clip, and that is, when the playhead reaches that script keyframe. A timeline can have multiple script keyframes, and each script keyframe can have one to several scripts written to it.

All objects have a normal state by default. You also can define states for a movie clip (such as over, down, out) in the States palette. Each movie clip state is provided with its own independent timeline. When you write a script to a state that you define, it executes only when the user activates that state, not at a predefined point in a movie clip's lifetime. Say, for example, the user presses the mouse on a movie clip for which you have defined a down state. This would activate any script you may have written for that state. You can write scripts to any or all states that you define for a movie clip. You also can write multiple scripts to the timeline of a single defined state.

Accessing Player scripts

You can access Player scripts from:

- *Script keyframes* (that is, script icons on frames) in a timeline. Clicking the script keyframe opens the Script Editor and displays the script attached to that frame on the timeline.
- The *Scripts button* towards the bottom of the States palette. Clicking the scripts button opens the Script Editor on the state currently selected in the States palette.

In LiveMotion 1.0, the Scripts button was called the Behaviors button. For your general reference, Figure 3.1 shows the LiveMotion 1.0 Behaviors button in a timeline and in the Rollovers palette. It also shows a keyframe in a timeline.

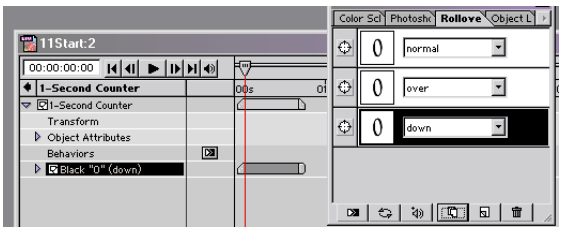

Figure 3.1 LiveMotion 1.0 Timeline window and Rollovers palette

Figure 3.2 shows the corresponding LiveMotion 2.0 Scripts button in a timeline and in the States palette. It also shows a script keyframe in a timeline.

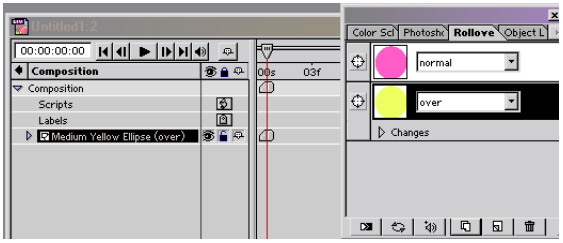

Figure 3.2 LiveMotion 2.0 Timeline window and States palette

(Advanced users) You can also access Player scripts by selecting Scripts>Editor from the main menu. Alternately, you can use the keyboard shortcut Ctrl + J (Windows) or Command + J (Mac OS). Then, double click the movie clip whose script you want to access in the Script Editor's Navigator. This takes you to that movie clip's scripts, but not necessarily to the script that you want. You must then navigate to the event, state, or script keyframe containing the script you want to access. For details on using the Navigator, see "Scripting Tools" on page 91.

Adding Scripts

To add a script to a movie clip state,

Note: *The first three steps to add a script to a movie clip state also open a script on a state. Compare steps 1 to 3 below to the procedure in "To open a script from a movie clip state," on page 47.*

1 Open the States palette to view the movie clip states.

2 In the States palette, select the movie clip state to which you want to add a script.

3 Click on the Scripts button in the palette. See Figure 3.2. This opens the Script Editor and displays the window in which you can add a script that will be attached to that movie clip state

4 Click the Scripting Syntax Helper button to open the list of LM 1.0 behaviors. Choose the desired script by its LM 1.0 behavior name, and click OK. The script for the behavior is added to the Script Editor window, as shown in Figure 3.3. For details on the Scripting Syntax Helper, see "Scripting Tools" on page 91.

5 Replace any parameters in the script with their required values.

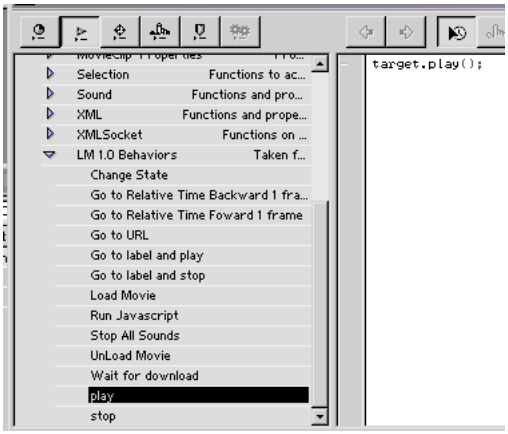

Figure 3.3 Scripting Syntax Helper open to LM 1.0 behaviors with the play behavior selected

To add a script to a movie clip timeline,

1 Navigate to the timeline where you want to add the script keyframe.

2 In the Timeline window, move the current-time marker to the frame to which you want to add a script. Optionally, click the Labels button (see Figure 3.2), and enter a name for the point in time where the script will be attached to the timeline.

3 Click the Scripts button on the timeline to create a script keyframe at the current-time marker and to open the Script Editor.

Note: *If a script keyframe already exists on the specified frame, clicking the Scripts button simply opens the Script Editor.*

4 Click the Scripting Syntax Helper button to open the list of LM 1.0 behaviors. Choose the desired behavior by its LM 1.0 name, and click OK. The script for the behavior is added to the Script Editor window.

5 Replace any parameters with their required values.

Opening scripts

To open a script from a movie clip state,

1 Open the States palette to view movie clip states.

2 In the States palette, select the movie clip state with the script you want to open.

3 Click the Scripts button in the palette. This brings up the Script Editor and displays the script for that movie clip state in the Script Editor window.

To open a script from the timeline,

1 Locate the script icon for the script you want to view, and double-click.

Deleting scripts

To delete a script from a movie clip state,

1 Open the States palette to view movie clip states.

2 In the States palette, select the movie clip state with the script you want to delete.

3 Click the Scripts button in the palette. This brings up the Script Editor and displays the script for that movie clip state in the Script Editor window.

4 Select the script implementing the behavior you want to delete, and press Delete.

To delete a script from the timeline,

1 Locate the script icon for the script you want to view, and double-click.

2 Select the script implementing the behavior you want to delete, and press Delete.

How to create the LiveMotion 1.0 behaviors using LiveMotion 2.0 scripts

This section provides details on how you create scripts for LiveMotion 1.0 behaviors. For your reference, Table 3.1 lists the LiveMotion 1.0 behaviors supported and the LiveMotion 2.0 scripts to which they map.

Table 3.1 LiveMotion 1.0 Behaviors and their corresponding Player scripts

LM 1.0 Behavior	Player script	Description
Change State	`movieClip.lmSetCurrentState(state);`	Change the state of the specified movie
Go to Relative Time Backward 1 frame	`movieClip.prevFrame();`	Go to the movie clip's relative time backward 1 frame
Go to Relative Time Forward 1 frame	`movieClip.nextFrame();`	Go to the movie clip's relative time forward 1 frame
Go to URL	`getURL(URL,target);`	Open a URL in the specified browser window or frame
Go to Label	`movieClip.gotoAndStop(label);`	Go to the specified label and stop
Go to label and play	`movieClip.gotoAndPlay(label);`	Go to the specified label and play
Load Movie	`loadMovieNum(URL,number);`	Load the specified URL into the specified SWF file level
Run JavaScript	`getURL(javascript:string);`	Run the javascript specified
Stop All Sounds	`stopAllSounds();`	Stop all sounds from playing but do not stop the movie
Unload Movie	`unloadMovieNum(number);`	Unload the specified movie

LM 1.0 Behavior	Player script	Description
Wait for download	`if (_framesloaded < lmFrameOfLabel(label)) { movieClip.gotoAndPlay(label); }`	Loop on a certain label until something is loaded
play	`movieClip.play();`	Start playing the specified movie
stop	`movieClip.stop();`	Stop playing the specified movie

Creating Change State scripts

The Change State script changes the state of the specified movie clip.

To change the state of a movie clip,

1 Navigate to the location where you want to add the state change. See "Adding Scripts" on page 45.

2 In the Script Editor, click the Scripting Syntax Helper button. Select Change State from the LM 1.0 behaviors list. The appropriate Player script appears in the Script Editor window:

```
movieClip.lmSetCurrentState(state);
```

3 Replace the arguments described below with the appropriate values. You can use the Composition Browser in the Script Editor to help fill in the values. For details on using the Script Editor features, see "Scripting Tools" on page 91.

movieClip is the movie clip whose state you want to change.

state is a string; the name of the state you want to set.

Creating scripts to manipulate a movie clip timeline

Four scripts can be used to manipulate a timeline. These are:

- Play
- Stop

- Go To Relative Time
- Go To Label
- Go to label and play

The Play and Stop scripts play or stop a specified timeline.You can, for example, add scripts to the first frame of a composition timeline to stop the timelines of all the movie clips it contains. Although the movie clip timelines will be stopped, the composition timeline will continue playing, enabling you to run individual movie clips as needed using the Play script.

In LiveMotion 2.0, the Go To Relative Time scripts only support going forward or backward one frame; whereas, the LiveMotion 1.0 behavior supported going forward or backward a specified number of frames. To achieve the same result as Go To Relative Time in LiveMotion 1.0, however, you can use the Go To Label script.

The Go to Label script moves the animation to a specific label in a timeline and stops the timeline.

The Go to label and play script sends the playhead of a movie clip's timeline to the specified frame or label to play the timeline at that frame.

To add a Play or Stop script,

1 Navigate to the location where you want to add the script. See "Adding Scripts" on page 45.

2 In the Script Editor, click the Scripting Syntax Helper button. Select Stop or Play from the LM 1.0 behaviors list. The appropriate Player script appears in the Script Editor window:

```
movieClip.stop();
```

or

```
movieClip.play();
```

3 Replace the *movieClip* argument described below with the appropriate value. You can use the Composition Browser in the Script Editor to help fill in this value. For details on using the Script Editor features, see "Scripting Tools" on page 91.

movieClip is the movie clip you want to start or stop at it's current frame. If the movie clip is stopping or playing itself, use `this` for the movie clip, for example,

```
this.stop();
```

or

```
this.play();
```

`play()` and `stop()` are movie clip methods that are equivalent in functionality to the respective LiveMotion 1.0 Play and Stop behaviors.

To add a Go to Relative Time script,

1 Navigate to the location where you want to add the script. See "Adding Scripts" on page 45.

2 Click the Scripting Syntax Helper button. Select Go to Relative Time Backward 1 frame or Go to Relative Time Forward 1 frame from the LM 1.0 behaviors list. The appropriate Player script appears in the Script Editor Composition window:

```
movieClip.prevFrame();
```

or

```
movieClip.nextFrame();
```

3 Replace the *movieClip* argument described below with the appropriate value. You can use the Composition Browser in the Script Editor to help fill in this value. For details on using the Script Editor features, see "Scripting Tools" on page 91.

movieClip is the movie clip you want to move backward or forward 1 frame.

To add a Go to Label script,

1 Navigate to the location where you want to add the script. See "Adding Scripts" on page 45.

2 Click the Scripting Syntax Helper button. Select Go to Label from the LM 1.0 behaviors list. The Player script appears in the Script Editor Composition window:

```
movieClip.gotoAndStop(label);
```

Replace the *movieClip* and *label* arguments described below with the appropriate values. You can use the Composition Browser in the Script Editor to help fill in these values. For details on using the Script Editor features, see "Scripting Tools" on page 91.

movieClip is the name of the movie clip that you want to go to *label* and stop.

label is a string associated with the frame on the movie clip's timeline to which the playhead will be sent and stopped.

Here is an example script with the values filled in:

```
_root.gotoAndStop("label1");
```

To add a Go to label and play script,

1 Navigate to the location where you want to add the script. See "Adding Scripts" on page 45.

2 Click the ActionScript browser button. Select Go to and play from the LM 1.0 behaviors list. The Player script appears in the Script Editor Composition window:

```
movieClip.gotoAndPlay(label)
```

3 Replace the *movieClip* and *label* arguments described below with the appropriate values. You can use the Composition Browser in the Script Editor to help fill in these values. For details on using the Script Editor features, see "Scripting Tools" on page 91.

movieClip is the name of the movie clip that you want to go to *label* and play.
label is a string associated with the frame on the movie clip's timeline to which the playhead will be sent to play.

Creating a Wait for download script

The Wait for Download script is a special case of timeline manipulation. It creates a looping animation that repeats until the specified object has been downloaded. This prevents poor performance for movie clips that include large objects, or for lengthy and complex movie clips. This script is used to download SWF files for playing; it is not used for downloading files in the broader sense of saving files to disk.

To wait for the movie clip to download,

1 Navigate to the location where you want to add the Wait for Download script. See "Adding Scripts" on page 45.

2 Click the Scripting Syntax Helper button. Select Load Movie from the LM 1.0 behaviors list. The Player script appears in the Script Editor Composition window:

```
if (_framesloaded < lmFrameOfLabel(label)
{
   movieClip.gotoAndPlay(label);
}
```

lmFrameOfLabel() is a global function that sets a label on the timeline indicating the number of frames that need to be downloaded before the movie clip begins.

3 Replace the *label* , *movieClip* , and *_framesloaded* arguments described below with the appropriate values. You can use the Composition Browser in the Script Editor to help fill in the values for *label* and *movieClip* . For details on using the Script Editor features, see "Scripting Tools" on page 91.

label is a string associated with the frame on the movie clip's timeline to which the playhead will be sent to play.

movieClip is the the name of the movie clip that will loop on the label.

_framesloaded is the number of movie clip frames that have been loaded.

The movie clip plays until the number of frames loaded is equal to or greater than the frame number upon which the label is located.

Creating scripts to command the Flash Player

Three scripts create commands to the Flash Player. These are:

- Load Movie
- Unload Movie
- Stop All sounds

Load Movie loads and plays a Flash format (SWF) file that can either replace the existing animation, or play a layer on top of the existing animation. Unload Movie removes an already-loaded SWF file from the screen. Stop All sounds stops all sounds in the Player, including event sounds.

To load a movie,

1 Navigate to the location where you want to add the script to load a movie. See "Adding Scripts" on page 45.

2 Click the Scripting Syntax Helper button. Select Load Movie from the LM 1.0 Behaviors list. The behavior script appears in the Script Editor Composition window:

```
loadMovieNum(URL ,number);
```

3 Replace the arguments described below with the appropriate values.

URL is the absolute or relative reference to the external SWF file. These are examples:

```
http://www.mydomain.com/loadedMovie.swf
```

or

```
loadedMovie.swf
```

number is a non-negative integer specifying the document level into which the SWF file will be loaded. Your default composition is considered to be level number 0. If the level already contains a SWF file, it is replaced by the one being loaded. For details on document level, see "Levels of SWF files" on page 77.

To unload a movie,

1 Navigate to the location where you want to add the script to unload a movie. See "Adding Scripts" on page 45.

2 Click the Scripting Syntax Helper button. Select Unload Movie from the LM 1.0 Behaviors list. The behavior script appears in the Script Editor Composition window:

```
unloadMovieNum(number);
```

3 Replace the argument described below with the appropriate value.

number is a non-negative integer specifying the document level of the SWF file to be unloaded. For details on document levels, see "Levels of SWF files" on page 77.

To stop all sounds,

1 Navigate to the location where you want to add the script to stop all sounds. See "Adding Scripts" on page 45.

2 Click the Scripting Syntax Helper button. Select Stop All Sounds from the LM 1.0 behaviors list. The Player script appears in the Script Editor Composition window:

```
stopAllSounds();
```

Creating scripts to control the web browser

There are two browser command scripts. These are:

- Run JavaScript
- Go to URL

Run JavaScript executes JavaScript code in the user's browser. The Go to URL script opens a specified URL in the user's browser and loads it into the browser at the specified target.

To run JavaScript,

1 Navigate to the location where you want to add the script to execute JavaScript. See "Adding Scripts" on page 45.

2 Click the Scripting Syntax Helper button. Select Run JavaScript from the LM 1.0 behaviors list. The Player script appears in the Script Editor Composition window:

```
getURL(javascript:string);
```

3 Replace the *string* argument with the appropriate value , as illustrated by the example below:

The *string* argument begins with:

```
javascript:
```

This is followed by the code. Here is a complete example:

```
getURL("javascript: window.alert('hello world');");
```

This code displays the string 'hello world' in the browser window.

To add a Go to URL script,

1 Navigate to the location where you want to add the Go to URL script. See "Adding Scripts" on page 45.

2 Click the Scripting Syntax Helper button. Select Go to URL from the LM 1.0 behaviors list. The Player script appears in the Script Editor Composition window:

```
getURL(URL,movieClip);
```

3 Replace the *movieClip* argument described below with the appropriate value. You can use the Composition Browser in the Script Editor to help fill in these values. For details on using the Script Editor features, see "Scripting Tools" on page 91.

URL is the URL to which you want to link.

movieClip is the desired HTML frame to load or a standard HTML frame type.

Here is an example:

```
getURL("http://www.adobe.com", _parent);
```