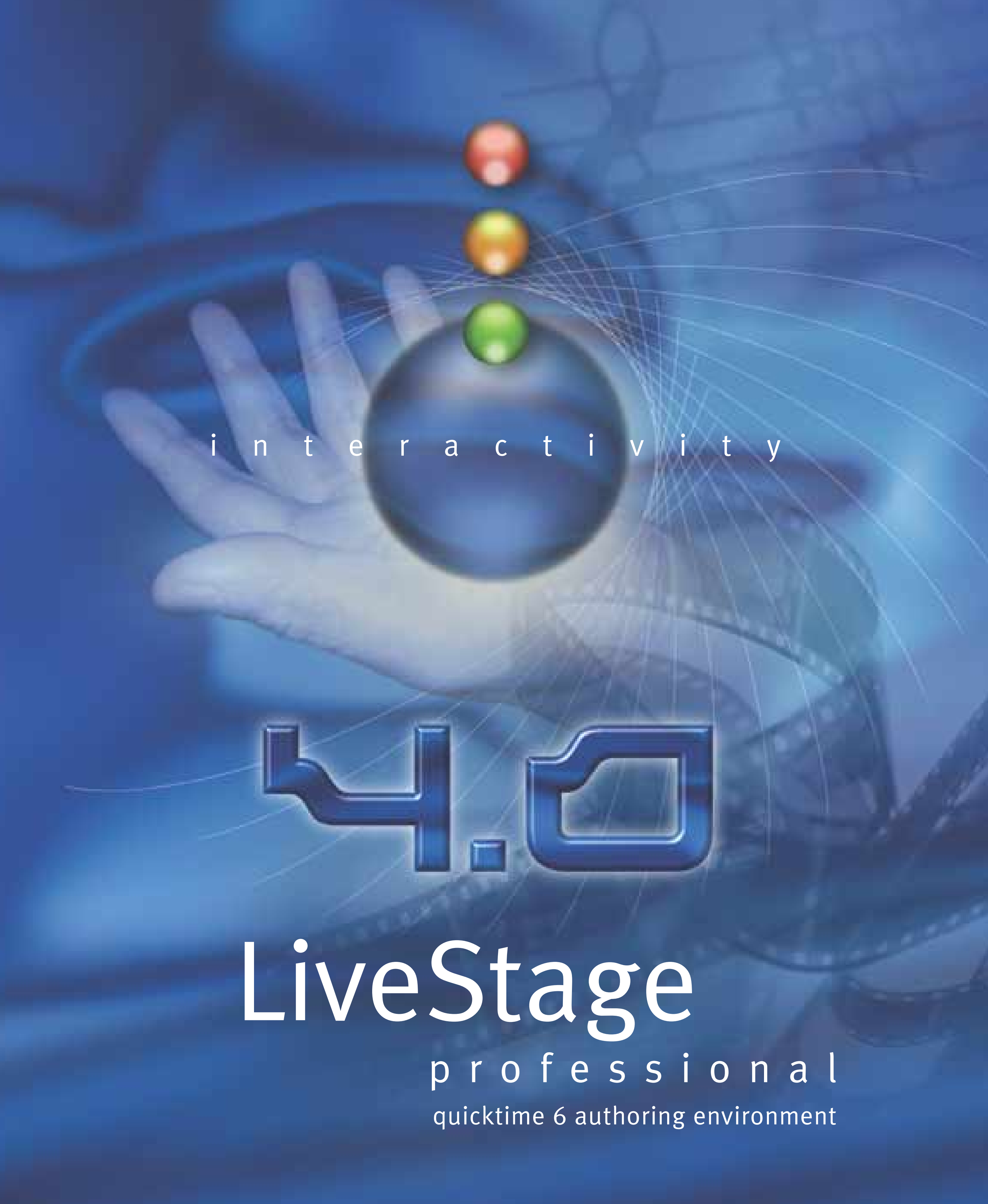


interactivity

4.0

LiveStage

professional

quicktime 6 authoring environment

Contents

Intro

Introduction

WELCOME

Welcome to LiveStage Professional 4.0, the premier QuickTime authoring tool for building innovative and compelling interactive media. LiveStage Professional 4.0 is a powerful cross-platform production application that fully accesses the extensive QuickTime architecture and allows you to take advantage of the many features QuickTime supports. This architecture enables you to incorporate over 200 different media types and formats including MPEG- 4 to build dynamic interactive media projects for delivery on the Web, CD-ROM, or any other QuickTime-enabled device. LiveStage Professional is designed to support continuing advances in QuickTime technology, so you can stay on the leading edge as interactivity becomes more sophisticated and the demands for flexibility and functionality increase.

In this chapter, you will find a summary of the new features added to LiveStage Professional 4.0. You will also find installation and system requirement information, as well as a list of the various resources in place to support your authoring at all stages of development.

WHAT'S NEW IN LIVESTAGE PROFESSIONAL 4.0

Significant improvements have been made to the user interface and workflow that increase productivity, encourage creativity, and make LiveStage Professional easier to use. The user interface has been completely revamped with a great new look and feel, designed to give video and multi-media professionals an environment they can quickly and easily grasp. Among the many enhancements in LiveStage Professional 4.0, you will find:

- Four new FastTracks™ to simplify the production of common projects
- Contextual menus
- A wide variety of keyboard commands
- An enhanced Undo feature with up to 25 levels
- In-place editing of element names
- Improved error reporting
- Full AppleScript support for Mac developers
- A Media Synchronizer

THE DOCUMENT WINDOW

What were once separate windows, have now been united to form the Document Window. The Stage and the Timeline have been merged into a single window, to better work in conjunction with one another. A simple click changes the window to reflect just the Stage or just the Timeline. You can also adjust the sizing of each pane to suit your needs and even save multiple layout versions to accommodate specific tasks. With a host of time saving toolbars and keyboard commands for creating common track types, the new Document Window offers an improved working environment designed to greatly enhance your productivity (see Chapter 3—Working in the LiveStage Professional Environment).

STAGE AREA

LiveStage Professional 4.0 has graduated the Stage to a powerful layout environment. New features include:

Toolbars

A variety of comprehensive tools make it a snap to skew, rotate and stretch sprites, resize and align tracks—even create new ones.

Layout Tools

A fixed or dynamic background, exact positioning guides and rulers for placing objects relative to the stage or each other, enhanced zoom functions and live snapping provide increased control and flexibility in building your projects visually.

Improved Object Control

Increase your productivity by grouping multiple tracks and manipulating them all at once. In-place drag-and-drop menus support greater control in sample editing, and track layering can now be changed directly on the Stage.

Accurate Movie Representation

The visual representation of the final movie has improved significantly. Elements such as Alpha Channels, Flash media, and Skins are now correctly rendered on the Stage.

TIMELINE AREA

A host of new features has been added to the timeline to enhance precision and workflow. The timeline now displays a track or sample's time code dynamically as you drag. Tools for creating Poster Frames and Preview Movies and an In/Out Point Editor for external media have been implemented as well. Creating Chapter tracks and synchronizing media have also been made easy with LiveStage Professional 4.0's new marker feature.

Track Functionality

Track functionality has been reworked to provide increased control. With the Group Track, you can organize tracks in your individual projects, reduce window clutter, and perform operations on multiple tracks simultaneously. The addition of a Track Manipulation Tool makes it easy to set the duration and offset of a track to precise points in the timeline. Finally, track ordering is now in the hands of the developer. Select Layering, Loading, or Custom View to work with your tracks according to a variety of objectives.

Sample Control

Sample management is approaching the refinement of non-linear video editing programs through the addition of new tools that allow for creating, deleting, splitting, expanding and stretching of individual or multiple samples. Dragged samples are positioned dynamically and automatically snap to significant times or markers. You can also manipulate multiple samples within multiple tracks simultaneously.

INSPECTOR

A brand new addition to the LiveStage Professional interface, the Inspector brings the Movie and Track Properties out of hiding. Now, by simply selecting the movie or track, you can access and adjust its most common properties without opening a new window (see Chapter 3—Working in the LiveStage Professional Environment).

QSCRIPT REFERENCE

The QScript Reference window has become a powerful ally of LiveStage Professional developers of all skill levels. The QScript Reference, now easily accessible from any Script Editor Window, contains a powerful search function and provides a more in-depth explanation of each script—including examples. It even has links to web content for more information (see Chapter 16—Introduction to QScript).

FASTTRACKS™

Four new “FastTracks™” allow you to add Playback or VR Controls, Media Skins or a Status Bar to your movies instantly, with easy to use drag-and-drop editors (see Chapter 2—FastTracks™).

MEDIA SYNCHRONIZER

Synchronize your Video and Sound Tracks with Text and Picture Tracks easily with the new Media Synchronizer. Drag and drop functionality, and quick and easy in/out point setting synchronize your media quickly and precisely (see Chapter 14—Working with Multiple Tracks).

VIDEO TRACK

Tween controls now allow you to adjust the video’s opacity within the Timeline. You can also dynamically set Markers in your Timeline during video playback (see Chapter 8—External Tracks).

AUDIO TRACK

You can now view your Audio Track’s waveform and adjust its volume through the new volume tween right in the timeline. In addition, the Track Properties Window and Inspector both house volume and balance controls (see Chapter 8—External Tracks).

TEXT TRACK

By setting markers in your Timeline, you can now automatically create Chapter Tracks at precise points in your movie. You can now also create Text Tracks from QTText files, which allow you more control over formatting and timing—a tremendous aid for captioning (see Chapter 6—The Text Track).

SKIN TRACK

The Skin Track automatically sizes itself to fit both your movie's physical size and its duration. It also comes complete with a revamped Editor. Not only can you set your Drag and Window areas, but you can also automatically add pre-scripted Close and Full Screen buttons (see Chapter 2—FastTracks™).

SPRITE TRACK

Workflow in the busiest track has also been enhanced. Creating a new Sprite is now as simple as dragging an image into the Sprite List. The Script Editor has also been improved. A new Sprite property indicates if the Sprite is clickable (see Chapter 7—The Sprite Track).

VR TRACK

Now you can set the Pan Angle, Tilt Angle, and the Field Of View for a node you can access through a HotSpot (see Chapter 10—The VR Track).

FLASH TRACK

Flash 5 support has been added and you can now also view, open and script Movie Clips. Additionally, you can now Reedit buttons in your Flash file without fear of losing your button scripting in LiveStage Professional (see Chapter 9—The Flash Track).

MOVIE TRACK

Creating Movie Tracks has never been easier. Now you can enter multiple movie URLs and reorder Child Movies with a simple drag-and-drop. LiveStage Professional now recalls previously entered URLs and you can preview your linked movies right in the Movie Sample Editor (see Chapter 10—The Movie Track).

COLOR TRACKS

A much improved user interface for editing gradients has been added to the Color Track, allowing you to specify linear gradients, circular gradients, or solid colors (see Chapter 13—Additional LiveStage Professional Tracks).

PICTURE TRACK

Sophisticated drag-and-drop sample editing makes replacing and adding individual images directly into the timeline easy. The sample editor now offers more control over compression quality (see Chapter 5—The Picture Track).

BEHAVIORS

Behaviors can now be reordered in the Sprite Editor's Behavior tab by simply dragging and dropping. In addition, improved error reporting specifies which Behavior caused the error. This comes in handy when you are working with custom Behaviors you have created (see Chapter 17—Behaviors).

About QuickTime

QuickTime is much more than just a video and audio player. It is a true multimedia architecture that allows the integration of text, graphics, video, audio, Flash, 3D, VR, and skins, while maintaining original file integrity. QuickTime's extensive architecture provides a wealth of functionality and possibilities for interactive multimedia development. As LiveStage Professional is based on and completely embraces the QuickTime platform, an understanding of QuickTime will greatly enhance the quality of your presentations and the construction of your projects. You will find pre-production tips related to QuickTime throughout this user guide, but for additional information and resources, visit http://www.apple.com/quicktime.

About the User Guide

The LiveStage Professional 4.0 User Guide is somewhat of a departure from prior editions. This guide will offer the same high level of technical expertise, delivered in a manner that is designed to provide a readable and comprehendible experience to developers—new and seasoned alike. Each chapter is designed to be a standalone reference so you do not have to read the entire User Guide to access the specific information you need. Not only are definitions and overviews provided, but the User Guide also includes brief step by step How-To's, "hip" Tips, and real-world examples of when and why you would want to employ each function and/or feature.

The User Guide starts with basics and progresses to details. Chapters earlier in the User Guide will provide you with broad rudimentary information that can help introduce you to, or refresh your memory on, LiveStage Professional basics. As you proceed through the User Guide, the information will become more detailed and specific. Finally, the Appendices in the User Guide will provide you with reference materials on subjects ranging from Draw Modes to QScript.

LIVESTAGE PROFESSIONAL WORK AREA

This section provides a basic overview of the different interface elements you will encounter while working in the LiveStage Professional authoring environment.

WORKING IN THE LIVESTAGE PROFESSIONAL ENVIRONMENT

This chapter offers a breakdown of major elements and tools you will use.

INTRODUCTION TO TRACKS

The Introduction to Tracks chapter begins to delve into the common features and processes involved with these building blocks of your LiveStage Professional projects.

TRACK CHAPTERS

Finally, the specifics of working with each track type are detailed in individual chapters.

Contents of the LiveStage Professional 4.0 CD-ROM

The LiveStage Professional 4.0 CD-ROM contains the following items:

- Read Me
- License Agreement
- Release Notes
- LiveStage Professional Installer
- Documentation folder
- Project Samples folder
- Behavior Samples folder
- Tutorials folder
- Totally Hip Demos folder
- Extra Library Media folder
- Showcase Movies folder
- AppleScripts folder
- 3rd Party Tools folder
- Acrobat folder
- CarbonLib folder
- QuickTime folder

See the “Read Me” file on the CD-ROM for detailed information on each item.

Beyond the User Guide

In addition to the information, tutorial and step-by-step instructions found in this User Guide, there are several means of advancing your LiveStage Professional 4.0 authoring skills.

LIVESTAGE PROFESSIONAL TALK LIST

This active and informative discussion list is made up of a wide range of LiveStage Professional developers, from newcomers to experts. The list offers valuable insight into specific project-related issues as well as general tips, tricks and concepts involved in LiveStage Professional authoring. To subscribe to this ongoing wealth of information, visit www.totallyhip.com/permanent/support.html.

STAGEDOOR

StageDoor (http://stagedoor.totallyhip.com) is a LiveStage Professional and QuickTime showcase site that offers featured real-world projects, interviews with developers, and hands-on tutorials, along with an active and growing discussion forum. StageDoor is a great place to go for both information and inspiration as you work through your own creative interactive projects.

LSDN

The LiveStage Developer Network (LSDN) at http://www.totallyhip.com/lsdn is an indispensable resource for finding information. For tools, scripts, sample projects, mailing list archives and technical information, LSDN provides an extensive repository of resources supporting the LiveStage Professional community.

CERTIFIED INSTRUCTORS

LiveStage Professional authoring is offered as part of a new media curriculum at a number of post-secondary and multimedia institutions throughout North America. Instructors at these facilities have been certified through the LiveStage Professional Educational Development program. Contact individual schools for more information:

- Vancouver Film School—Vancouver, BC
- Sheridan College—Toronto, ON
- Savannah College of Art and Design—Savannah, GA
- University of Michigan College of Communication Arts and Science—Ann Arbor, MI
- Palomar Community College—San Marcos, CA

WORKSHOPS

Hands-on authoring workshops are held throughout the year across North America. These intensive two-day sessions, taught by Totally Hip as well as third party instructors, accelerate the LiveStage Professional learning curve through low student to teacher ratios and comprehensive supporting materials. For more information, visit http://www.totallyhip.com.

LIVESTAGE PROFESSIONAL AUTHORING WORKBOOK

For developers who prefer self-study, Totally Hip also offers the Workbook and CD-ROM that provides the foundation of the hands-on courses. This illustrated User Guide covers everything from LiveStage Professional basics to intermediate authoring skills in a series of exercises, lessons, and self-test sections. All required media is provided along with sample projects on the associated CD-ROM. Visit http://www.totallyhip.com or contact Totally Hip directly for more information.

System Requirements

MAC

- Power Macintosh G3 333MHz or faster
- Mac OS 9.1 or OSX 10.1.5 or greater
- QuickTime 6.0 or greater (Full Install)
- CarbonLib 1.5 or greater (Run Apple System Profiler to determine your CarbonLib version)
- 20 MB of application RAM

- 15 MB of hard disk space for basic installation
- Adobe Acrobat Reader 4.0 for PDF document reading

WINDOWS

- Pentium II 400 Mhz or faster
- Windows 98/NT4/XP/2000
- QuickTime 6.0 or later (Full Install)
- 64 RAM minimum
- 15 MB of hard disk space for basic installation
- Adobe Acrobat Reader 4.0 for PDF document reading

Before Installing LiveStage Professional

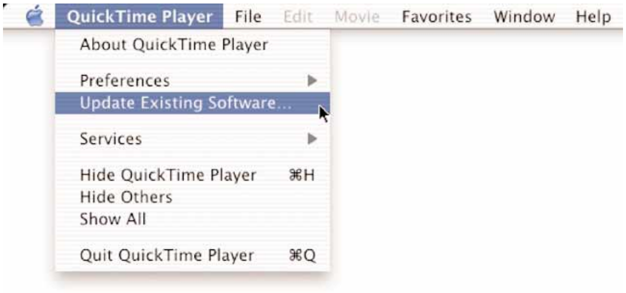
Before you install LiveStage Professional, you will need to have QuickTime installed on your hard disk. The Full Install of QuickTime 6.0 or later is a required component for LiveStage Professional. The LiveStage Professional installer will notify you if you do not have the correct version of QuickTime installed. You will need to install the correct version prior to running LiveStage Professional 4.0. In the event that you do not have software installed that will allow you to read PDF documents, an installer for Adobe Acrobat has also been provided. This will allow you to view electronic documentation that has been provided with LiveStage Professional.

DETERMINING IF YOU HAVE A FULL INSTALL OF QUICKTIME

If you already have QuickTime installed, make sure that you have a Full Install.

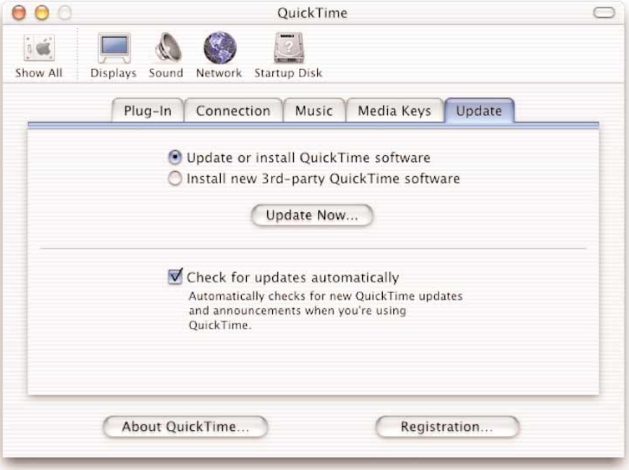
1. Launch the QuickTime Player.

2. Under the QuickTime Player menu select Preferences > QuickTime Preferences.



3. Select Update.

4. Select “Update or install QuickTime Software” and click Update Now...



● Update or install QuickTime software
○ Install new 3rd-party QuickTime software

☑ Check for updates automatically
Automatically checks for new QuickTime updates and announcements when you're using QuickTime.

5. Follow the instructions in the Updater

6. Repeat these steps to “Install Additional QuickTime Software” as well.

INSTALLING QUICKTIME

The QuickTime installer is provided for you on the LiveStage Professional CD-ROM.

1. Insert the LiveStage Professional CD-ROM into your CD drive.

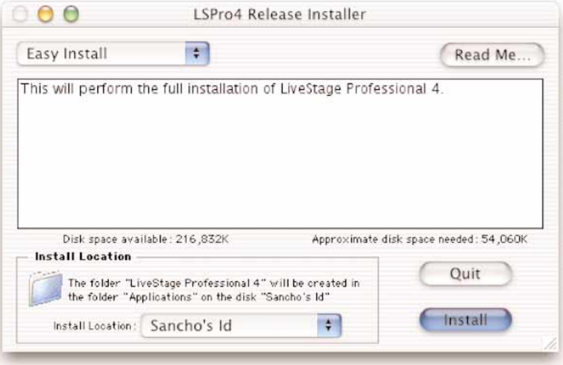
2. Double-click the LiveStage Professional CD icon on your desktop.

3. Open the ‘QuickTime’ folder.

4. Double-click the ‘QuickTime Installer’ icon.

5. Follow the installation instructions on your screen and select Full Install.

6. After completing the installation, restart your computer.



Easy Install

This will perform the full installation of LiveStage Professional 4.

Disk space available: 216,832K
Approximate disk space needed: 54,060K

Install Location

The folder "LiveStage Professional 4" will be created in the folder "Applications" on the disk "Sancho's Id"

Install Location: Sancho's Id

INSTALLING ADOBE ACROBAT READER

1. On the LiveStage Professional CD-ROM, open the Acrobat folder.

2. Double-click 'Acrobat Reader Installer' to launch it.

3. Follow the installation instructions as they appear on the screen.

Installing LiveStage Professional on Macintosh Systems

CD INSTALLATION:

1. Double-click the LiveStage Pro Installer application icon.

2. Follow the installation instructions provided.

3. The installer will check for QuickTime, Mac OS version, and CarbonLib (if you are running OS 9). If you do not have the correct version of Carbon Lib or QuickTime, it will notify you and you can install them prior to running LiveStage Professional. If you do not have the proper Mac OS version the installer will quit.

For Windows installation instructions, please review the "Read Me" file on the LiveStage Professional 4.0 CD-ROM.

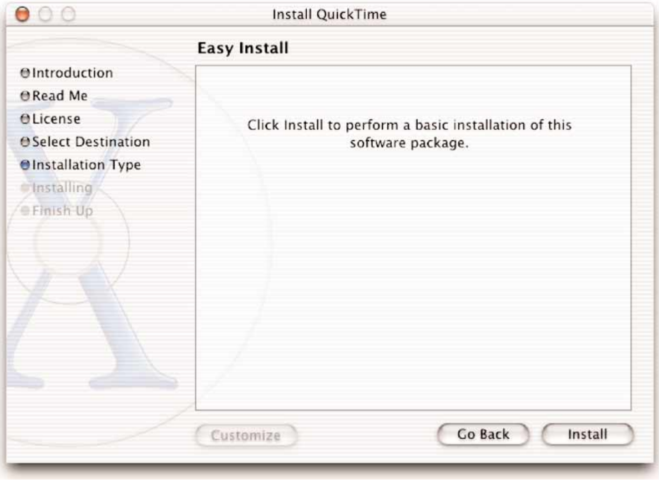
ONLINE INSTALLATION:

1 Double-click the installation icon

2. Follow the instructions provided.

3. The installer will check for QuickTime, Mac OS version, and CarbonLib (if you are running OS 9). If you do not have the correct version of Carbon Lib or QuickTime, it will notify you and you can install them prior to running LiveStage Professional. If you do not have the proper Mac OS version the installer will quit.

The first time LiveStage Professional is launched, you will be prompted to enter your name, company name (optional), and the product serial number. The serial number, included with the CD, must be entered exactly as shown, with all capital letters and no spaces.



Click Install to perform a basic installation of this software package.

REGISTRATION

Make sure to register your copy of LiveStage Professional to take advantage of upgrade pricing, technical support and other special savings. You can register directly within LiveStage Professional 4.0 under the Help Menu or you can visit www.totallyhip.com/permanent/register.html to register online.

Getting Technical Support

Technical assistance for LiveStage Professional is available for registered users via email at techsupport@totallyhip.com. For assistance with project deployment or development please visit StageDoor (http://stagedoor.totallyhip.com), LSDN (http://www.totallyhip.com/lsdn), or post your question or concern on the LiveStage Professional Talk List or StageDoor forums. Totally Hip also offers specialized consulting for short or long-term projects. Contact Totally Hip for details.

Installation Troubleshooting

WHERE IS THE SERIAL NUMBER?

The serial number is located on the CD-ROM Envelope. If there is no serial number present, contact Technical Support for assistance and have your purchase information available.

LIVESTAGE PROFESSIONAL DOES NOT ACCEPT MY SERIAL NUMBER

The serial number must be entered exactly as it is shown on the CD envelope—all capitals with no spaces.

Chapter 1

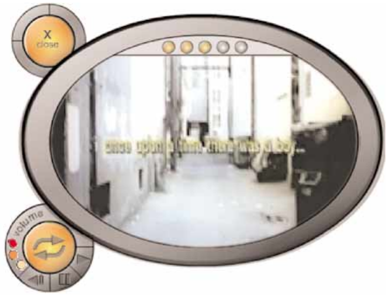
LiveStage Professional Tutorial

INTRODUCTION

This tutorial will introduce you to the basics of LiveStage Professional authoring. Through this hands-on exercise, you will become familiar with the fundamentals of working in LiveStage Professional as you construct an interactive skinned video player.

By the conclusion of this tutorial, you will know how to:

- Manage your project's media elements;
- Modify individual tracks;
- Frame a video with a graphic 'mask';
- Create interactive buttons;
- Incorporate a Flash controller;
- 'Skin' your movie;
- Export your project as a QuickTime movie.



This Tutorial should take approximately 30 minutes to complete.

Attention Windows Users: Keyboard shortcuts presented in this User Guide are Mac-based. Whenever you see "Command-" substitute "Control-" to achieve the same functionality.

GETTING STARTED

If you have not yet installed LiveStage Professional 4.0, refer to Installing LiveStage Professional on page 12.

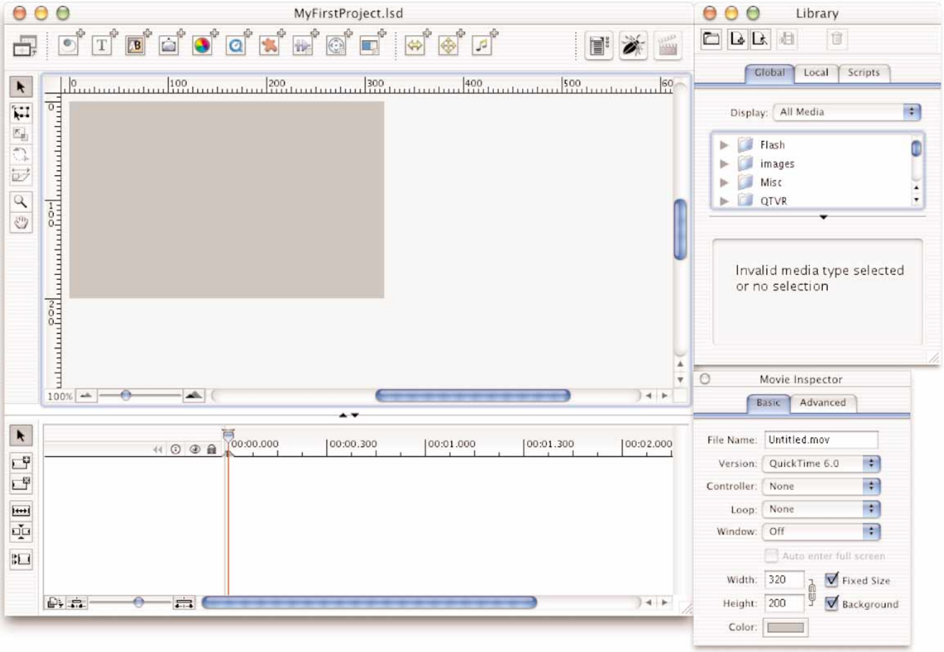
Prior to beginning this project, it is also recommended that you review the LiveStage Professional Work Area fold-out page located with the CD-ROM to become familiar with common elements of the LiveStage Professional interface.

1. Locate the Tutorial folder on the LiveStage Professional 4.0 CD-ROM and drag it to your desktop.
2. Double-click the LiveStage Professional 4.0 icon to launch the program.
3. Create a New Project.
4. From the File menu, select Save (Command-S).

   The Save dialog window opens.

5. Name the project ‘MyFirstProject.lsd’ and save it to your desktop.

The main window that appears is the Document Window. This window is divided into two sections: the Stage at the top and the Timeline below. Also visible are the Movie Inspector and the Library. If the Movie Inspector is not visible, type Command-I to view it; if the Library is not visible, type Command-Y to view it.



Display: All Media

File Name: Untitled.mov
Version: QuickTime 6.0
Controller: None
Loop: None
Window: Off
Width: 320 ☑ Fixed Size
Height: 200 ☑ Background

BEGINNING A PROJECT

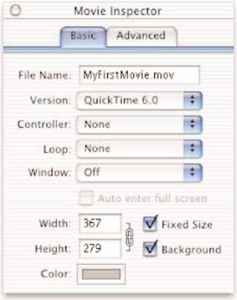
LiveStage Professional is a media integration tool, not a media creation tool. Thus, it should be used in the final stage before deploying your media. You will create your media in other applications, such as Photoshop(r), Flash(r), and Final Cut Pro(r) before bringing it into LiveStage Professional.

ASSIGNING MOVIE SETTINGS WITH THE MOVIE INSPECTOR

Projects built in LiveStage Professional produce QuickTime movies. A good first step in your workflow when you start a project is to set up the parameters for your final movie output. You can set the properties of the QuickTime movie your project will produce in the Movie Inspector.

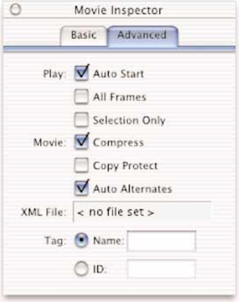
1. In the File Name field of the Movie Inspector, enter ‘MyFirstMovie.mov’. This will be the file name of your QuickTime movie.

2. Make sure ‘Fixed Size’ is checked. In the Width field enter 367 and in the Height field enter 279. These will be the final dimensions of your movie.



File Name: MyFirstMovie.mov
Version: QuickTime 6.0
Controller: None
Loop: None
Window: Off
□ Auto enter full screen
Width: 367 ☑ Fixed Size
Height: 279 ☑ Background
Color:

3. Click on the Advanced Tab in the Inspector and select Auto Start. This will make your exported movie start as soon as it is launched.



Play: ☑ Auto Start
□ All Frames
□ Selection Only
Movie: ☑ Compress
□ Copy Protect
☑ Auto Alternates
XML File: < no file set >
Tag: ● Name:
○ ID:

BRINGING MEDIA ELEMENTS INTO LIVESTAGE PROFESSIONAL

LiveStage Professional allows you to integrate over 200 different types of media formats, all of which can be incorporated into your projects with a simple drag-and-drop.

THE LIBRARY FOLDER

When you first saved your project, LiveStage Professional created a folder called ‘Library’ for you in the same directory or folder as your project file. This is the folder where you will store the media associated with your present project.

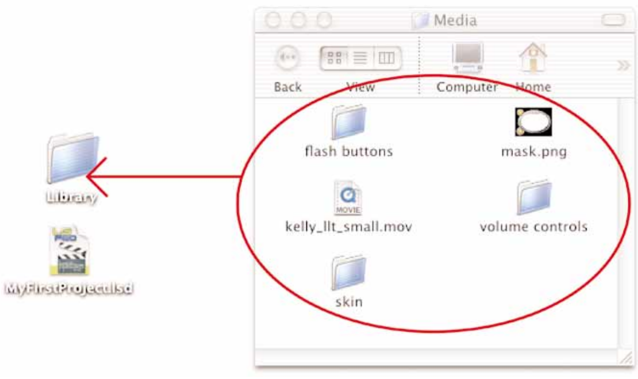
1. Open the Tutorial folder you dragged to the desktop from the LiveStage Professional 4.0 CD-ROM.

2. Open the Media folder and select all the files inside it.

3. Drag the selected files to the Library folder located in the same directory as your project file.

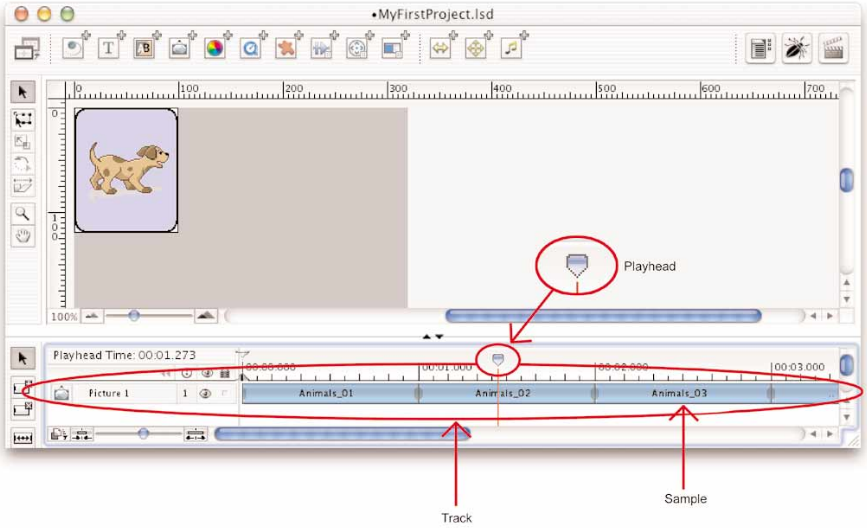
4. Return to the Document Window in LiveStage Professional. In the Library Window, click on the Local Tab.

The Local Tab corresponds to the 'Library' folder in your project file's directory. All the media you just dragged to your project file's Library folder is now visible and accessible through this window.



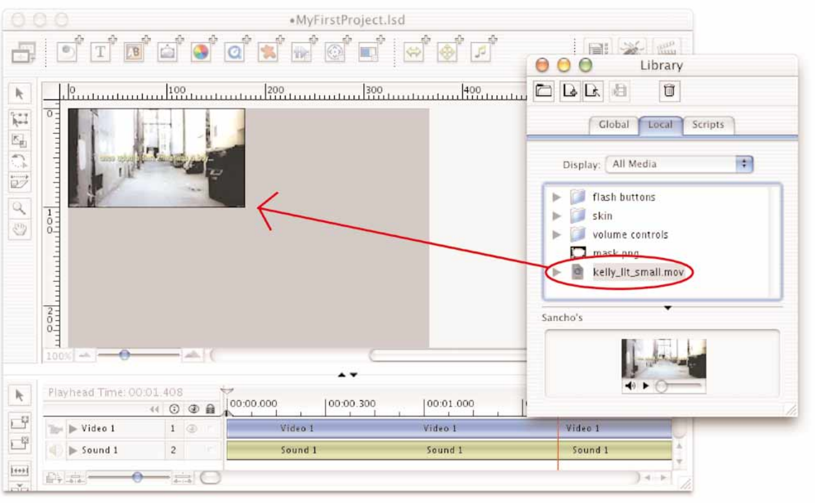
Working in the Timeline

The Timeline houses *Tracks* that correspond to the type of media involved (e.g., a Picture Track for still images, a Sound Track for audio), which in turn house individual instances, or *Samples*, of that media. Tracks are laid out along the Timeline. The Stage will display visible tracks in the Timeline at the point where the Playhead is currently located.

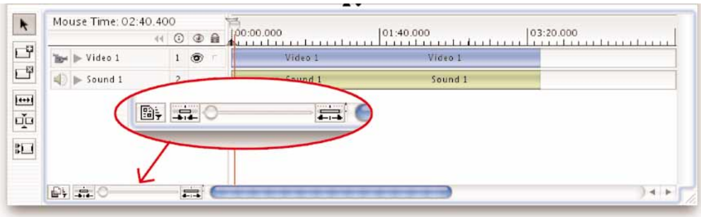


1. From the Local tab in the Library Window, click on 'kelly_llt_small.mov' and drag it to the Stage.

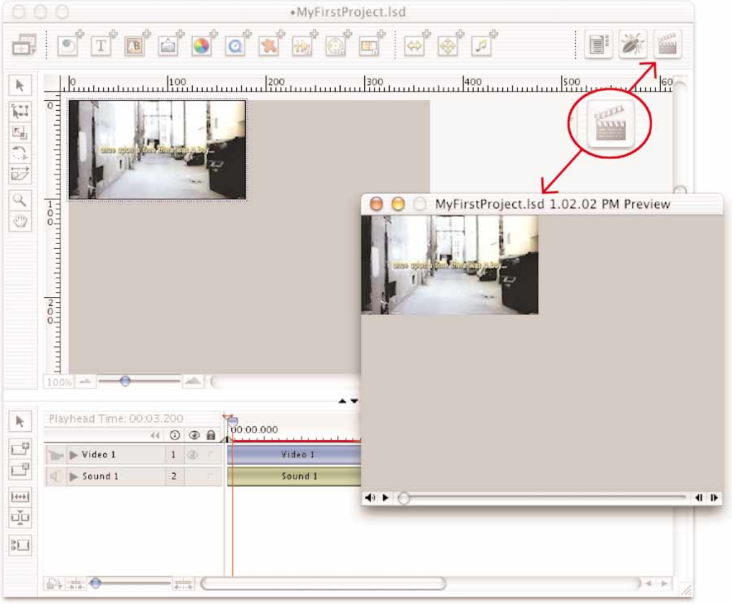
   Two tracks appear—'Video 1' and 'Sound 1'. As the video contains a soundtrack, LiveStage Professional automatically separates the video and sound into two tracks.



2. Click on the Zoom Out button or drag the Zoom Slider in the bottom left corner of the Timeline until you see the tracks in their entirety.



3. In the top right hand corner of the Stage/Timeline window, click on the button with the movie slate. This button 'previews' your work so far. The movie you just dragged in appears over a grey background.



4. Close the Preview window and save your work (Command-S).

ADDING A FRAME TO YOUR VIDEO

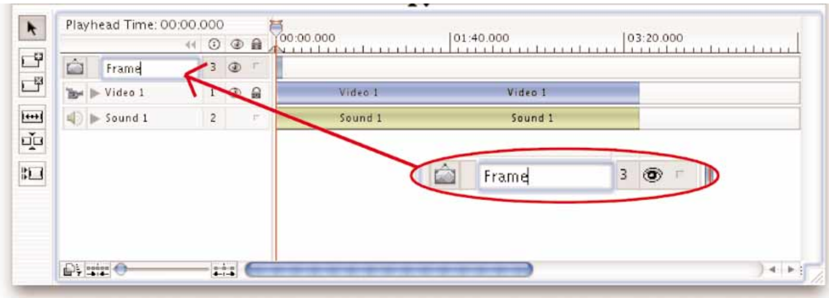
A simple way to add polish and impact to a simple video is to present it through a custom frame. This can be particularly helpful if you want your video to have a shape other than a typical rectangle, or you want to incorporate it seamlessly into web page's background graphics.

1. From the Local Library folder, drag and drop 'mask.png' into the Timeline.

   A Picture Track appears in the Timeline with the label 'Picture 1' in the Track Header at the left of the track. The track also shows up in the Stage. Note the track duration defaults to one second.

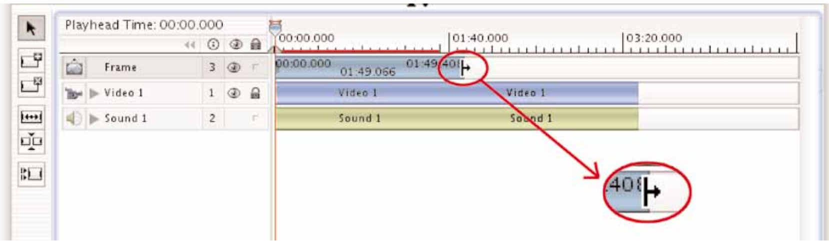
2. Click once and pause your Mouse in place over the 'Picture 1' label in the Track Header.

   When a highlight appears on the name, you can edit it. Change the name of the Track to 'Frame'. It is a good idea to always change the names of your tracks to something meaningful.

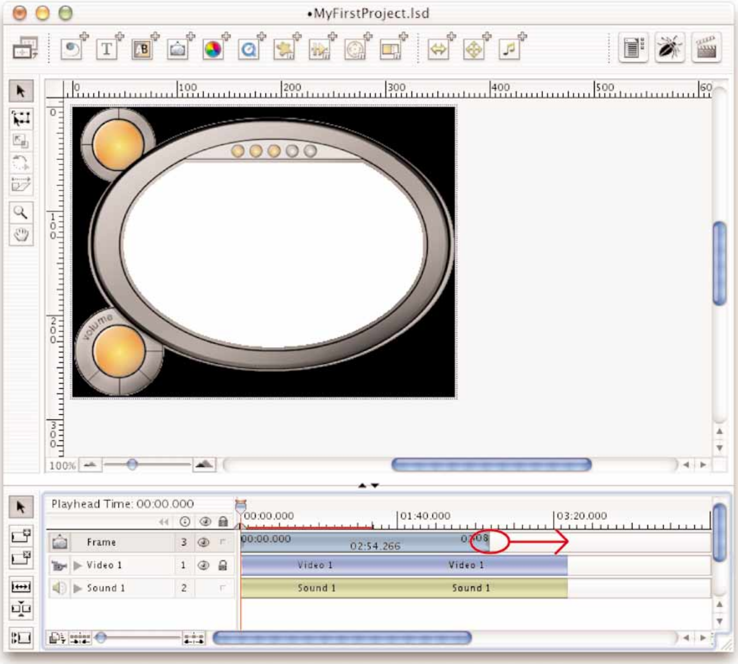


3. To make sure the frame is the same duration as the video you will need to change its duration or it will disappear after the movie plays for one second.

   Move your mouse over the right edge of the 'mask.png' sample in the 'Frame' track until it turns into a drag tool. If the Sample is too small and the cursor does not reflect the drag tool, zoom in a bit using the zoom sliders on the timeline.



4. Click and drag the right edge of the Frame Sample until it ‘snaps’ to the end of the ‘Video 1’ and ‘Sound 1’ tracks.



ADJUSTING MEDIA

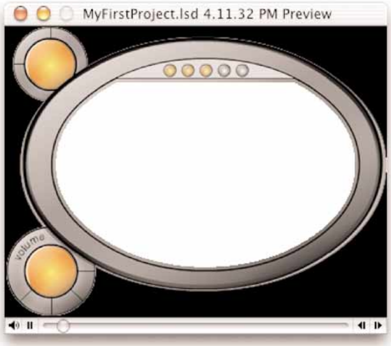
While you generally only bring completed media elements into LiveStage Professional, adjustments are often necessary once media has been added to the project.

SELECTING DRAWING MODES

Drawing Modes are used to alter the appearance of your existing visual media. For instance, you can activate transparency in a 32-bit image through the Alpha Channel Drawing Mode.

1. Preview your movie.

   The soundtrack plays, but the video does not show through the frame. The Frame graphic is a PNG file created in Photoshop with an Alpha Channel that, when activated, will render the white portion in the top center transparent.

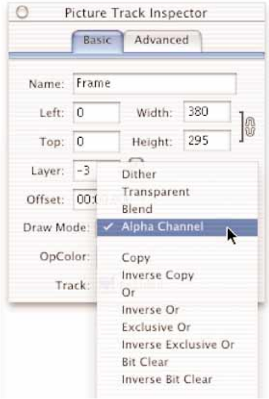


2. Close the preview.

3. Click on the Track Header for the “Frame” Picture Track.

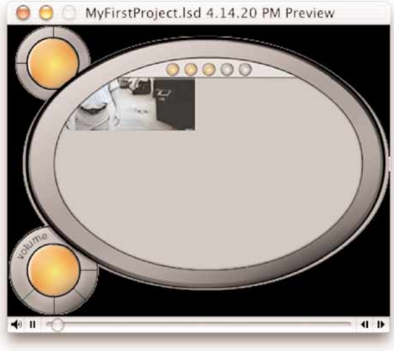
   The Inspector changes to a ‘Picture Track Inspector’ and reveals the properties of the Frame Track.

4. From the Draw Mode pull-down menu of the Inspector Window, select Alpha Channel.



Name: Frame
Left: 0 Width: 380
Top: 0 Height: 295
Layer: -3
Offset: 00:(
Draw Mode: Alpha Channel

5. On the Stage, the video shows through but it is not centered and is too small. You will need to correct the position and size of the video.

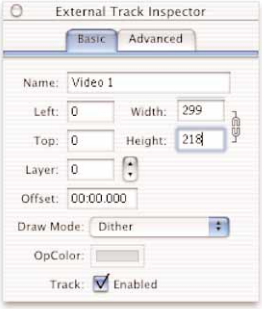


SCALING MEDIA

Generally, it is best to ensure that your media is appropriately sized prior to bringing it into LiveStage Professional. When adjustments need to be made, it is easy to make alterations within LiveStage Professional. However, scaling media may cause performance problems in the final movie.

1. To resize your video, click the ‘Video 1’ Track Header. The Inspector will change to display ‘Video 1’ Track information.

2. Change the Width to 299 and the Height to 218.



External Track Inspector

Basic | Advanced

Name: Video 1

Left: 0 Width: 299

Top: 0 Height: 218

Layer: 0

Offset: 00:00.000

Draw Mode: Dither

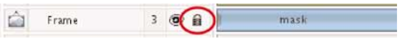
OpColor:

Track: ☑ Enabled

REPOSITIONING TRACKS ON THE STAGE

While you could reposition the video by entering coordinates into the Left and Top fields of the ‘Video 1’ Track Inspector, we will use the Stage instead. First, you will need to lock down the Frame Track so that you will not move it unintentionally.

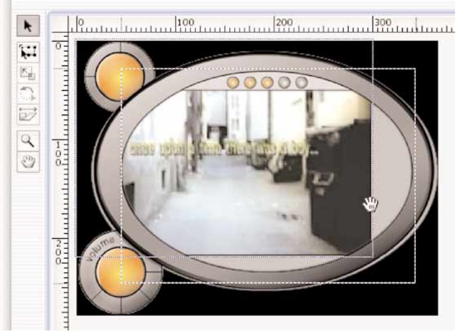
1. Click on the Frame Track in the Timeline and click the lock box to the right of the eye icon. This locks the track in place.



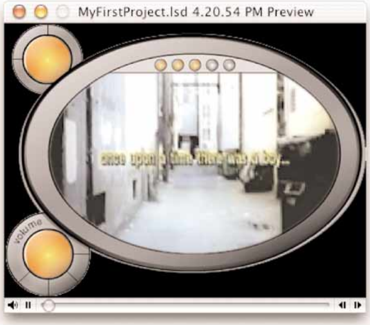
2. Click the ‘Video 1’ Track in the Timeline.

   A blue bounding box appears on the Stage. This is the ‘Video 1’ Track.

3. Move your cursor into the blue outline and when it turns into a hand, click and drag to reposition the video within the frame cutout. The final position will be roughly left=43 and top =25.



4. Preview your movie. The video now plays through the frame.


MyFirstProject.lsd 4.20.54 PM Preview
once upon a time there was a boy...
volume

5. Close the Preview and Save your project (Command-S).

WORKING WITH THE SPRITE TRACK

Sprite Tracks do the most of the ‘heavy lifting’ where LiveStage Professional interactivity is concerned. Generally speaking, when you want to have an element in your movie to ‘do’ something, create a Sprite Track. Sprite Tracks can operate independently or they can interact with other tracks, samples, or even different movies.

ADDING VOLUME CONTROLS

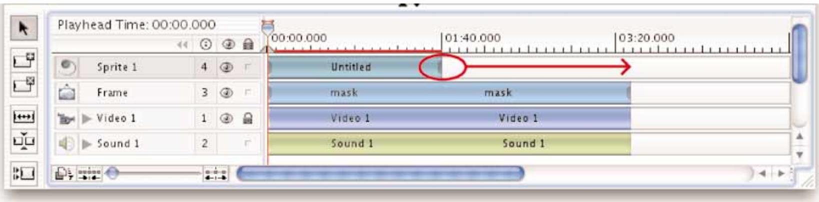
Sprite Tracks are often used to create buttons to control elements of an interactive movie, or the entire movie itself. In this section, you will use a Sprite Track to create buttons to control the volume of the movie.

1. Click on the to create a Sprite Track.

   The Sprite Track appears as a white area on the Stage.

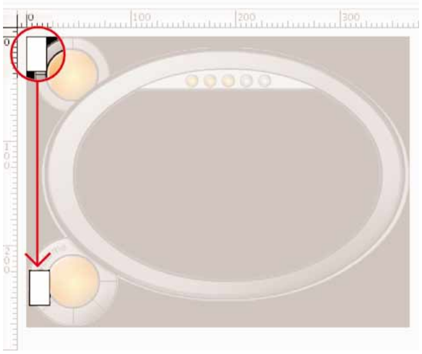
2. Rename the Sprite Track “Volume Controls.”

3. In the Timeline, drag the right edge of the Sprite Track’s sample to snap its duration to match that of the other tracks. Use the zoom tools if necessary to view the tracks in the timeline.

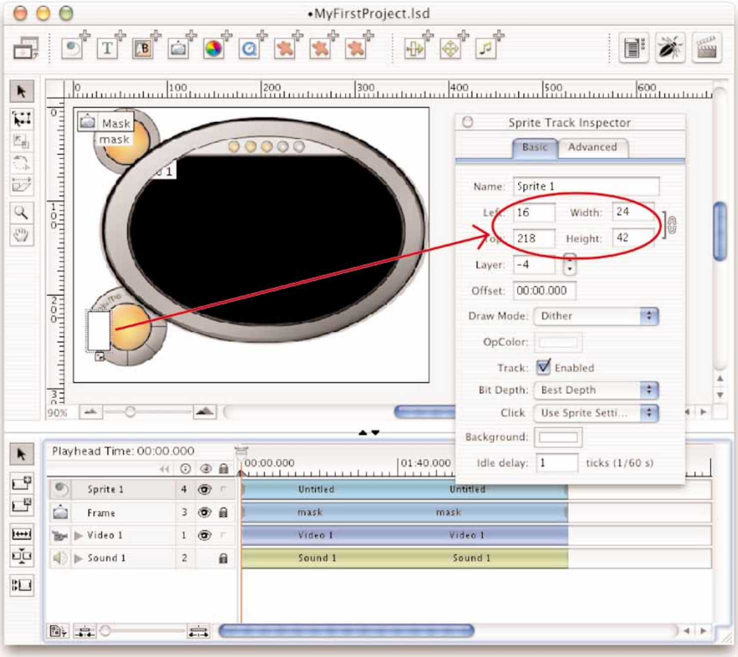


4. On the Stage, click and drag on the bottom right hand corner of the white Sprite Track area until it is roughly 20 pixels wide and 34 high.

   You can determine or set these coordinates using the Width and Height fields in the Sprite Track Inspector.



5. Now click and drag the entire track until the Left edge (or X coordinate) is at 3 and the Top edge (or Y coordinate) is at 224. Again, these coordinates can be obtained or set using the Top and Left fields of the Sprite Track Inspector.



•MyFirstProject.lsd

Sprite Track Inspector

Basic | Advanced

Name: Sprite 1

Left: 16 Width: 24

Top: 218 Height: 42

Layer: -4

Offset: 00:00.000

Draw Mode: Dither

OpColor:

Track: ☑ Enabled

Bit Depth: Best Depth

Click: Use Sprite Setti...

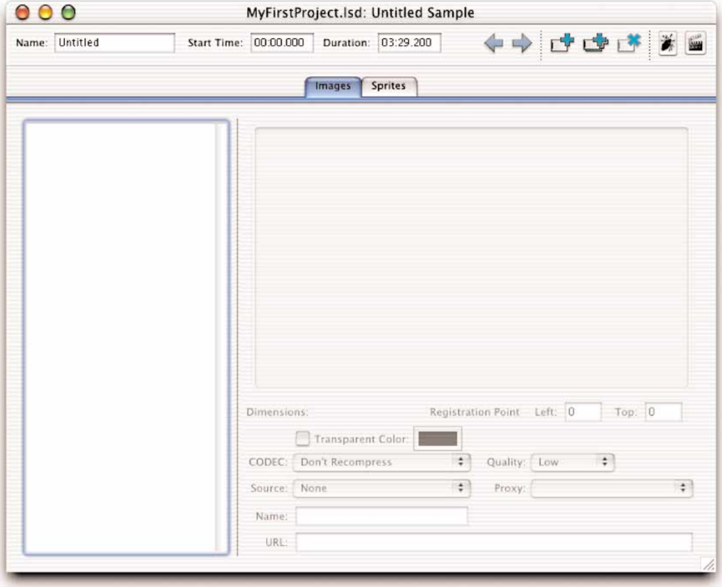
Background:

Idle delay: 1 ticks (1/60 s)

Playhead Time: 00:00.000

Track	Layer	Samples
Sprite 1	4	Untitled, Untitled
Frame	3	mask, mask
Video 1	1	Video 1, Video 1
Sound 1	2	Sound 1, Sound 1

6. Now that you have a Sprite Track in place, it is time to create some Sprites. Double-click on the 'Untitled' Sprite Sample of the Sprite Track in the Timeline. The Sprite Sample Properties window opens. Click on the Images tab.



MyFirstProject.lsd: Untitled Sample

Name: Untitled Start Time: 00:00.000 Duration: 03:29.200

Images | Sprites

Dimensions:

Registration Point Left: 0 Top: 0

☐ Transparent Color:

CODEC: Don't Recompress Quality: Low

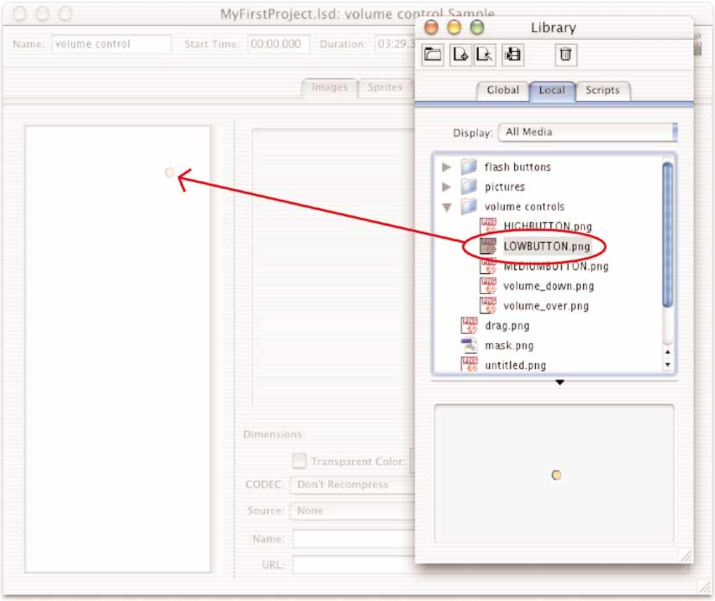
Source: None Proxy:

Name:

URL:

7. In the Local tab in the Library window, open the “Volume Controls” folder by double-clicking on it and then drag ‘LOWBUTTON.png’ to the pane at the left of the Sprite Sample Properties window.

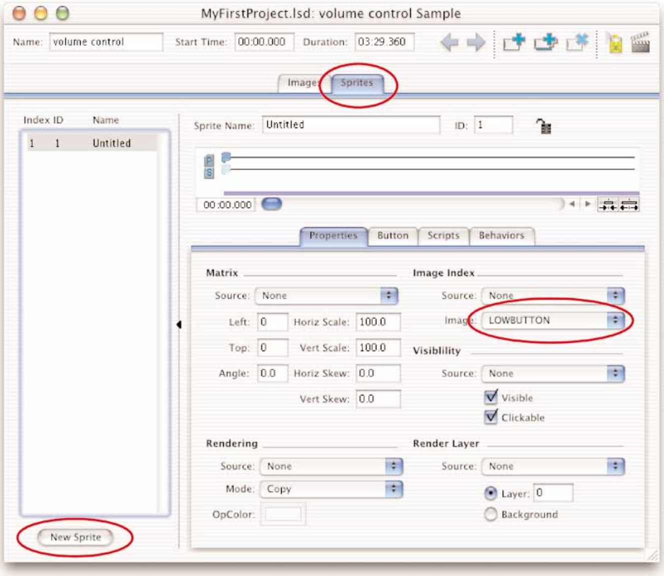
   The name ‘1 LOWBUTTON.png’ appears in the left pane, and the image shows up in the right pane.



Name: volume control
Start Time: 00:00.000
Duration: 03:29.3
CODEC: Don't Recompress
Source: None
Display: All Media

- flash buttons
- pictures
- volume controls
  - HIGHBUTTON.png
  - LOWBUTTON.png
  - MEDIUMBUTTON.png
  - volume_down.png
  - volume_over.png
- drag.png
- mask.png
- untitled.png

8. Click the Sprites Tab of the Sprite Sample Properties window and click the ‘New Sprite’ button located at the bottom left corner of the window.



Name: volume control
Start Time: 00:00.000
Duration: 03:29.360

Index	ID	Name
1	1	Untitled

Sprite Name: Untitled
ID: 1
00:00.000

Matrix
Source: None
Left: 0
Horiz Scale: 100.0
Top: 0
Vert Scale: 100.0
Angle: 0.0
Horiz Skew: 0.0
Vert Skew: 0.0

Image Index
Source: None
Image: LOWBUTTON

Visibility
Source: None
☑ Visible
☑ Clickable

Rendering
Source: None
Mode: Copy
OpColor:

Render Layer
Source: None
● Layer: 0
○ Background

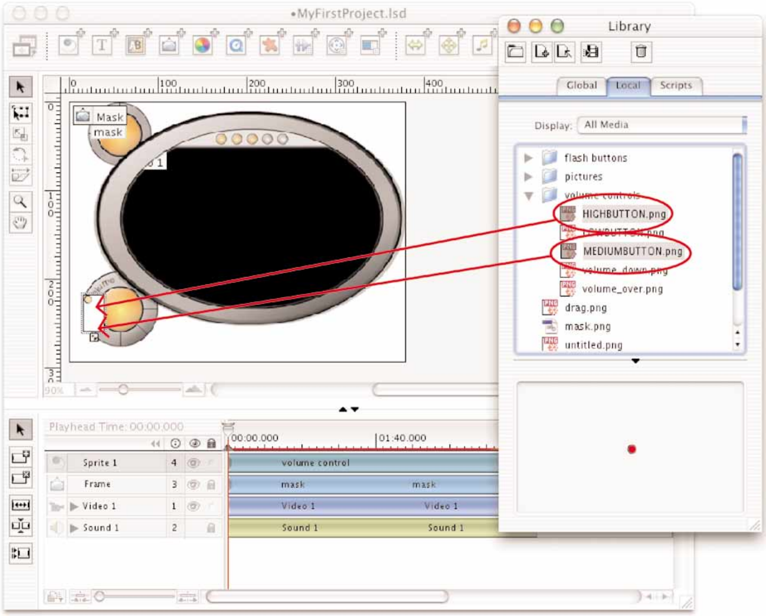
A new sprite called ‘Untitled’ appears. Change the Sprite name to “LOWBUTTON.”

Notice in the Image Index section at the right of the Sprite Sample properties window that the Image is already set to LOWBUTTON.png. If you had dragged in several images, you would need to specify which image you wanted your sprite to have through this pop-up menu.

9. Close the Sprite Sample properties window. You will see that a Sprite displaying ‘LOWBUTTON.png’ has appeared on the Stage.

   You can also create Sprites automatically by simply dragging images directly to the Sprite Track on the Stage or by dragging them to the Sprite Sample in the Timeline.

10. From the “Volume Controls” folder in the Local tab of the Library Window, drag ‘MEDIUMBUTTON.png’ and ‘HIGHBUTTON.png’ into the Sprite Track on the Stage. If you like, you can use the zoom slider on the stage to better view the Sprite Track.



11. Click on the small checkered box at the bottom right corner of the Sprite Track on the Stage.

    This accesses the Sub-Object Editor and allows you to work with the individual sprites you have just created in the Sprite Track.

12. Click on the LOWBUTTON.png sprite on the Stage.

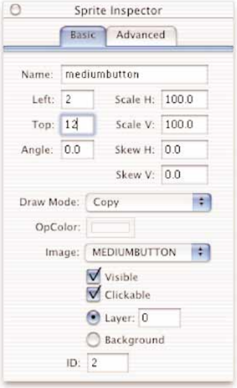
    The Inspector updates to reflect the individual sprite’s properties. (If the Inspector is not visible, select Show Inspector (Command- I) from the Window menu.)

13. In the Left field, enter 9. In the Top field, enter 22.

14. Set the coordinates for the high and medium buttons by clicking on each and adjusting the Left and Top fields in the Sprite Inspector as follows:

HIGHBUTTON.png	Left = 1
	Top = 2

MEDIUMBUTTON.png	Left = 5
	Top = 12

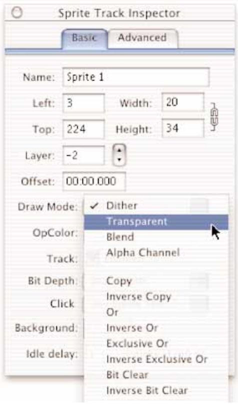


Name: mediumbutton
Left: 2 Scale H: 100.0
Top: 12 Scale V: 100.0
Angle: 0.0 Skew H: 0.0
Skew V: 0.0
Draw Mode: Copy
OpColor:
Image: MEDIUMBUTTON
☑ Visible
☑ Clickable
● Layer: 0
○ Background
ID: 2

15. Click on the Sub-Object Editor button to deactivate sub-object editing.

The Inspector changes to reflect the Sprite Track properties.

16. In the Sprite Track Inspector, select ‘Transparent’ from the Draw Mode pull-down menu to make the white background of the Sprite Track transparent.



Name: Sprite 1
Left: 3 Width: 20
Top: 224 Height: 34
Layer: -2
Offset: 00:00.000
Draw Mode: Transparent
OpColor:
Track:
Bit Depth:
Click
Background:
Idle delay:

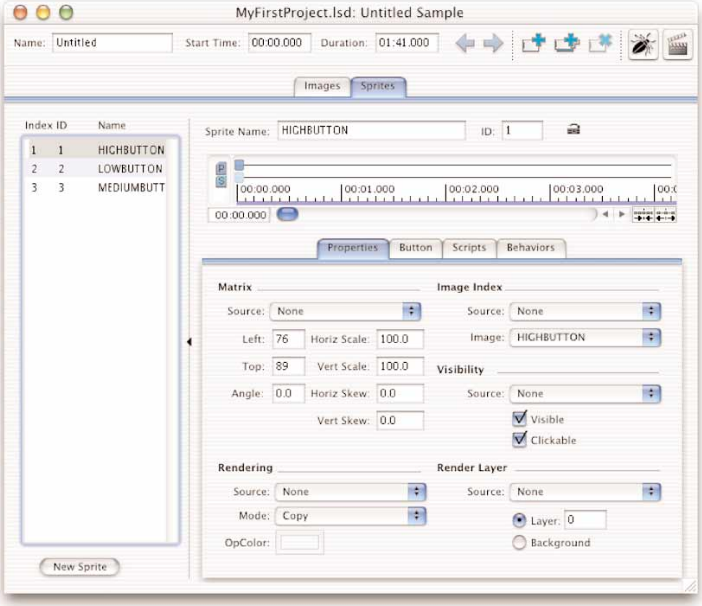
tip

When your Track background color is anything other than white, you need to select the same background color in the OpColor swatch to render the background transparent.

SCRIPTING VOLUME CONTROLS

Simple commands turn these Sprite images into interactive controllers through QScript, or ‘QuickTime Script’.

1. Double-click on the Sprite Track Sample to open the Sprite Sample Window. The button names appear in the Sprite tab’s left pane.



MyFirstProject.lsd: Untitled Sample

Name: Untitled Start Time: 00:00.000 Duration: 01:41.000

Index	ID	Name
1	1	HIGHBUTTON
2	2	LOWBUTTON
3	3	MEDIUMBUTT

Sprite Name: HIGHBUTTON ID: 1

00:00.000

Matrix
Source: None
Left: 76 Horiz Scale: 100.0
Top: 89 Vert Scale: 100.0
Angle: 0.0 Horiz Skew: 0.0
Vert Skew: 0.0

Image Index
Source: None
Image: HIGHBUTTON

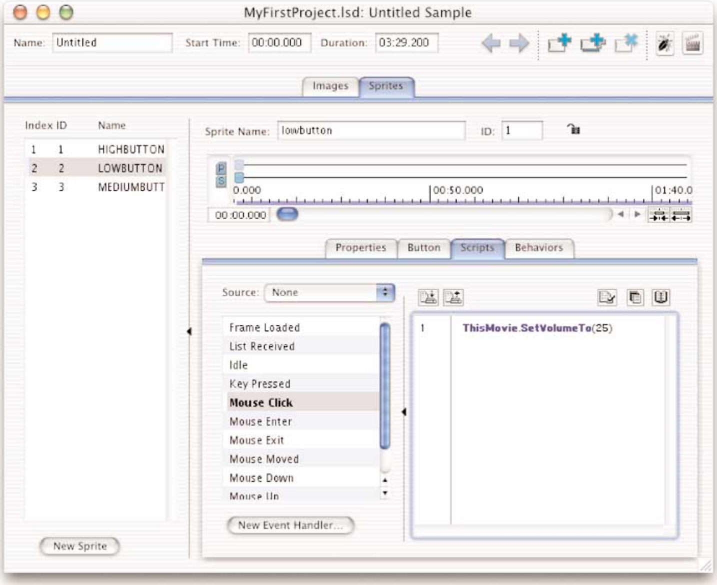
Visibility
Source: None
☑ Visible
☑ Clickable

Rendering
Source: None
Mode: Copy
OpColor:

Render Layer
Source: None
● Layer: 0
○ Background

2. Select ‘LOWBUTTON’ and click on the Scripts tab.

3. Select Mouse Click from the list that appears and in the window to the right enter: ThisMovie.SetVolumeTo(25).



MyFirstProject.lsd: Untitled Sample

Name: Untitled Start Time: 00:00.000 Duration: 03:29.200

Index	ID	Name
1	1	HIGHBUTTON
2	2	LOWBUTTON
3	3	MEDIUMBUTT

Sprite Name: lowbutton ID: 1

00:00.000

Source: None

Mouse Click

```
ThisMovie.SetVolumeTo(25)
```

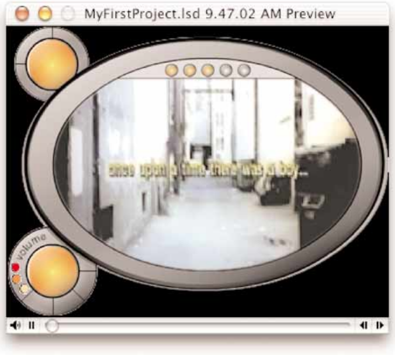
Volume settings have a minimum of 0 (mute) and a maximum of 255 (loudest). The number in the brackets following the 'SetVolumeTo' determines the volume level.

4. Repeat steps 2 and 3 for the remaining buttons with the following scripts:

'MEDIUMBUTTON'	ThisMovie.SetVolumeTo(125)
'HIGHBUTTON'	ThisMovie.SetVolumeTo(255)

5. Close the Sprite Sample window.

6. Preview your movie, and play with the buttons to hear the difference.


MyFirstProject.lsd 9.47.02 AM Preview
once upon a time there was a boy...
volume

7. Close the Preview and save your project (Command-S).

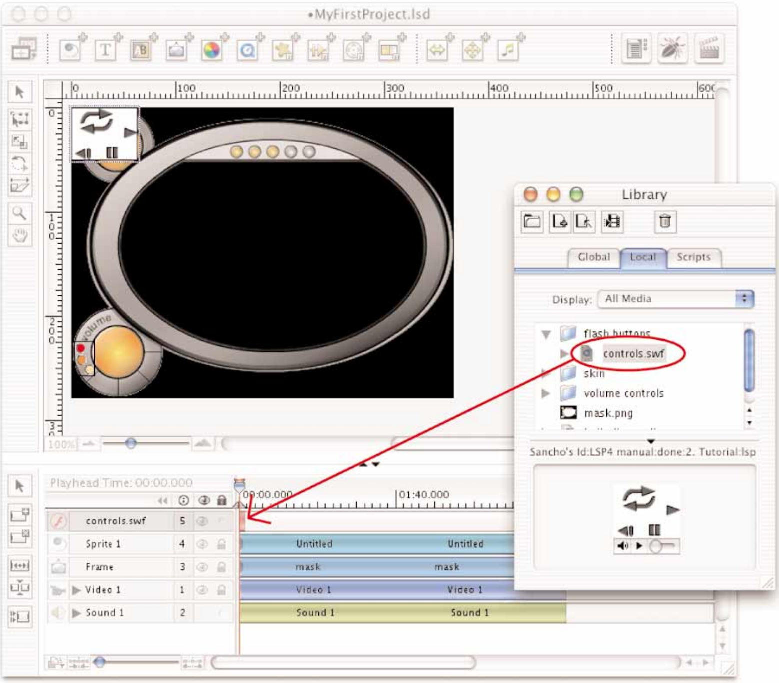
WORKING WITH FLASH

Flash media—or .swf files—are one of the many types of media that QuickTime and LiveStage Professional recognize. This allows you to incorporate impressive Flash animations with a multitude of other media. You can even reprogram Flash buttons to control specific QuickTime functions.

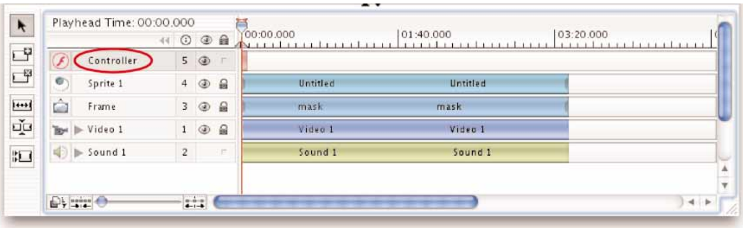
ADDING A FLASH CONTROLLER

An animated video controller built in Flash is easy to incorporate and script in LiveStage Professional.

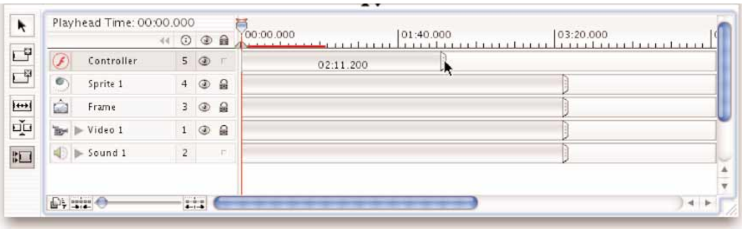
1. Drag in 'controls.swf' from the Flash buttons folder in the local tab of the Library window into the timeline.



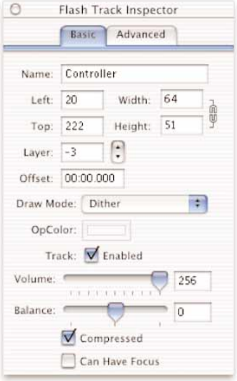
2. Click the ‘controls.swf’ name in the Track Header and change it to ‘Controller’.



3. Click on the Track Manipulation tool to the left of the Timeline and then drag the sample to match the duration of the rest of your tracks. Use the zoom tools to view the tracks if necessary.



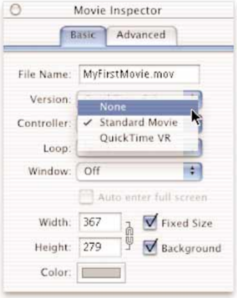
4. In the Inspector, set the Left value to 20, the Top value to 222, the Width to 64, and the Height to 51.



Name: Controller
Left: 20 Width: 64
Top: 222 Height: 51
Layer: -3
Offset: 00:00.000
Draw Mode: Dither
OpColor:
Track: ☑ Enabled
Volume: 256
Balance: 0
☑ Compressed
☐ Can Have Focus

5. As you are adding a controller, you will no longer need the standard one that QuickTime provides. Click in the empty area below the tracks in the Timeline or outside of the tracks on the Stage to make the Movie Inspector active.

6. Under the Controller pull-down menu select None.



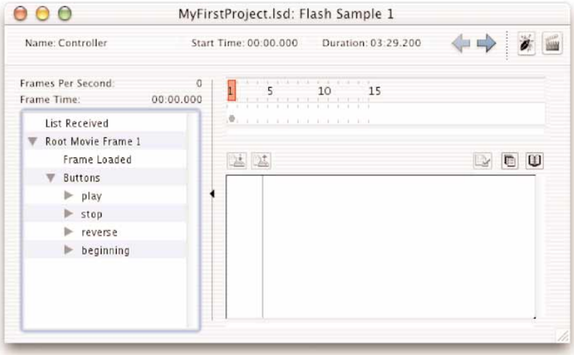
File Name: MyFirstMovie.mov
Version:
Controller: None
Loop:
Window: Off
☐ Auto enter full screen
Width: 367 ☑ Fixed Size
Height: 279 ☑ Background
Color:

SCRIPTING A FLASH CONTROLLER

Flash animations retain the scripting given to them in Flash. You can also augment their functionality with QScript to control functions within your QuickTime movie.

1. Double-click on the Controller Sample. The Flash Sample Properties window opens.

2. Click on the triangle to the left of 'Root Movie Frame 1', then click on 'Buttons'. A list of the controller's buttons is revealed.

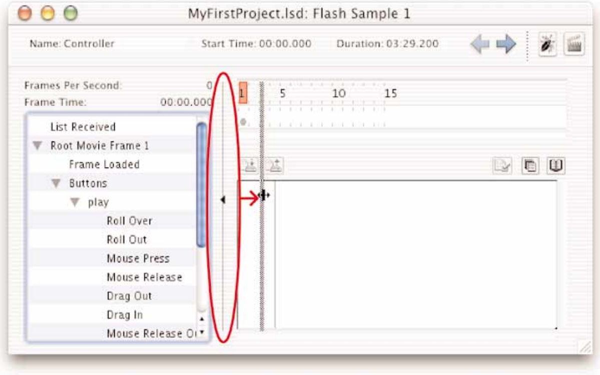


Name: Controller
Start Time: 00:00.000
Duration: 03:29.200
Frames Per Second: 0
Frame Time: 00:00.000

List Received
▼ Root Movie Frame 1
Frame Loaded
▼ Buttons
▶ play
▶ stop
▶ reverse
▶ beginning

3. Open the Play Button by clicking on its triangle.

   If you need to expand the window to view the button names in their entirety, move your mouse over the center line until it turns into a drag tool and drag it to achieve the desired view.



MyFirstProject.lsd: Flash Sample 1

Name: Controller

Start Time: 00:00.000

Duration: 03:29.200

Frames Per Second: 0

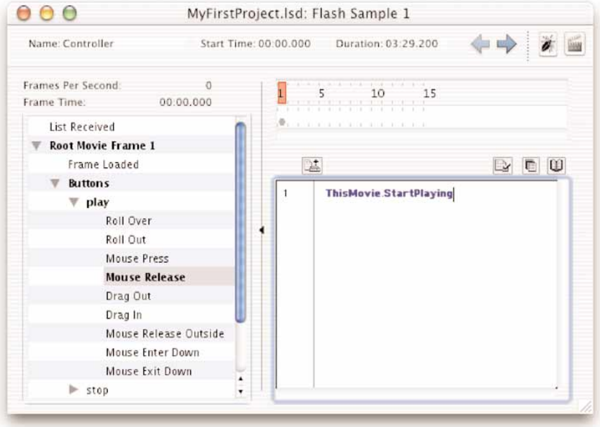
Frame Time: 00:00.000

4. Select Mouse Release.

5. In the Script Edit Window, enter the following script: ThisMovie.StartPlaying

   The script turns blue as you type, confirming that its syntax is valid.

   Note that this ONLY confirms proper QScript syntax, not whether or not the script will execute with the desired results.



MyFirstProject.lsd: Flash Sample 1

Name: Controller

Start Time: 00:00.000

Duration: 03:29.200

Frames Per Second: 0

Frame Time: 00:00.000

```
ThisMovie.StartPlaying
```

6. Select the Stop Button and then select Mouse Release. Scripting the Stop Button is as simple as scripting the Play Button.

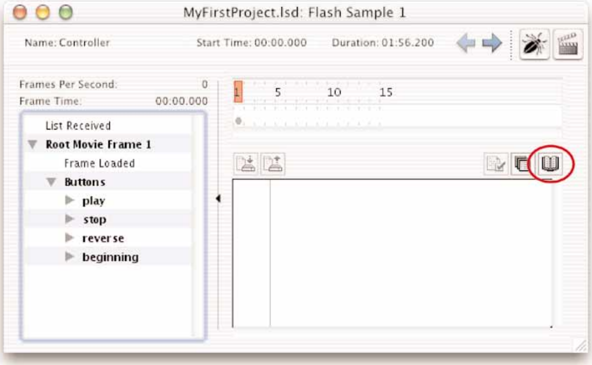
   ThisMovie.StopPlaying

   StartPlaying and StopPlaying are intuitive enough, but what if you have no idea what the script would be? LiveStage Professional provides an invaluable reference that will help you find the exact script you need: the QScript Reference.

THE QSCRIPT REFERENCE

Scripting interactivity in LiveStage Professional is not solely the purview of experts. QScript, or 'QuickTime Script' is an intuitive language that is easy to learn, and LiveStage Professional makes it even easier with a comprehensive, highly functional QScript Reference.

1. Click the book icon just above the top left corner of the Script Edit Window to open the QScript Reference if it is not already open.



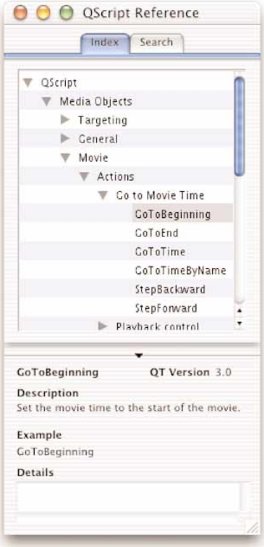
MyFirstProject.lsd: Flash Sample 1

Name: Controller Start Time: 00:00.000 Duration: 01:56.200

Frames Per Second: 0
Frame Time: 00:00.000

If you can't remember where a particular script is located in the QScript Reference window, click on the Search tab and enter a keyword such as 'beginning' or 'play'. All the different scripts that are relevant to these keywords will display. You can also drag scripts into your project directly from the Search tab.

2. The QScript Reference houses all the scripting elements for creating interactive projects. In the Index tab, click the triangle next to 'Media Objects'. Click on the triangles next to 'Movie' and then 'Actions' to reveal the various scripts available to control the whole movie. Finally, click to expand the 'Go to Movie Time' section.



GoToBeginning QT Version 3.0

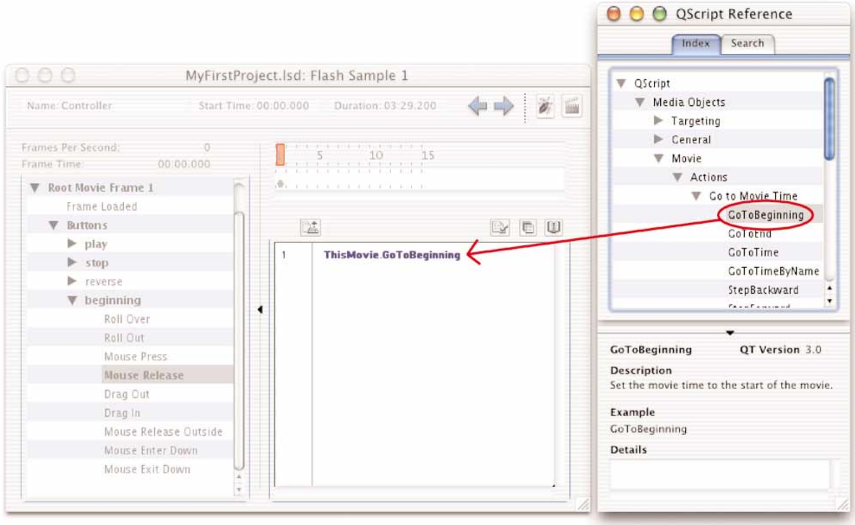
Description
Set the movie time to the start of the movie.

Example
GoToBeginning

Details

3. Click on the 'GoToBeginning' item listed in this section.

4. The 'GoToBeginning' script will send the Movie's playhead to the start of the movie. To script this, select the 'Beginning' button in the Flash Sample Properties window, select the mouse release event and type in the script edit window "This Movie." Then drag the GoToBeginning item from the QScript Reference Window to the Script Edit Window.



MyFirstProject.lsd: Flash Sample 1

Name: Controller Start Time: 00:00.000 Duration: 03:29.200

Frames Per Second: 0
Frame Time: 00:00.000

```
1 ThisMovie.GoToBeginning
```

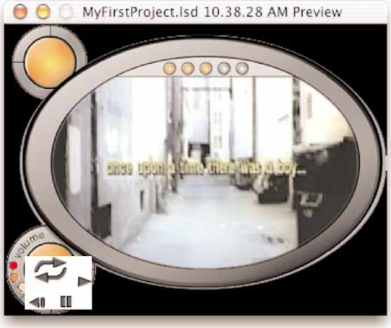
QScript Reference

GoToBeginning QT Version 3.0
Description
Set the movie time to the start of the movie.
Example
GoToBeginning
Details

5. Complete the mouse release events of the remaining buttons in the same way:
Stop: ThisMovie.StopPlaying
Reverse: ThisMovie.SetRateTo(-1)

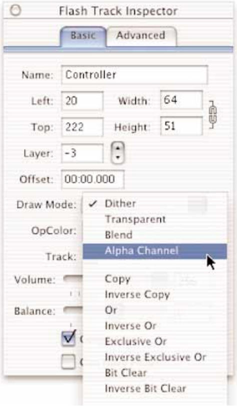
A negative rate plays the movie in reverse: the higher the number, the faster in reverse the movie plays. Close the Flash Sample Properties window.

6. Preview your movie and test the buttons.



MyFirstProject.lsd 10.38.28 AM Preview
once upon a time there was a boy...
volume

7. To make the white background of the controller transparent, select ‘Alpha Channel’ from the Track Inspector with the Flash Track selected/inspected.



Flash Track Inspector

Name: Controller
Left: 20 Width: 64
Top: 222 Height: 51
Layer: -3
Offset: 00:00.000

8. Close the Preview and save your project/document (Command-S).

ADDING A MEDIA SKIN

Your LiveStage Professional project now exports an interactive video presentation that can be launched within the QuickTime Player or a browser. However, you can take this one step further. Move it beyond the browser or conventional player and create a completely branded cohesive presentation through Media Skins. LiveStage Professional's new FastTrack™ feature makes building these impressive presentations quick and easy.

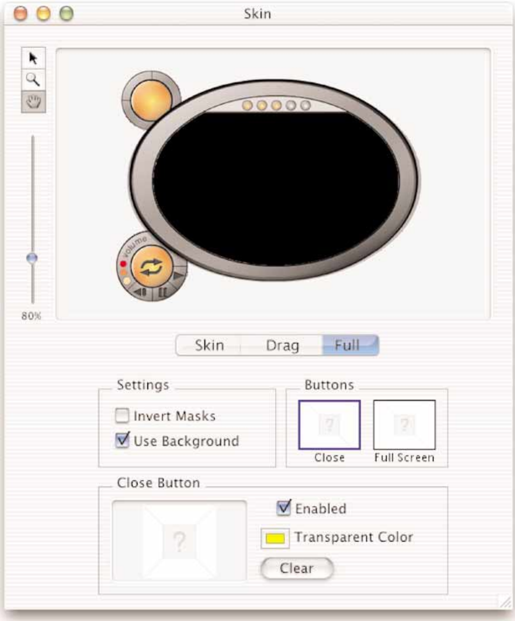
SKIN ELEMENTS

Skins consist primarily of two elements—the Window Shape that defines the shape of your player, and the Drag Area. The Drag Area defines where on the presentation the viewer will be able to click in order to drag the movie around the desktop. Applying a skin removes the brushed metal skin of the standard QuickTime player. The brushed metal skin is used to drag the player around the desktop, so once you have a skin you will need to define a new Drag Area. The media elements that are used to create Skins are simple 1-bit images created in Photoshop or another graphics program that represent the shapes you wish to impose on your movie.

1. If you do not have the Track Creation toolbar active, click on the Switch Toolbar button.

2. Click on the to create a Media Skin FastTrack™ and open its editor.

   A default Media Skin background image appears in the editor.



☐ Invert Masks

☑ Use Background

☑ Enabled

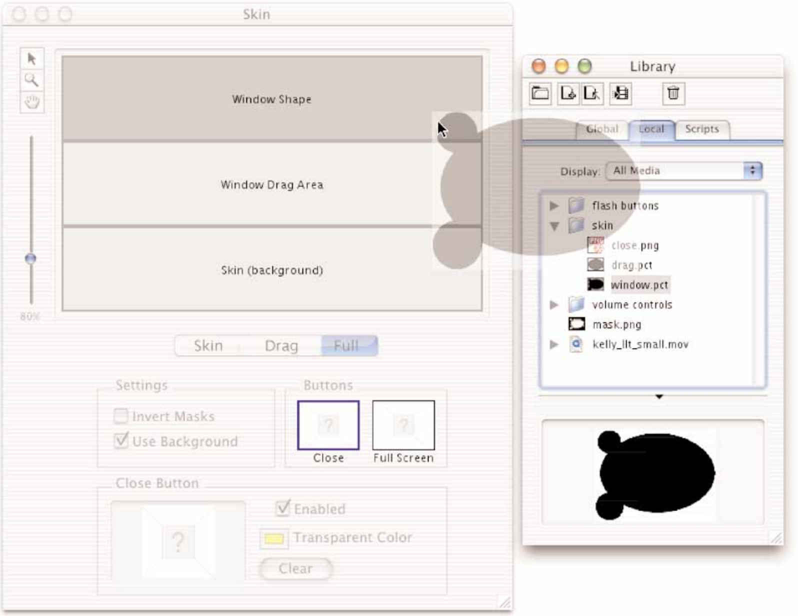
3. Deselect the "Use Background" checkbox. The default background image disappears.

4. From the Skin Folder in the Local Library, select 'window.pct' and drag it to the Media Skin FastTrack Editor main pane without releasing the mouse.

   Three options appear: Window Shape, Window Drag Area, and Skin (Background).

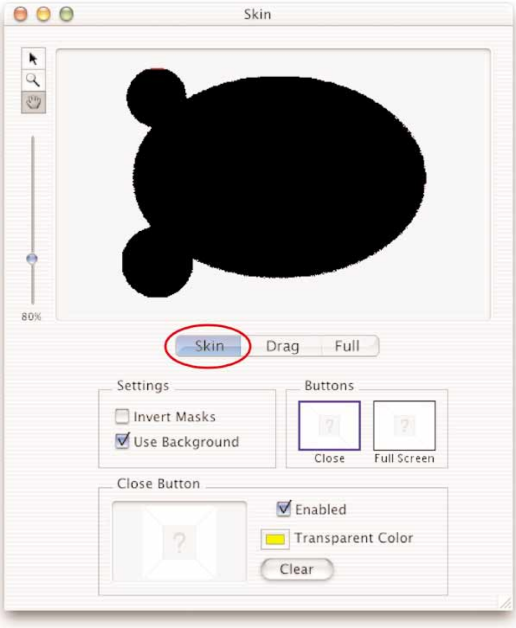
5. Release the image over the 'Window Shape' option.

   Use the zoom tool and the hand tool to fit the whole movie inside the main pane if necessary.



□ Invert Masks
☑ Use Background
☑ Enabled
Transparent Color
Clear

6. Click on the 'Skin' Button to view the cutout your Skin will create.



□ Invert Masks
☑ Use Background
☑ Enabled
Transparent Color
Clear

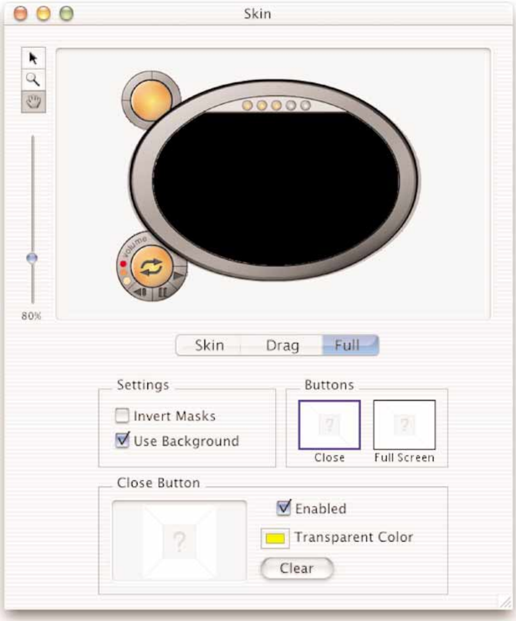
7. Now select ‘drag.pct’ from the Skin Folder in the local Library and drag it to the Skin Editor main pane, dropping it onto the Window Drag option.

   As the movie already has the frame graphic for a background in the Picture Track, you will not need to use a skin background.

8. To view the area that the user will be able to click and drag on, click the Drag Button.

9. To view the full effect, either click on Full, or go back to the Stage.

   The Skin Track is much like a cookie cutter that has determined the shape your movie will take.



Settings

□ Invert Masks

☑ Use Background

☑ Enabled

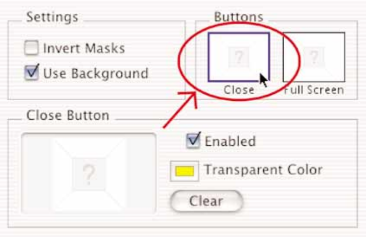
Transparent Color

ADDING A CLOSE BUTTON

One final element that is required with a Skinned Movie is the Close Button. Without one, your audience will likely not know how to exit the presentation.

1. If the Skin Editor is not already open, double-click the Skin sample to open it.

2. Click on the white box just above Close.
   The box outlines in blue and the Image Well below changes to ‘Close Button’.



Settings

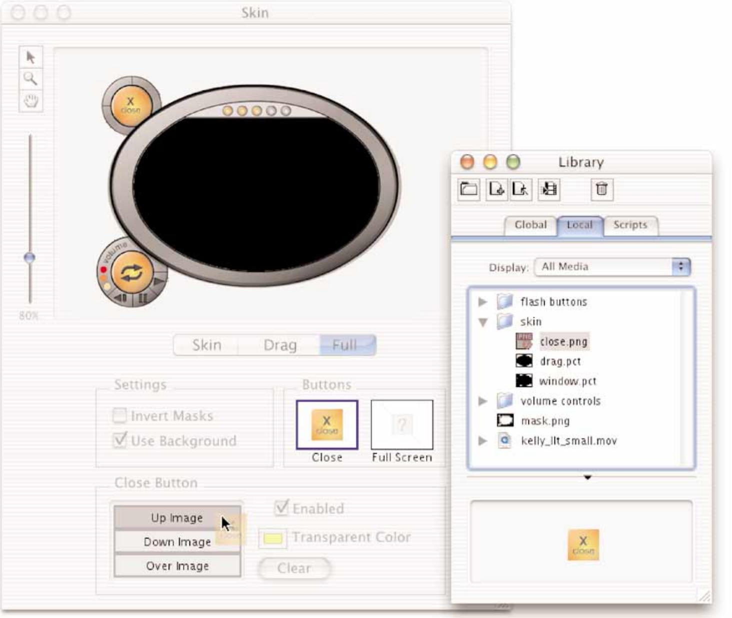
□ Invert Masks

☑ Use Background

☑ Enabled

Transparent Color

3. From the Skin Folder of the Local Library, drag ‘close.gif’ into the image well.



□ Invert Masks
☑ Use Background
☑ Enabled

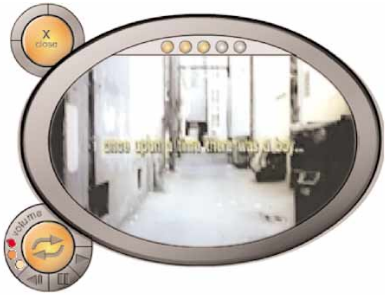
4. Click on the Close Button in the preview pane and drag it to the top left circle in the skin.

5. Click on white box above full screen. Click on “Enabled” to disable the full screen button.

6. Close the Media Skin Fast Track Editor.

7. Save your movie (Command-S).

8. Select File>Export Wired Movie and save your movie to your desktop. Locate and open MyFirstMovie.mov and enjoy!



Congratulations! You have just completed your first fully interactive QuickTime Movie!

Chapter 2

FastTracks™

FASTTRACK™ OVERVIEW

LiveStage Professional's new FastTracks™ are powerful drag and drop tracks that allow you to quickly create detailed media projects with no scripting. The FastTracks™ in version 4.0 include Media Skins, Playback Controls, VR Controls, and Status Bar FastTracks™. What would normally take multiple tracks and considerable scripting has been simplified into a straightforward script-free user interface. Using FastTracks™, you can use your own media to build full-screen movie trailers, corporate presentations, as well as sophisticated and functional movie elements.

WHEN TO USE A FASTTRACK™

FastTracks™ make it easy for novices and professionals alike to quickly construct functional interactive movie elements.

Use a FastTrack™ when you want to:

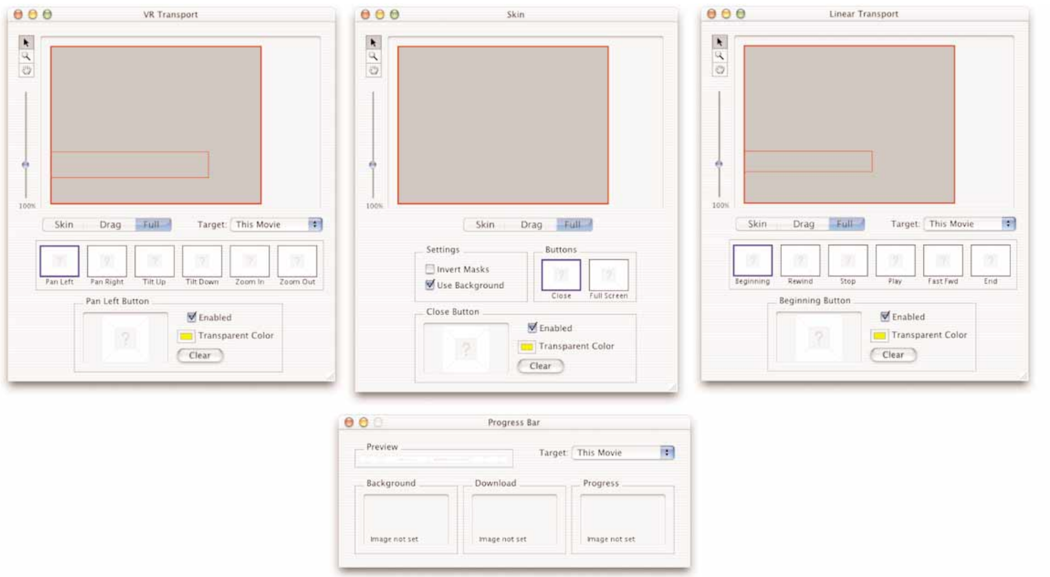
- create a 'skinned' player for your movie (Media Skin FastTrack™);
- quickly add playback controls to your movie without writing a single line of QScript (Playback Controls FastTrack™);
- add pan, tilt, and zoom controls to your VR movie, instead of using the less intuitive VR dragging (VR Controls FastTrack™);
- include a progress or status bar that displays the download and playback status of your movie (Status Bar FastTrack™).

WORKING WITH FASTTRACKS™

To create a FastTrack™, click on the applicable FastTrack™ button in the Track Creation Toolbar or select Create Track through the main Menu Bar and choose the FastTrack™ you wish to construct. You can also create FastTracks™ through a convenient contextual menu right in the Timeline. Simply control click in the Track List area (see Chapter 4—Introduction to Tracks, page 117) and select the FastTrack™ you want to create from the menu that appears. The track will appear in the Stage and in the Timeline, and the FastTrack™ Editor will open.

FastTrack™ Editors

Each FastTrack™ provides an intuitive FastTrack™ Editor to aid in fast project construction.



Target: This Movie

☑ Enabled

Transparent Color

Target: This Movie

☐ Invert Masks

☑ Use Background

☑ Enabled

Transparent Color

Target: This Movie

☑ Enabled

Transparent Color

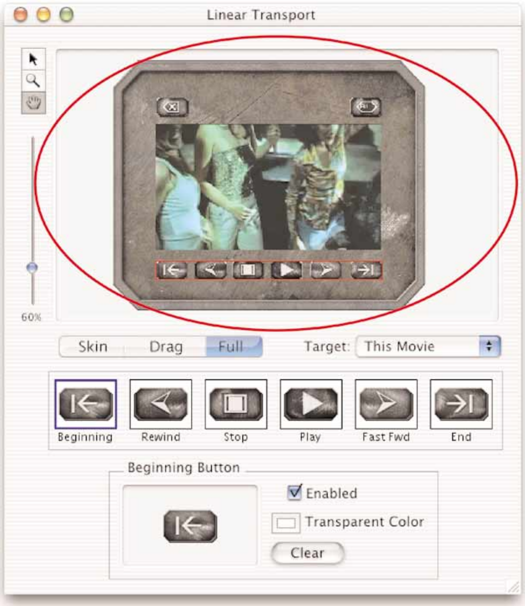
Target: This Movie

MEDIA SKINS, PLAYBACK CONTROLS, AND VR CONTROLS FASTTRACKS™

The Editors for the Media Skin, VR Controls, and Playback Controls FastTracks™ all contain a number of common features.

Preview Window

The Preview Window provides an accurate display of how your movie will look with the incorporation of the FastTrack™.



Target: This Movie

☑ Enabled

☐ Transparent Color

The FastTrack™ appears in the Preview Window as a red outline within which the FastTrack™ elements can be arranged. In the VR or Playback Controls FastTrack™, the overall track size will adjust to accommodate the final positions of the buttons. The Tools to the left of the window allow you to adjust the window's viewing area and manipulate the FastTrack™ elements.

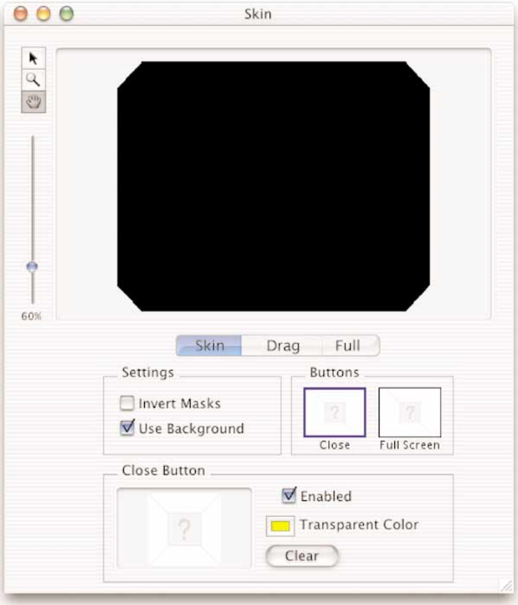
TOOL	FUNCTION
↖	Select and drag FastTrack™ buttons to position them.
🔍	Click on the preview to magnify the viewing area. Option-click on the preview to reduce the viewing area.
✋	Click on the viewing area and drag to position the movie in the viewing area.

In addition to these buttons, you will find a slider that you can drag to alter the magnification of the Preview Window's viewing area.

View Options

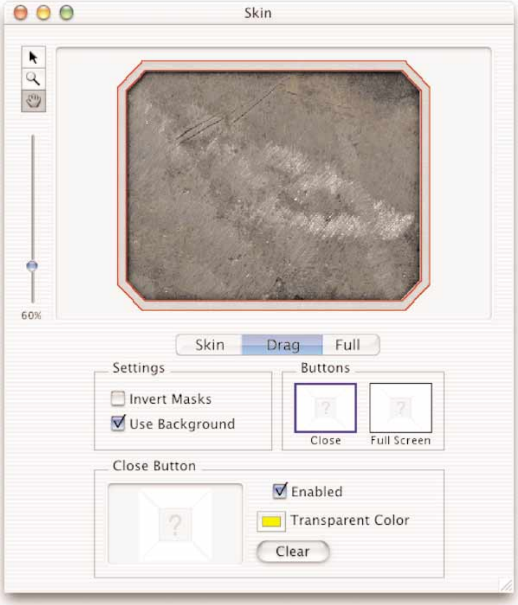
If you have added a Media Skin to your movie (see the Media Skin FastTrack™ section following this one), you can use the Skin, Drag, and Full buttons to determine how your FastTrack™ elements can best be laid out.

BUTTON	FUNCTION
Skin | Drag | Full	Displays the Window Shape of your movie's Media Skin, if applicable.



□ Invert Masks
☑ Use Background
☑ Enabled
🟨 Transparent Color

BUTTON	FUNCTION
Skin | Drag | Full	Displays the Drag Area of your movie's Media Skin in gray outlined in red, if applicable.



Settings

☐ Invert Masks

☑ Use Background

Buttons: Close, Full Screen

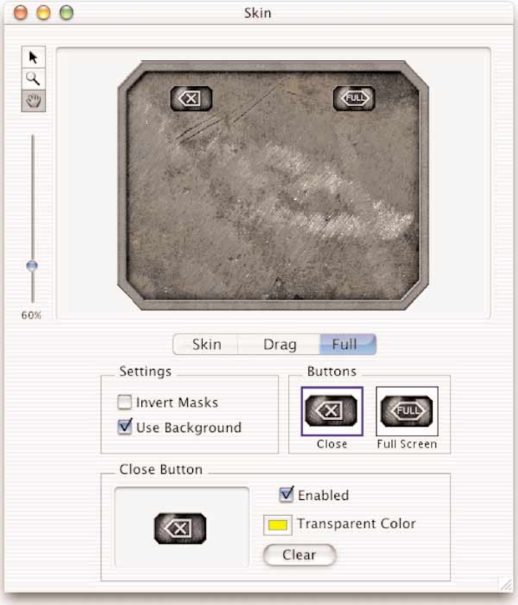
Close Button

☑ Enabled

Transparent Color

Clear

BUTTON	FUNCTION
Skin | Drag | Full	Displays the final effect of all included elements.



Settings

☐ Invert Masks

☑ Use Background

Buttons: Close, Full Screen

Close Button

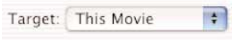
☑ Enabled

Transparent Color

Clear

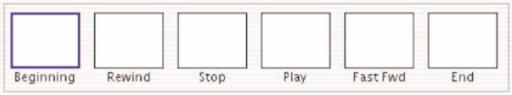
Target

When you are creating a Playback Controls, VR Controls, or Status Bar FastTrack™, you need to assign the target on which the FastTrack™ will operate. For example, if you have a VR Track called “MyVRTrack” that you want to pan, tilt, or zoom with a VR Controls FastTrack™, select “MyVRTrack” from the Target popup menu.


Target: This Movie

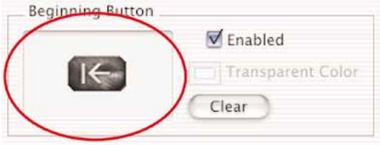
Button Frames

The Button Frames present the buttons that are available for a particular FastTrack™. When a button has been created, its Button Frame will display its ‘Up’ state image.



Button Image Well

This is where you create the various buttons in your FastTrack™. Select the appropriate Button Frame and then drag images from the Library window or your computer into this well to create ‘Up’, ‘Down’ and ‘Over’ states. The Button Image Well also acts as a functional preview to view the rollover states of your new buttons.


Beginning Button

☑ Enabled

□ Transparent Color

Enabled

When this option is unchecked, the selected button will be disabled and therefore will be neither present in the movie nor visible in the Preview Window.

Transparent Color

If you wish to make a color in your button transparent, you can select that color in this color swatch. Clicking on this swatch will present you with the standard Color Picker dialog, from which you can choose the desired transparent color. This option will not be available if your button images use an alpha channel.

Clear

Click this to remove the ‘Up’, ‘Down’, and ‘Over’ images that make up the selected Button.

tip

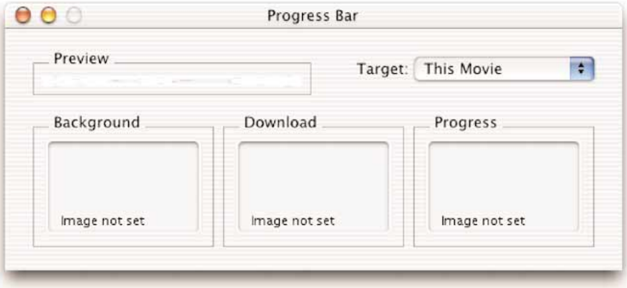
To automatically create ‘Up’, ‘Down’, and ‘Over’ button states with a single drag and drop operation, create your states on different layers in Photoshop and name the respective layers ‘up’, ‘down’, and ‘over’. Then drag the PhotoShop file directly to the appropriate Button Well to create the button, with all three states included. You can also create a folder and put the ‘up.xxx‘, “down.xxx‘, and ‘over.xxx’ image files in that folder and then just drag the entire folder into the button well. ‘xxx’ denotes the file type extension for a QuickTime compatible image format.

tip

On Macs, while in the color picker, you can option-click on any color on your screen to select it.

STATUS BAR FASTTRACK™

The Status Bar FastTrack™ Editor is a simplified version of the other FastTrack™ Editors. Because users do not interact with a status bar, it contains only image wells, and no buttons.



Preview

This frame presents a preview of what your status bar will look like when constructed. The preview will display with the download progress set to half and the play progress set to one quarter.

Target

The Status Bar FastTrack™ will follow and display the progress of the track specified in the Target popup. This must be either the current movie or a Movie Track.

Background

The image dragged into this well will be used as the background for the Status Bar.

The background is 10 pixels high and will be stretched to fit the width of the track. The image you drag into this well will be resized to fit these dimensions.

Download

The image in this well will be used to display the progress of the target track as it is downloading.

The download bar is 6 pixels high and will be stretched to fit the width of the track. The image you drag into this well will be resized to fit these dimensions.

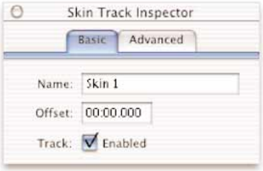
Progress

This image will be used to display the progress of the target track as it is playing.

The progress bar is 2 pixels high and will be stretched to fit the width of the track. The image you drag into this well will be resized to fit these dimensions.

FastTrack™ Inspector

The FastTrack™ Inspector allows a limited number of options to be adjusted. For a breakdown of the options available, see Chapter 3—Working in the LiveStage Professional Environment, page 106].



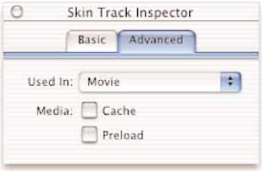
Skin Track Inspector

Basic | Advanced

Name: Skin 1

Offset: 00:00.000

Track: ☑ Enabled



Skin Track Inspector

Basic | Advanced

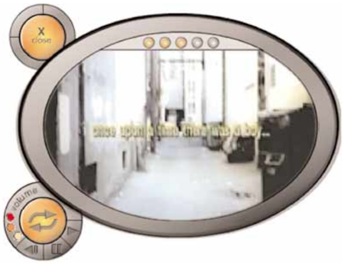
Used In: Movie

Media: □ Cache

□ Preload

MEDIA SKIN FASTTRACK™

Media Skins provide a distinct shape that customizes and enhances your movie. Instead of limiting your presentation to the browser, or confining your movie to a standard rectangular form in the QuickTime Player, you can design a uniquely shaped player of your own.



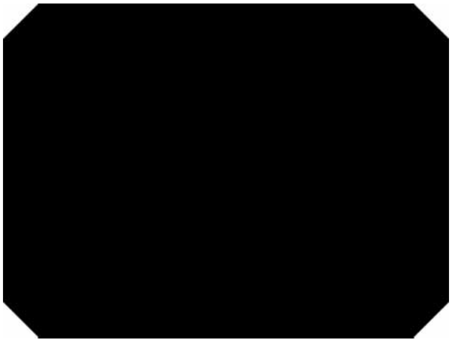
When designing your Media Skin, take into account the fact that your Window Shape image can only be a 1-bit black and white image. While you can use any shape you like, large sweeping curved lines can look jagged.

Media Skin Components

At their most basic, Media Skins are comprised of two primary components: the Window Shape and the Drag Area.

WINDOW SHAPE

The Window Shape image defines the actual shape of your finished movie, much like a cookie cutter defines a unique shape for a cookie. It can be made up of complex shapes and transparent areas. A Window Shape image is a black and white image that represents a silhouette of the window's shape. Black regions in the image represent the visible portions of the window and white regions represent transparent or non visible areas.

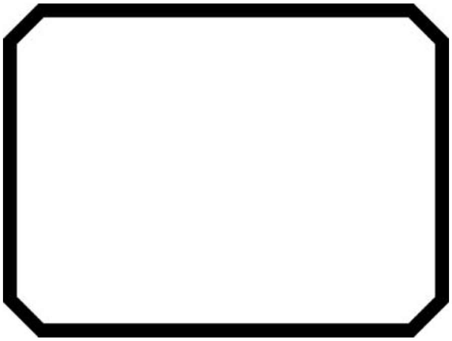


tip

The Window Shape image file and the Drag Area image file must have exactly the same dimensions.

DRAG AREA

The Drag Area image is also a simple black and white image that describes the area within the window shape that the user can click and drag to move the window. The black area in this image indicates the 'click and drag' region. This drag region, in effect, places a transparent mask over the movie and does not allow the user to interact with any media elements located under it. Make sure, therefore, that the black regions in your Drag Area image do not cover any interactive elements in your movie such as buttons.



Working with the Media Skin FastTrack™

The Media Skin FastTrack™ simplifies the skinning process. It automatically conforms its physical dimensions and duration to match those of the movie, so you do not have to worry about making manual adjustments. In addition, the Media Skin FastTrack™ Editor allows you to simply drag and drop media elements in place to automatically construct the Media Skin.

tip

A movie can only contain a single Media Skin that is present throughout the movie's duration. You cannot therefore, create additional Media Skin FastTracks™, nor can you adjust the track's duration.

CREATING A MEDIA SKIN FASTTRACK™

Prior to creating your Media Skin FastTrack™, you will need to create two black and white images in a graphics program such as Photoshop. One of the images defines the shape of your skin, and the other defines the drag area. Generally, you will want the two images to be identical in shape and size, the only difference being the exclusion of non-drag areas in the Drag Area image. The drag area MUST also be black in the shape image.

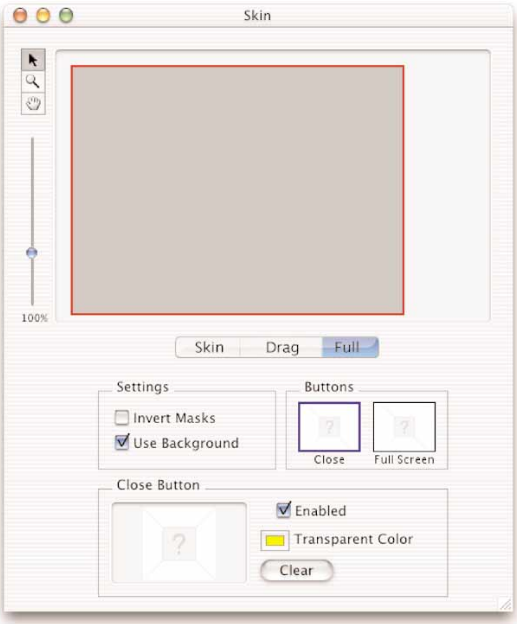
1 Create a new project and save it.

2. Drag your media elements into the Library folder that has been created in the directory where you have just saved your project.

3. Click on the 🧩 or select Create Track>Skin (Option-Command-S) to create a Media Skin FastTrack™.

   The Media Skin FastTrack™ Editor opens. A default skin appears in the editor.

Before bringing it into LiveStage Professional, crop the Window Shape image so that the black area of the Window Shape touches the border of the image. If there is some white between the black area and the border of the image, the skin will not work as expected.



☐ Invert Masks
☑ Use Background
☑ Enabled
Transparent Color

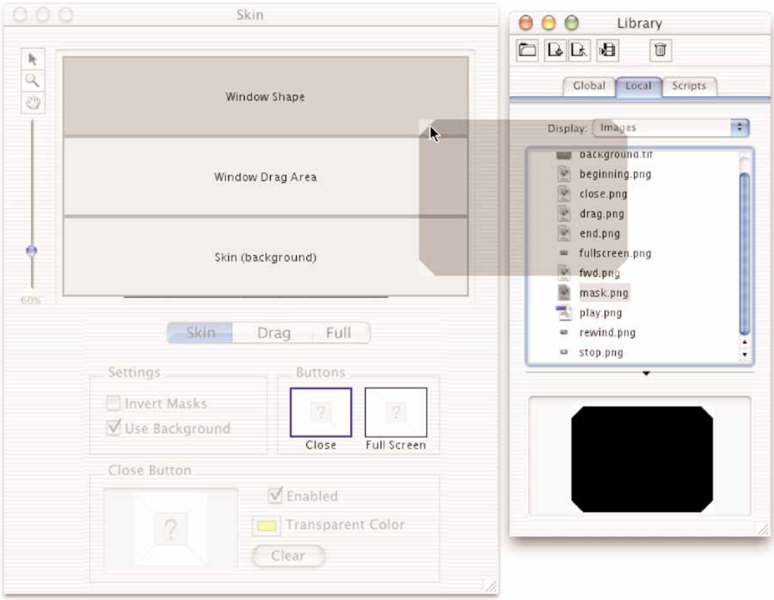
4. From the Local tab of the Library window, drag your Window Shape image to the Preview Window in the Media Skin FastTrack™ Editor.

   Three options appear over the window as you drag in the image:

   - Window Shape
   - Window Drag Area
   - Skin (Background)

The Global tab of the Library window contains a variety of media elements that you are free to incorporate into your project, including Window Shape and Drag Area images. These can be found in the Global tab in the Images>InterfaceElements>Skin folder.

5. Drop the image into the Window Shape option.



Display: Images

background.tif
beginning.png
close.png
drag.png
end.png
fullscreen.png
fwd.png
mask.png
play.png
rewind.png
stop.png

tip

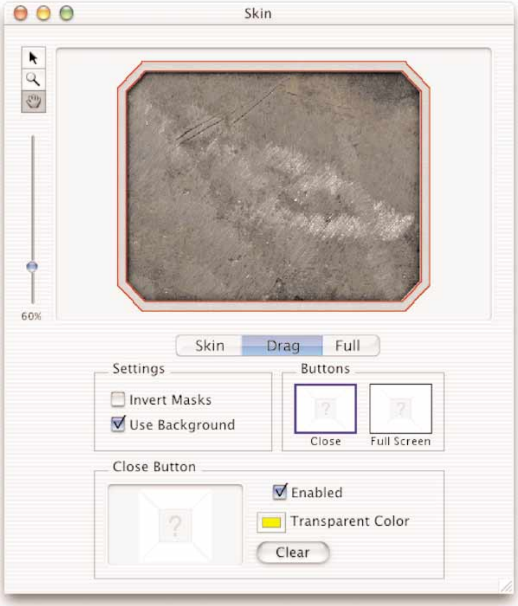
If you are using the Skin images provided in the Global tab of the Library, you will notice that many have dimensions as part of the image name (e.g., 6_192-144_drag.gif = 192 x 144 pixels). Whenever possible, select the size that is closest to your movie dimensions. While any image you select will scale to fit your movie, those that are closest in size will produce better results when curved edges are involved.

The Window Shape appears in gray in the Preview Window.

6. Now select the Drag Area image from the Library and drag it over the Preview Window.

7. Drop the Drag Area image into the Window Drag Area option.

8. Click the Drag button to see a representation of how the drag area will fit over the Skin.



□ Invert Masks
☑ Use Background

☑ Enabled
🟨 Transparent Color

A gray area outlined in red will indicate the drag area as it is superimposed over the Window Shape.

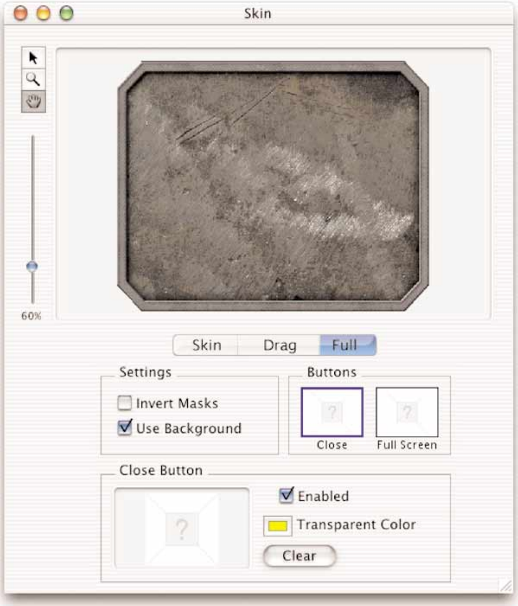
9. If you would like to use an image as the background for your movie, drag it from the Library into the Preview Window. Background images are optional.

10. Drop the background image on the Skin (Background) option.

    If the Full button is active, the Preview Window will display the completed effect that the Skin and Background images produce.

tip

You can create a skin with a single drag and drop operation using one of the following approaches. In Photoshop, create the Window Shape, Drag Area and Background on different layers, naming the layers as follows: ‘shape’, ‘drag’, and ‘background’. Then simply drag and drop the .psd file into the Preview Window of the Media Skin FastTrack™. Alternatively, you can place separate images into a folder using the same naming conventions (shape.xxx, drag.xxx, background.xxx) into a folder and then drag the folder to the Preview Window. This folder can also include all the buttons as outlined in the tip on the following page. The skin media in the Global Library demonstrates this. Just drag the whole Skin folder into the Preview Window.



Settings
□ Invert Masks
☑ Use Background

Close Button
☑ Enabled
🟨 Transparent Color

At any time, you can also click the Skin button to view just the Window Shape.

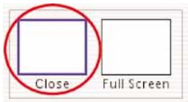
Adding Buttons

The Media Skin FastTrack™ Editor provides the means for you to easily construct Close and Full Screen buttons.

Close Button

Unlike movies presented in the QuickTime Player, movies with Media Skins do not have a controller that allows the user to close the movie. Using the Media Skin FastTrack™ Editor, you can create a fully functional Close button with Up, Down, and Over mouse states.

1. Select the Close Button Frame in the Media Skin FastTrack™ Editor.



The frame outlines in blue and the Button Well title displays ‘Close Button’.

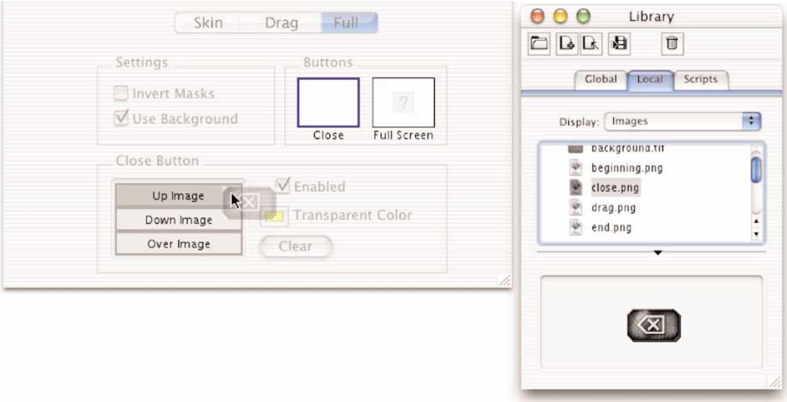
2. Select the image you wish to represent the ‘Up’ state for your Close button from the Library or your computer.

3. Drag the image to the Close Button Well.

   Three options appear as you drag the image into the well:

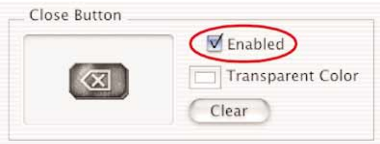
   - Up Image
   - Down Image
   - Over Image

4. Drop the image on the ‘Up Image’ option.

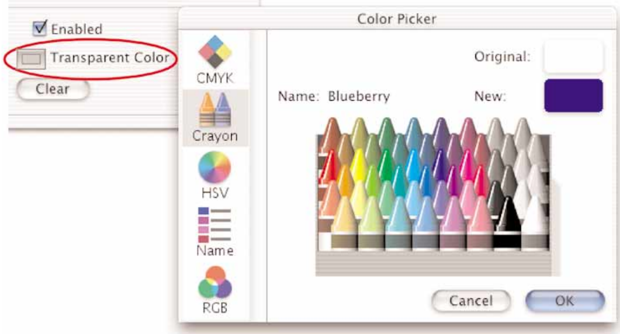


5. Follow Steps 2-4 for the ‘Down’ and ‘Over’ button states and drop each image on the appropriate option.

6. Make sure that the ‘Enabled’ option is checked.



7. If your button images have a background color that you do not want to appear in the movie, click on Transparent Color and select the precise color you want to render transparent. This is only enabled if your images do not contain an Alpha Channel. If an Alpha Channel is present, it will automatically be used.

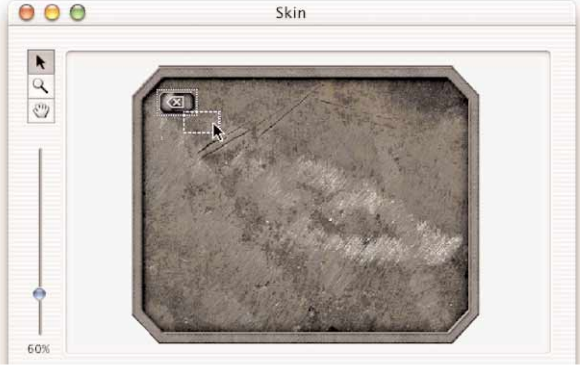


8. Click the ↖ from the tools to the left of the Preview Window.

tip

When creating your buttons, follow the simple naming convention of “up”, “down”, and “over”, followed by the file extension (e.g. up.gif or down.jpeg), then drag the entire folder containing your images directly to the Button Well. All three states will be created for you automatically. Each button will require its own folder with its own set of button states.

9. Click and drag your new Close button in the Preview Window to position it on your movie.



tip

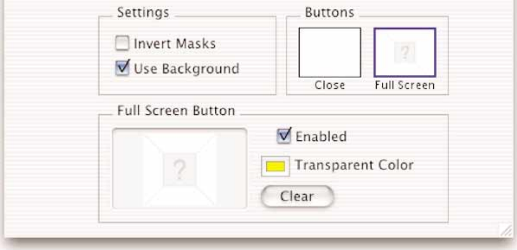
To build a skin complete with all buttons and their respective button states, in a single drag and drop, add a 'close' folder and a 'full' folder to the folder described in the tip on page 55. Within the close and full folders, place their respective 'up.xxx', 'down.xxx', and 'over.xxx' states. These images can be any image format that QuickTime recognizes, such as gifs, jpegs, pngs, etc. Drag this folder to the Editor. The Media Skin FastTrack™ will recognize the file names and apply them to the appropriate buttons and skin elements to automatically create the complete Media Skin.

Full Screen Button

QuickTime movies can be viewed Full Screen, meaning that the screen's background and the size of your movie is altered to fill the user's monitor. For complete information on the available Full Screen options when authoring your movie, see Chapter 15—Exporting your Movie, page 302.

1. Select the Full Screen Button Frame in the Media Skin FastTrack™ Editor.

   The frame outlines in blue, and the Button Well title displays 'Full Screen Button'.



Settings
□ Invert Masks
☑ Use Background

Buttons
Close
Full Screen

Full Screen Button
☑ Enabled
Transparent Color

2. Select the image you wish to represent the 'Up' state for your Full Screen button from the Library or your computer.

3. Drag the image to the Close Button Well.

   Three options appear as you drag the image into the well:

   - Up Image
   - Down Image
   - Over Image

4. Drop the image on the ‘Up Image’ option.

5. Follow Steps 2-4 for the ‘Down’ and ‘Over’ button states and drop each image on the appropriate option.

6. Make sure that the ‘Enabled’ option is checked.

7. If your button images have a background color you do not want to appear in the movie, click on Transparent Color and select the precise color that you want to render transparent. This is disabled if your image contains an Alpha Channel.

8. ↖ from the tools to the left of the Preview Window.

9. Click and drag your new Full Screen button in the Preview Window to position it on your movie.

tip

Media Skin FastTrack™ buttons appear by default near the top left corner of the movie’s rectangular dimensions. If the Window Shape that is used does not reach the top left corner, the newly created buttons will not be visible. Simply click in the top left corner of the Preview Window (first making sure that the button is enabled), and the button’s bounding box will appear. You can then drag and position it within your Media Skin.

Decompiling the Media Skin FastTrack™

The Media Skin FastTrack™, like all FastTracks™, is actually a compound track made up of several different LiveStage Professional tracks:

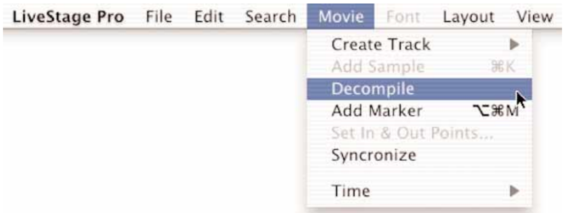
- a Skin Track to define the Window Shape and Drag Areas.
- one or more Sprite Tracks to add the scripting for the Close and Full Screen buttons.
- a Picture Track that holds the background image, if you have one.

‘Decompiling’ the Media Skin FastTrack™ breaks the track into its component tracks. This enables you to explore how the track works, and have more control over the inner workings of the simplified FastTrack™ process.

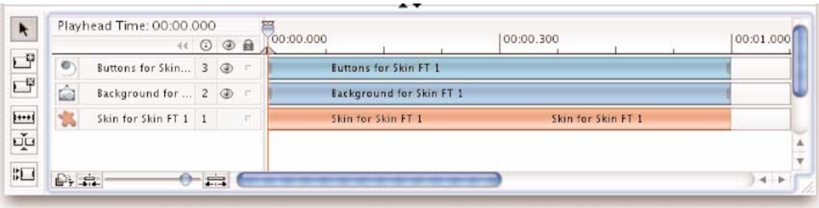
To decompile the Media Skin FastTrack™:

1. Click on the Media Skin FastTrack™ in the Timeline.

2. In the Menu Bar under Movie, select Decompile.



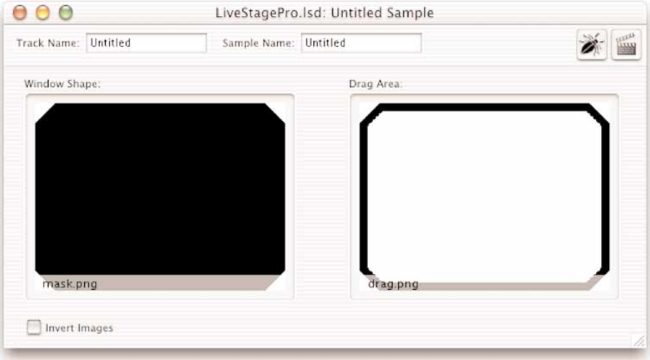
Several tracks are added to the Timeline: a Window Shape, one or two Sprite Tracks (if either the Close button, or the Full Screen button were present), and a Picture Track (if the Background image was present). Each can now be modified and augmented on their own.



The Sprite Track is covered in detail in Chapter 7 and the Picture Track is explained in Chapter 5.

THE (DECOMPILED) MEDIA SKIN TRACK

Double clicking on the Skin Track opens the Skin Sample Properties window.



LiveStagePro.lsd: Untitled Sample

Track Name: Untitled

Sample Name: Untitled

□ Invert Images

You can alter the name of both the Skin Track and the sample in the fields at the top of this window. The Skin Track sample presents a side-by-side view of the Window Shape and Drag Area images. You can view the original file names at the bottom of each window, and hold your mouse over the images to determine the path to their original files. Double clicking on either pane will open a full size preview of the image. If you select the Invert Images option at the bottom left of the Sample Properties window, the black areas of your images will become white, and the white will become black. This will change your window shape and drag areas accordingly, effectively reversing how they originally appeared.

PLAYBACK CONTROLS FASTTRACK™

Allowing the user to control the playback of your movie is one of the most basic ways to give movies some interactivity. Using the Playback Controls FastTrack™, you can add buttons that will allow the user to stop, play, fast-forward, rewind, and jump to the beginning and end of your movie. You can add as many or few of these options to your movie as you like.

The button options that are provided with the Playback Controls FastTrack™ are:

Beginning
When the user clicks this button, the movie will jump to its beginning and stop.

Rewind
On Mouse Click, the movie will visually and/or audibly play in reverse at double speed.

Stop
On Mouse Click, the movie will pause in place.

Play
On Mouse Click, the movie will play forward at normal speed from the point at which the Playhead was last stopped.

Fast Fwd
On Mouse Click, the movie will visually and/or audibly play forward at double speed.

End
On Mouse Click, the movie will jump to its end and stop.

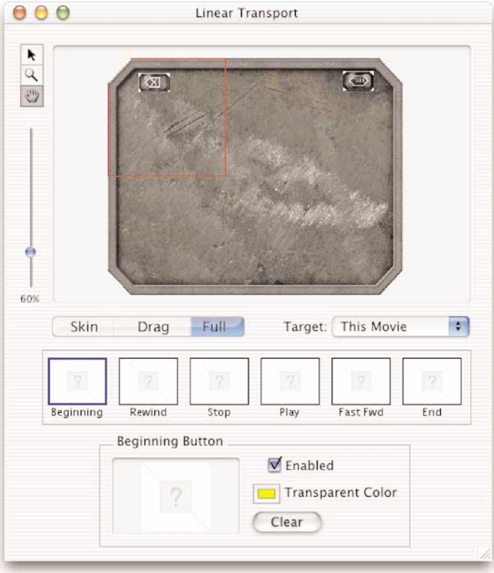
Working with the Playback Controls FastTrack™

As with the Media Skin, the duration of the Playback Controls FastTrack™ will automatically match your movie duration. The FastTrack's™ Preview Window provides visual and snap-to guides to assist you in precisely laying out the various buttons you create.

CREATING A PLAYBACK CONTROLS FASTTRACK™

1. Click on the or select Create Track>Playback to create a Playback Controls FastTrack™. You can also control the Track list of the Timeline and use the Contextual Menu to create the FastTrack™.

   The Playback Controls FastTrack™ Editor opens. Default buttons appear.


Target: This Movie
☑ Enabled

tip

Make sure that the original images you are using for your buttons are the size you want them to be in the final movie. Buttons are imported at their actual size and cannot be resized within the FastTrack™.

2. Select the Beginning Button Frame in the Playback Controls FastTrack™ Editor.

   The frame outlines in blue and the Button Well title displays ‘Beginning Button’.

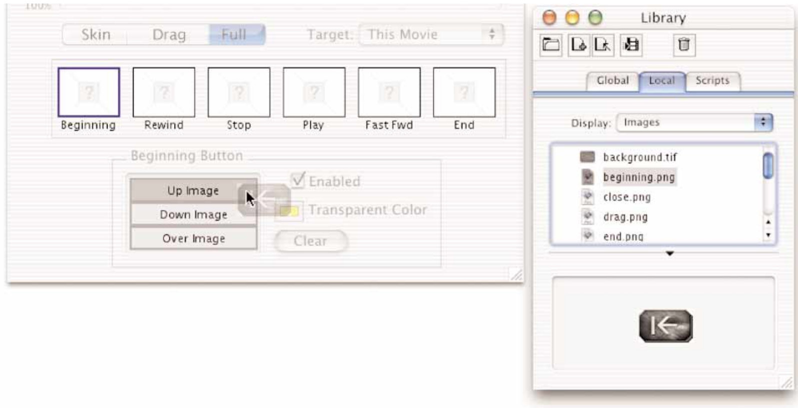
3. Select from the Library or your computer the image that you wish to use for your ‘Up’ state.

4. Drag the image to the Beginning Button Well.

   Three options appear as you drag the image into the well:

   - Up Image
   - Down Image
   - Over Image

5. Drop the image on the ‘Up Image’ option.



6. Follow Steps 3-5 for the ‘Down’ and ‘Over’ button states and drop each image on the appropriate option.

7. Make sure the ‘Enabled’ option is selected.

   As you enable each button, it will appear in the Preview Window.

8. Select a Transparent color if necessary to render the background of your buttons transparent if they do not already contain an Alpha Channel.

9. Repeat Steps 2-8 for each of the buttons you would like to include in your movie.

10. In the Preview Window click on the ↖.

11. Click on one of your buttons.

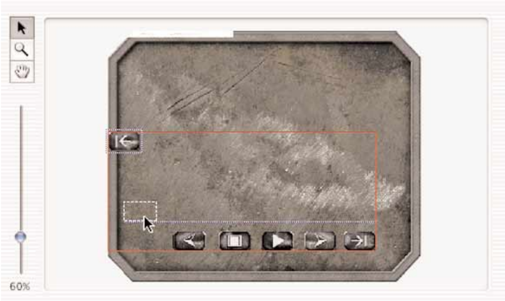
    The button is outlined in a bounding box.

tip

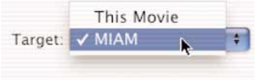
If you plan to incorporate a Media Skin and any other FastTracks™, it is best to create the Media Skin first. If you create and position the controls first, the Window Shape you use may cut off any ill-placed buttons. You will also want to ensure that your buttons do not fall within the Drag Area, as this would render them unclickable.

12. Click and drag the button to the desired location on your movie.

    The button will snap to the horizontal and vertical boundaries of each of the other buttons to enable you to easily align and position it. It will also snap to guides and to other elements in your movie, if present.



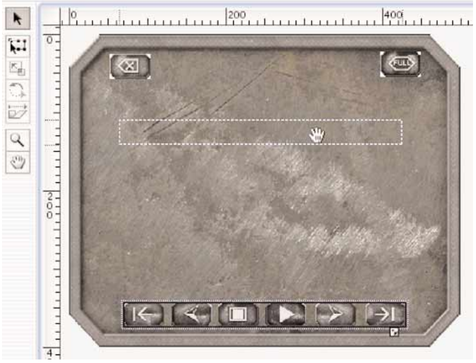
13. From the Target popup menu, set the track you want your Playback Controls FastTrack™ to control.



    While often you will just use the default ‘This Movie’, the Playback Controls FastTrack™ can also come in handy when you are building a Movie-in-a-Movie (see Chapter 10—The Movie Track, page [??].

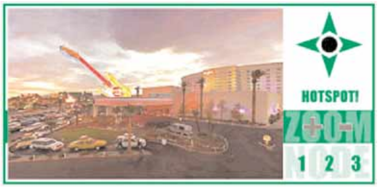
14. Close the FastTrack™ Editor and if necessary, return to the Stage to reposition the entire track.

    Click on the Playback Controls FastTrack™ in the Timeline or the Stage and then select the ↖ . On the Stage, drag and position the Playback Controls FastTrack™. You can also use the Sub-Object Editor (Chapter 7—The Sprite Track, page 173) to reposition the individual buttons.



VR CONTROLS FASTTRACK™

QuickTime VRs are tremendously immersive interactive experiences on their own (see Chapter 10—The VR Track, page 201). However, they can sometimes be less than intuitive to a user who has not yet been introduced to the technology. Unless users drag the mouse over one of these 'images', or something instructs them to do so, the immersive experience can be lost on them. Adding VR controls not only informs the user that interactivity is available, but can also create a far more engaging environment.



The VR controls that you can add through the VR Controls FastTrack™ are:

Pan Left, Right
This rotates the VR's viewing area horizontally to the left or right by 5 degrees on each Mouse Click.

Tilt Up, Down
This rotates the VR's viewing area vertically up or down by 5 degrees on each Mouse Click.

Zoom In, Out
This increases or decreases the VR's field of view by 5 degrees on each Mouse Click.

Working with the VR Controls FastTrack™

The VR Controls FastTrack™ is virtually identical to the Playback Controls FastTrack™, the only difference being the button functions. Again, the VR Controls FastTrack's™ duration will automatically match your movie duration. The VR Controls FastTrack's™ Preview Window, like that of the Playback Controls FastTrack™, provides visual and snap-to guides to assist you in precisely laying out your buttons.

CREATING A VR CONTROLS FASTTRACK™

1. Click on VR to create a VR Controls FastTrack™ or use the contextual menu by control-clicking in the Track List area in the Timeline.

   The VR Controls FastTrack™ Editor opens. Default buttons appear in the Editor.

2. Select the Pan Left Button Frame in the VR Controls FastTrack™ Editor.

   The frame outlines in blue and the Button Well title displays 'Pan Left Button'.

3. From the Library or your computer select the image you wish to represent the 'Up' state for your Pan Left button.

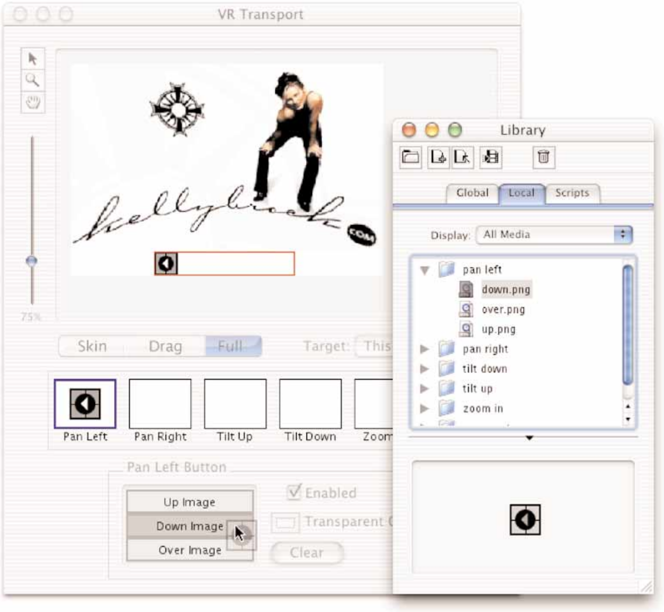
4. Drag the image to the Pan Left Button Well.

   Three options appear as you drag the image into the well:

   1. Up Image
   2. Down Image
   3. Over Image

5. Drop the image on the 'Up Image' option.

6. Follow Steps 3-5 for the 'Down' and 'Over' button states and drop each image on the appropriate option.



7. Make sure the 'Enabled' is selected.

   As you enable each button, it will appear in the Preview Window.

8. Select a Transparent color if necessary to render the background of your buttons transparent, provided they do not contain an Alpha Channel.

9. Repeat Steps 2-8 for each of the controls you would like to provide.

10. In the Preview Window click on the ↖ .

11. Click on one of your buttons.

    The button is outlined in a bounding box.

12. Click and drag the button to the desired location on your movie.

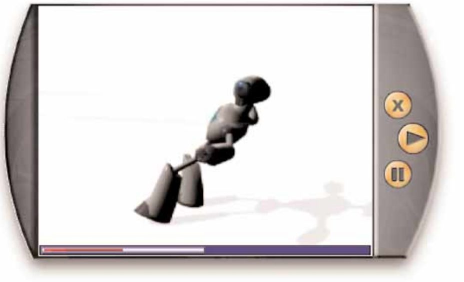
    The button will snap to the horizontal and vertical boundaries of each of the other buttons to enable you to easily align and position it. It will also snap to guides and to other elements in your movie, if present.

13. Close the FastTrack™ Editor and, if necessary, return to the stage to reposition the entire track.

14. Finally, from the Target popup menu set the QTVR Track that you want your VR Controls FastTrack™ to control.

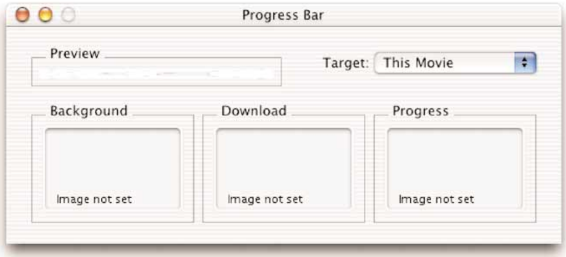
STATUS BAR FASTTRACK™

To provide users with as much information as possible when downloading and/or playing back your movie, you should include a Status Bar. The Status Bar consists of both a Download and Progress bar. The Download Bar displays how much of a file has downloaded to the user's machine if the movie is being served off the Web. This gives your audience an idea of the speed of the download process. The Progress Bar shows users where in the movie's timeline the Playhead is currently located. A Status Bar is also effective for CD ROM delivery to display how much of the movie is left to play.



Working with the Status Bar FastTrack™

The Status Bar FastTrack™ is set up somewhat differently from the other FastTracks™, as there are no buttons to be added. Instead, three simple image wells comprise the Status Bar FastTrack's™ Editor, and all three require images for your movie to compile.



Progress Bar

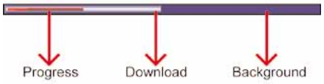
Preview

Target: This Movie

Background: Image not set

Download: Image not set

Progress: Image not set



BACKGROUND

The image dragged to this well will provide the background of the Status Bar, on top of which the Progress and Playback bars will advance. This image will be resized to the track if necessary.

DOWNLOAD

As the movie downloads to the user's computer, this image will expand to the right. When the movie has downloaded in its entirety the download bar will have reached the far right edge of the Status Bar. This image will be resized to fit within the background.

PROGRESS

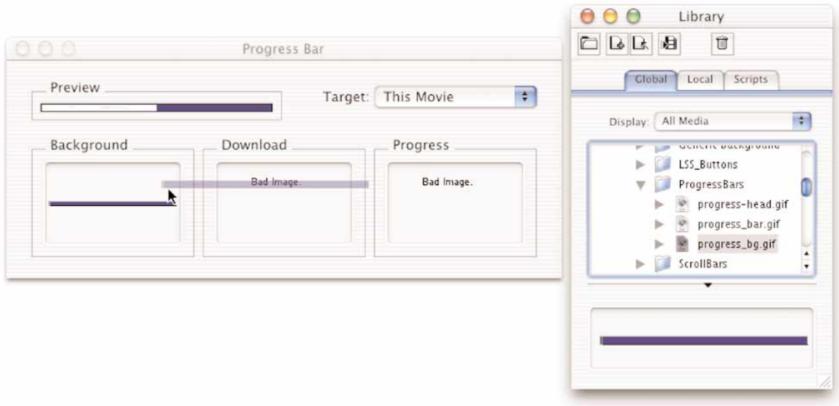
As the movie plays, this image will expand to the right. When the movie has completed its playback, it will have reached the far right edge of the Status Bar.This image will be resized to fit within the download image.

CREATING A STATUS BAR FASTTRACK™

1 Click on the or select Create Track>Status Bar to create a Status Bar FastTrack™. The Status Bar FastTrack™ Editor opens with default images.

2. From the Library or your computer, select the image you would like to act as the background of your Status Bar.

3. Drag it to the Background well.



4. Repeat Steps 2-3 for the Download image and the Progress image.

5. If necessary, select which Track's progress your Status Bar will follow from the Target popup menu.

6. Close the FastTrack™ Editor and if necessary, return to the stage to reposition the track.

FASTTRACKS™ WRAP UP

These new FastTracks™ will help you to quickly achieve sophisticated results that are standard in many real-world applications. You can see a number of impressive presentations that utilize the different elements that FastTracks™ build on StageDoor (http://stagedoor.totallyhip.com) in the Featured Sites Archive; and on the LiveStage Professional 4.0 CD-ROM check out the Hard Rock Café movie in the QTVR folder to see both a stylized Media Skin along with custom VR controls. They are engaging and eye-catching projects that could now be built with FastTracks™ in a fraction of the time they originally took.

Chapter 3

Working in the LiveStage Professional Environment

OVERVIEW

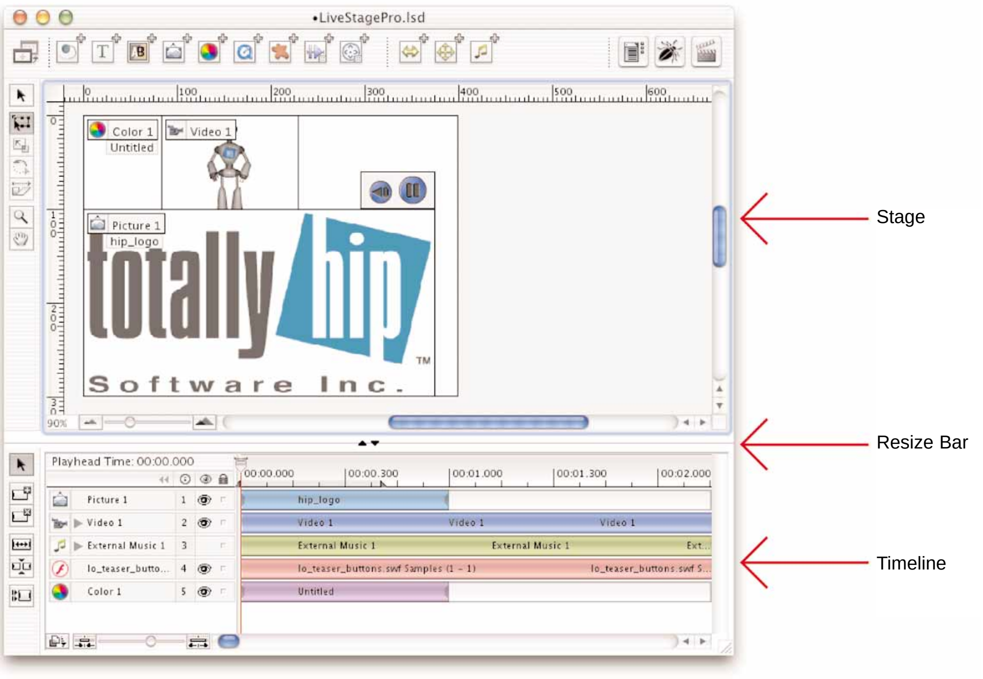
When you build a project in LiveStage Professional, most of your work will take place in the Document Window. You will also use two prominent supporting windows—the Library and the Inspector—to help you manage and modify your project.

tip

Window Users: Keyboard shortcuts presented in this User Guide are Mac-based. Whenever you see “Command-” substitute “Control-” to achieve the same functionality.

The Document Window

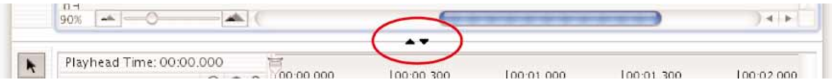
The Document Window is the primary window in LiveStage Professional. This is where you will lay out your project both visually and chronologically. The Document Window is separated into two main sections, the Stage and the Timeline, both of which provide a number of tools to help you work with your media elements.



ALTERING THE DOCUMENT WINDOW VIEW

You can customize the appearance of the Document Window. You may wish to emphasize one work area over another depending on the task you are currently working on in your project. For instance, you may wish to focus your attention on the Timeline and have it occupy the maximum amount of screen real estate available. Alternatively, you may want to zoom in tightly on the Stage but still see as much of your project as possible. The Resize bar makes it easy to hide either the Stage or the Timeline, as well as adjust the proportion of the window each occupies.

Click and drag the Resize Bar that separates the Stage from the Timeline to change their viewing areas. A simple click on the up triangle will hide the Stage. Similarly, clicking on the down triangle will hide the Timeline.



CUSTOMIZING THE DOCUMENT WINDOW VIEW

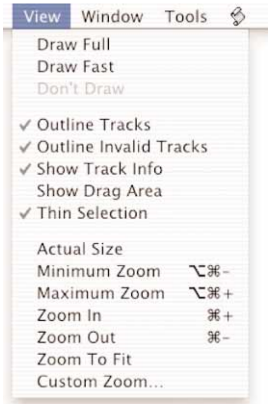
Once you have become accustomed to the different patterns of workflow in your projects, you may find that you use certain Document Window layouts repeatedly. LiveStage Professional makes it easy to set up customized views that you can then access with a simple click. Simply place the Resize Bar in the desired location, hold down the Command key, and press the F1 function key. Now whenever you press the F1 key, your screen will change to this preset customized view. You can resize and arrange all of LiveStage Professional's windows to suit your needs in this way, storing a different view in each of the 12 Function keys. The positions and views in each of these windows will be saved and restored as a complete set.

THE STAGE

The Stage is where you can visually lay out and modify your media. A variety of tools and functions are available in the Stage to assist you in building the look of your project. The Stage also serves as a display of what your movie will look like at any given point in time. For the most part, visible elements in the project appear on the Stage as they will appear in the final movie. Some notable exceptions to this rule are Movie Tracks and elements that are modified dynamically through scripting. You should always preview your movie when fine-tuning your project.

Viewing the Stage

There are several options in the View menu on the Menu Bar that will help you optimize the drawing on the Stage, based on your computer's processor power. This menu also provides options that specify the type of track information that is displayed within the Stage.



DRAW FULL

Select Draw Full to have the objects on the Stage drawn as accurately as possible. This produces output that is very close to the final QuickTime movie. Draw Full will draw elements according to their drawing modes, transparency settings, anti-aliasing, etc. This mode can be somewhat CPU intensive and should be reserved for faster machines.

DRAW FAST

Draw Fast displays objects as quickly as possible, prioritizing speed over accuracy. When you use this setting the Stage may not display the element's drawing modes, transparency settings, etc.

DON'T DRAW

This is the quickest draw setting because it effectively draws nothing. In order to use this setting and still be able to work with your elements on the Stage, make sure to select Outline Tracks so you can at least see their boundaries. This setting is only available when you are editing path tweens in the tween editor (see Chapter 13—Additional LiveStage Professional Tracks, page 259).

OUTLINE TRACKS

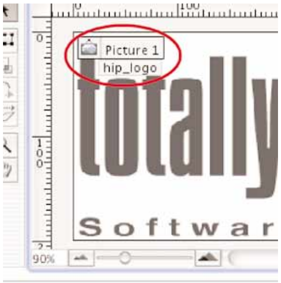
When Outline Tracks is set, an outline is drawn around each track on the Stage. Use this to see the full size of a track that has been set to use Transparent Draw Mode or when you use the Don't Draw option. Tracks that have backgrounds that blend in with other tracks will be easier to see as well.

OUTLINE INVALID TRACKS

This mode will outline a track in red when you are at a time in the movie when the track has no valid media samples.

SHOW TRACK INFO

This option displays the track icon, track name, and current sample name at the top left corner of each track on the Stage. If there is no sample present at the current time, only the icon and track name will be displayed. Some tracks, like the Movie track, will display additional information.

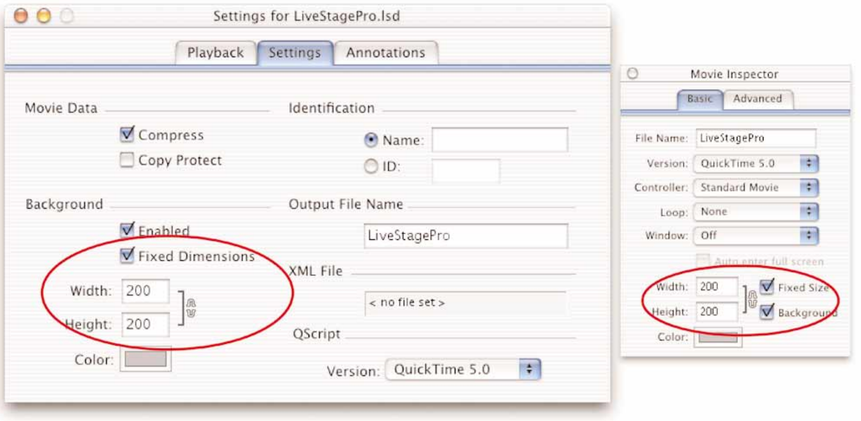


THIN SELECTION

When you select a track on the Stage, the outline that highlights the selection is thin. If you prefer a more prominent border, deselect Thin Selection. Thin Selection is the default and is particularly helpful when you are attempting to align tracks with each other, using the contents of the tracks to determine the correct positioning.

Working on the Stage

The Stage is the canvas where you create your movie. Like a canvas, you can have a solid color background or you can choose to leave it blank. Unlike a painter's canvas, the Stage's dimensions can be a fixed size or can change according to the elements you add and how you lay them out. These options can be set in either of two places: the Settings tab of the Document Settings window or in the Movie Inspector (Chapter 15 Exporting your Movie, page 299).



Movie Data
☑ Compress
□ Copy Protect

Identification
● Name:
○ ID:

Background
☑ Enabled
☑ Fixed Dimensions
Width: 200
Height: 200
Color:

Output File Name
LiveStagePro

XML File
< no file set >

QScript
Version: QuickTime 5.0

File Name: LiveStagePro
Version: QuickTime 5.0
Controller: Standard Movie
Loop: None
Window: Off
□ Auto enter full screen
Width: 200
Height: 200
☑ Fixed Size
☑ Background
Color:

BACKGROUND

Depending on how you design your movie, you may or may not require a background color, but the option is there if you need it. Your background will fill the entire Stage, regardless of whether its dimensions change, and you can choose your color from the many palettes.

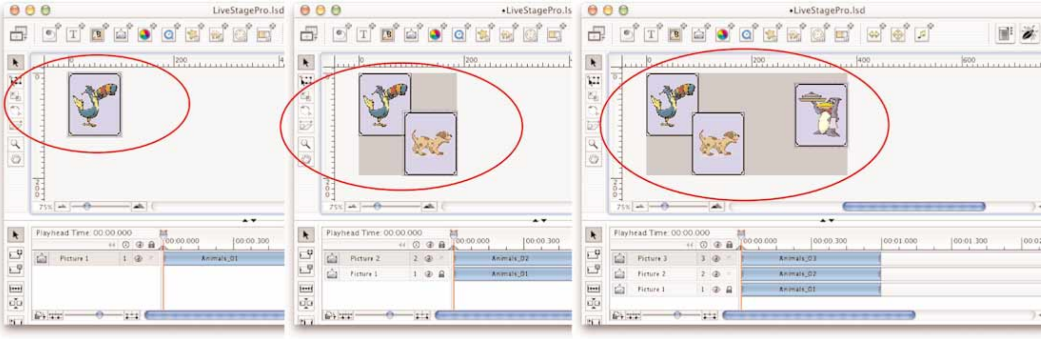
FIXED SIZE

A fixed-size Stage will maintain its dimensions regardless of the size of the media elements you bring into your project. If you are creating a movie that has to fit within a specified space, or for a particular monitor resolution, constraining your Stage to a Fixed Size makes it easy to stay within those dimensions.

It is important to understand how media reacts to a fixed size Stage. If your Stage has a fixed size of 200 pixels high by 200 pixels wide, any still images or Flash files larger than these dimensions will be resized and scaled to fit these fixed proportions. Media that is smaller than the fixed size will be unaffected. In addition, a Fixed Size Stage will confine the repositioning of a track to the Stage dimensions. You will not be able to move a track even partially outside a Fixed Size Stage.

DYNAMIC SIZE

Deselecting the Fixed Size option converts the Stage to dynamic sizing. This means that the Stage will conform to fit the media that is on the Stage. If you start your project by dragging in a 5-pixel by 5-pixel graphic, the Stage—and the resulting QuickTime movie—will be 5 pixels by 5 pixels. If you then drag in a 20-pixel by 20-pixel element, the Stage will become 20 pixels by 20 pixels. If you reposition that new element just below the 5-pixel graphic, the Stage will change to 25 pixels high by 20 pixels wide. No matter where you drag your media on a dynamically sized Stage, the Stage will resize to accommodate the outermost edges of all media elements.



ALTERING STAGE SIZING

You can switch your project from a Fixed Size Stage to a Dynamic Stage, or vice versa, at any point during production. You cannot choose a Fixed Stage that is smaller than the area your tracks currently cover. If you begin with Fixed and then switch to Dynamic, the Stage will resize to accommodate your media.

CROP MOVIE

Selecting the Edit>Crop Movie menu command reduces the dimensions of your Fixed Stage to the smallest size that will fully accommodate all the visual elements in your movie.

Creating Tracks

You can create tracks in one of three ways. You can use the Track Creation Toolbar, the Movie Menu, or you can simply drag your media directly into the Stage or Timeline.

TRACK CREATION TOOLBAR

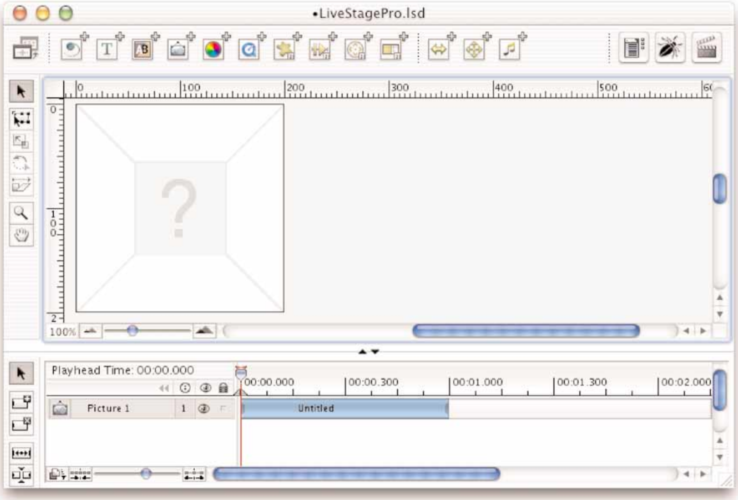
The Track Creation Toolbar located along the top of the Stage allows you to create a new track with a click of a button. The track appears in both the Timeline and the Stage, provided it is a visual track. Tracks with no visual elements, like a Sound Track, will appear only in the Timeline. When you create a visual track, it will appear on the Stage with an empty sample into which you can drag your media. FastTracks™ will automatically open a media editor for you to edit the track.

ICON	TRACK CREATED	ICON	TRACK CREATED	ICON	TRACK CREATED
	Sprite Track	T	Text Track		Playback Controls FastTrack™
	Effect Track		Picture Track		VR Controls FastTrack™
	Color Track		Movie Track		Status Bar FastTrack™
	Media Skin Fast Track		Tween Track		
	Modifier Track		Instrument Track		

STEP-BY-STEP: TRACK CREATION TOOLBAR

1. Click on the , or select Movie>Create Track>Picture (Option-Command-P) from the Menu Bar to create a Picture Track.

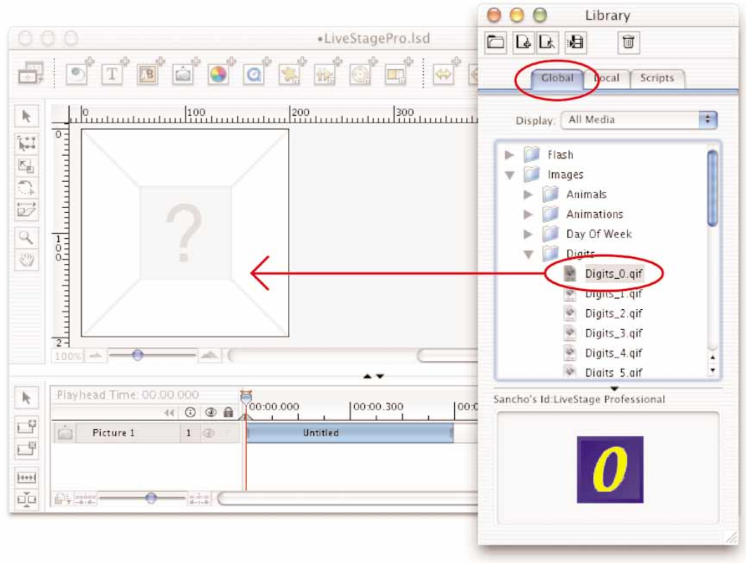
   An empty Picture Track measuring 200 pixels high and 200 pixels wide appears in the top left corner of the Stage background. This is the default size and position for newly created visual tracks.



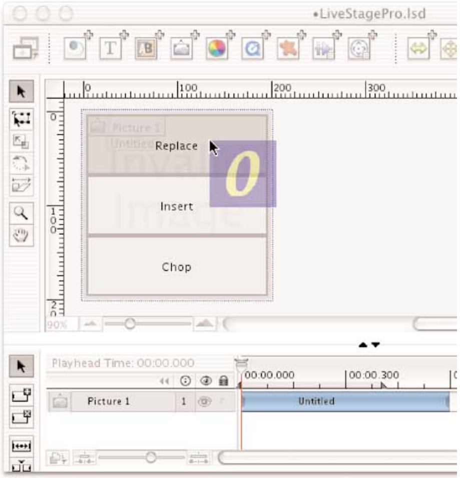
2. Open your Library window. Select Window>Show Library Window from the Menu Bar (Command-Y) to view the Library (detailed on page 99).

3. In the Library window, click on the Global tab and open the ‘Images’ folder.

4. Open the ‘Digits’ folder and choose one of the images listed in it. Drag it directly into the empty Picture Track on the Stage.



5. Three options appear as you enter the track on the Stage: Replace, Insert, and Chop. Drag the image and drop it on the Replace section.



This replaces the empty Picture Track's sample with this image. For more information on sample editing options, see the Editing Samples section on page 93 of this chapter.

MOVIE MENU

Creating tracks through the Movie menu located in the Menu Bar produces the same results as using the Track Creation Toolbar. A visual track with an empty sample appears in the Timeline and on the Stage at the top left corner. However, the Movie menu enables you to access and incorporate external files, like Streaming, Flash, QTVR, and Text Tracks. It is also through the Movie Menu that you can group multiple tracks together. To learn more about grouping tracks, go to Chapter 14—Working with Multiple Tracks on page 283.

CREATING TRACKS BY DRAGGING IN MEDIA

Dragging media elements from the Library window or anywhere else on your computer to the Stage or the Timeline is the quickest means of creating a track and sample.

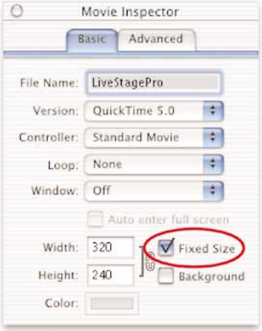
tip

To quickly create multiple tracks, drag a folder of media directly to the Stage or Timeline. Each media file will create a separate track. Alternately, shift-selecting a group of images and then dragging them in creates a single Picture track with multiple samples. Note: The elements must all be media formats that QuickTime understands, and the track dimensions will be those of the largest image.

STEP-BY-STEP: DRAGGING IN MEDIA

1. Open the Movie Inspector. Select Window>Show Inspector (Command-I) from the Menu Bar (Note: If you already have a track in your project, you may need to click outside of the track on the Stage for the Inspector to reflect the Movie's settings.)

2. Make sure that "Fixed Size" is checked. For more information on fixed vs. dynamic Stage dimensions, see the 'Preparing the Stage' section earlier in this chapter (page 72).



Movie Inspector

Basic | Advanced

File Name: LiveStagePro

Version: QuickTime 5.0

Controller: Standard Movie

Loop: None

Window: Off

☐ Auto enter full screen

Width: 320 ☑ Fixed Size

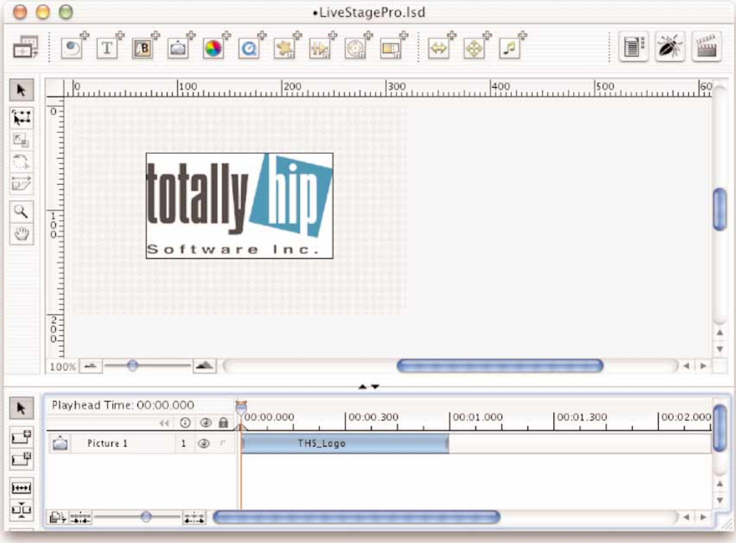
Height: 240 ☐ Background

Color:

3. In the Global tab of the Library, open the Images folder and from one of the folders available select any image. Drag it anywhere on the Stage.

   The image appears in the position where you dropped it on the Stage, provided it falls within the bounds of the Stage background. Otherwise, it will appear at the closest position to the point it was dropped, within the bounds of the background.

   The track and sample show up in the Timeline.



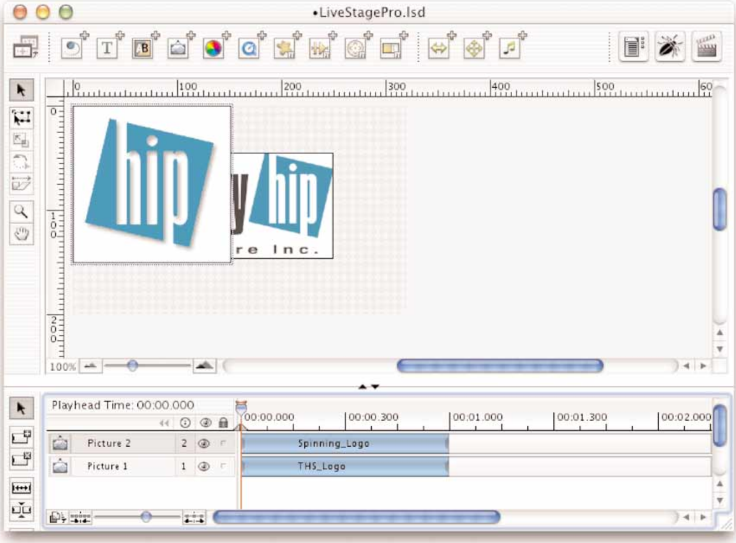
•LiveStagePro.lsd

Playhead Time: 00:00.000

Picture 1	1	THS_Logo

4. Now select a different image from the Library, and this time drag it to the Timeline instead of the Stage.

   The track appears in the Timeline and the image appears in the top left corner of the Stage. The position Left=o, Top=o is the default starting location for media that is dragged to the Timeline.



Laying out Media Elements on the Stage

Aside from providing a visual representation of your movie, the Stage can also be used to manipulate media elements.

TOOLBAR SWITCH BUTTON

Clicking the Switch Toolbar button switches the active toolbar between the Track Creation Toolbar and the Alignment Toolbar.

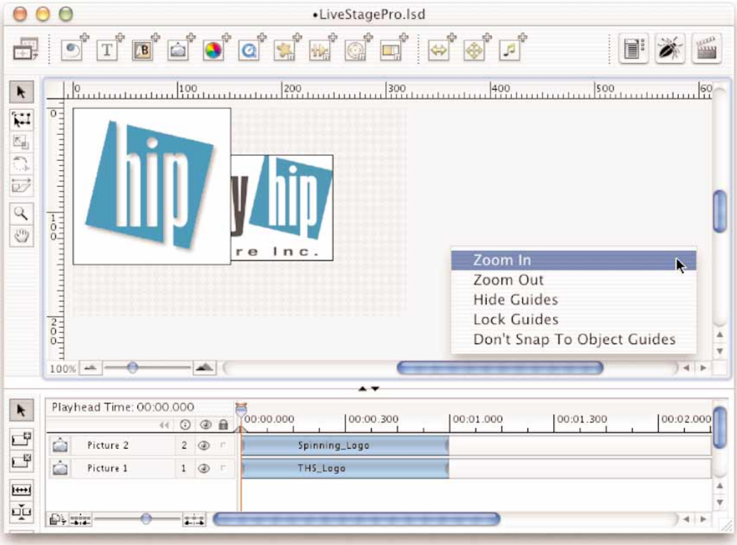
ALIGNMENT TOOLBAR

The alignment tools in this toolbar, common to most graphics software packages, will help you precisely position multiple items on the Stage. Hold down the Shift key to select multiple media elements at once, and then click the appropriate alignment tool.

ICON	FUNCTION
	Aligns all selected media to the left edge of the left-most selected element.
	Aligns all selected media to the right edge of the right-most selected element.
	Aligns all selected media to the top edge of the top-most selected element.
	Aligns all selected media to the bottom edge of the bottom-most selected element.
	Aligns the horizontal center points of all selected media to the average horizontal center point of the media.
	Aligns the vertical center points of all selected media to the average vertical center point of the media.
	Evenly distributes the selected media horizontally from the left-most selected element to the right-most selected element according to their horizontal center points.
	Evenly distributes the selected media vertically from the top-most selected element to the bottom-most selected element according to their vertical center points.

CONTEXTUAL MENU: STAGE

A handy contextual menu is available to you with a control-click or right-click inside the Stage area. You can select from Zoom In or Out, as well as change the status of guides (see ‘Guides’ on page 84 of this chapter) quickly and easily.



Modifying Elements on the Stage

Common functions such as cutting, copying, pasting, duplicating, and deleting function the much same way as all standard applications.

MENU COMMAND	SHORTCUT KEY	FUNCTION
Edit>Copy	Command-C	Copies selected object; object remains in place
Edit>Cut	Command-X	Copies selected object; object is removed.
Edit>Paste	Command-V	Places cut or copied object at insertion point.
Edit>Duplicate	Command-D	Copies and pastes selected object in place.

While LiveStage Professional is not a media editor, certain adjustments can be made to media right on the Stage. You can modify the size and position of a track, and in the case of Sprite Tracks, position, distort, and rotate individual sprites within the track using the Transformation tools. You can also zoom in and scroll the Stage itself.

tip

Holding down shift while you rotate a sprite, will snap the rotation at 30° and 45° increments.

TRANSFORMATION TOOLS

Transformation Tools are primarily for use on sprites, but the Positioning and Resize tools will work on any visual track.

ICON	FUNCTION
↖	Selects and positions individual elements on the Stage. Select multiple elements by Shift-clicking.
⬚	Resizes a selection by dragging on handles.
⤡	Scales a sprite selection by dragging on handles.
↻	Rotates a sprite selection by dragging on a rotate control handle.
▱	Skews a sprite selection by dragging on handles.
🔍	Zooms in on Stage by clicking. Zooms out on Stage by Option-clicking.
✋	Moves Stage around by dragging.

CONTEXTUAL MENU: TRACKS ON STAGE

Control-clicking on individual tracks on the Stage brings up a useful contextual menu that can delete, hide or lock a track, as well as set its enabled state.

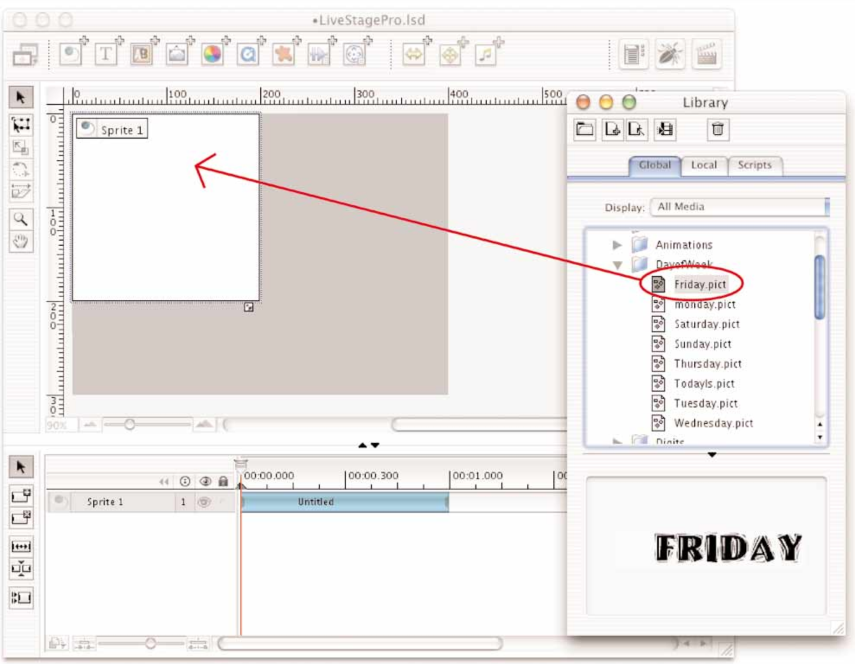
STEP-BY-STEP: MODIFYING ELEMENTS ON THE STAGE

1. Create a Sprite Track by clicking on the Sprite Track button in the Track Creation toolbar

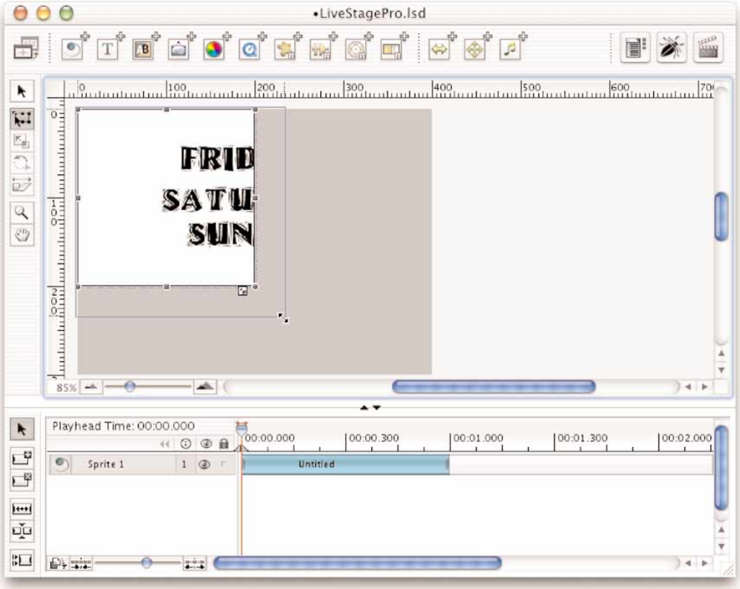
   An empty Sprite Track appears on the Stage.

2. Open the Images folder in the Global tab of the Library window. From the Days of the Week folder drag 'Friday.pict', 'Saturday.pict' and 'Sunday.pict' into the Sprite Track on the Stage. You have just created three sprites.

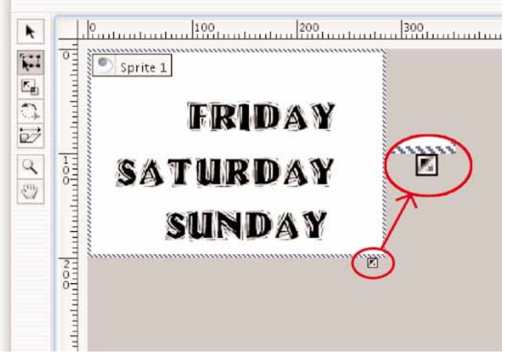
   These sprites are larger than the Sprite Track and therefore cannot be seen in their entirety. You can resize the track in order to see them.



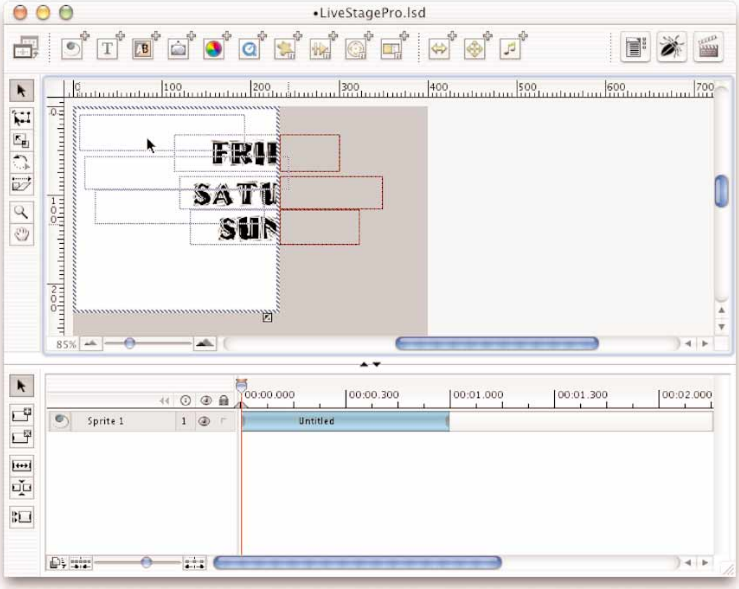
3. Click on the tool. Handles appear around the Sprite Track. Drag the bottom right corner handle to the right until all the sprites are visible.



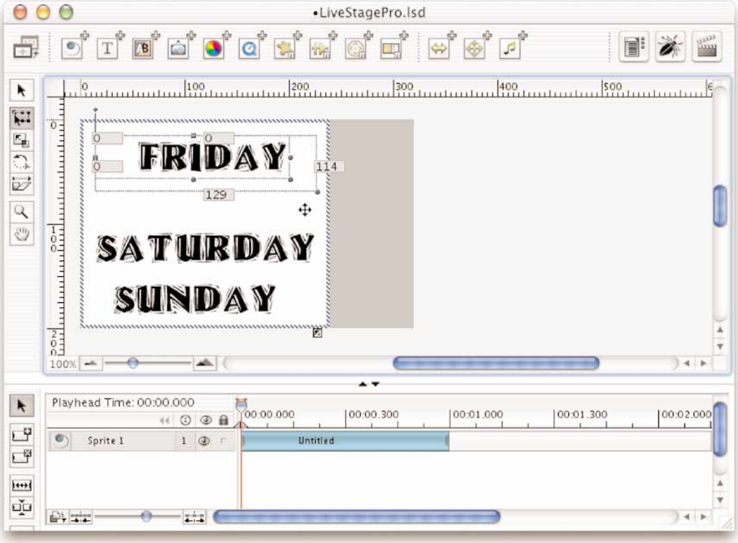
4. Click on the box at the bottom right corner of the Sprite Track. This allows you to edit the individual sprites within the Sprite Track through the Sub-Object Editor.



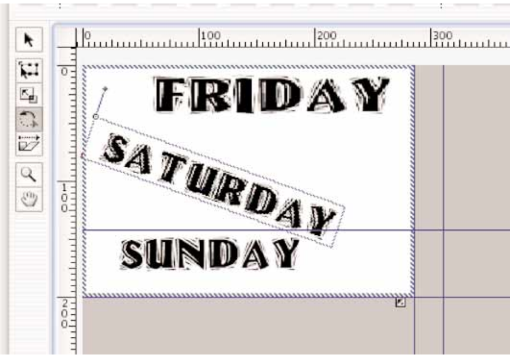
5. Click on the Positioning tool and Shift-click the three sprites to select them all. Drag them towards the top left corner of the Sprite Track.



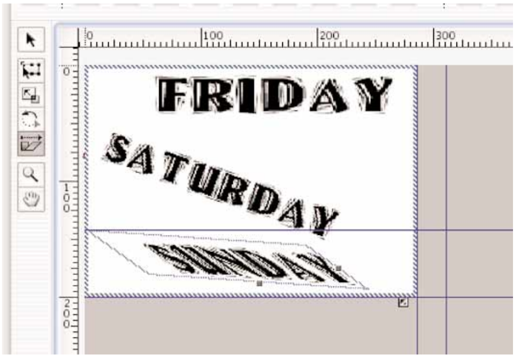
6. Click on the Scale tool and then select “Friday.” Drag one of the handles and scale the sprite slightly larger.



7. Now select the Rotation tool and click on “Saturday.” Click and drag on the handle that appears at the top left corner and rotate “Saturday” to any angle.

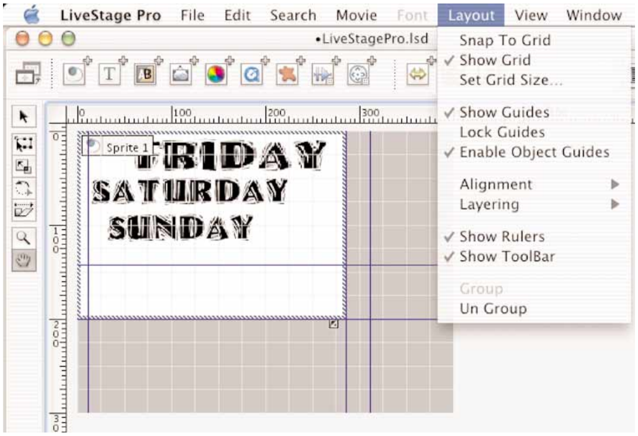


8. Finally, click on the Skew tool and select “Sunday.” Click and drag the handle to skew the image. Skewing an image can give it perspective.



Layout Aids

LiveStage Professional provides tools to assist you in correctly positioning your media and interactive elements. The Layout menu on the Menu Bar provides grids, guides, grouping, alignment, and layering controls. Sophisticated positioning information on the Stage helps take the guesswork out of arranging your media.



RULERS

The Stage is bounded on the top and left by rulers. The units of measurement are pixels. Depending on how zoomed in or out you are on the Stage, the units will change accordingly. Rulers can be turned on and off as you desire through the Layout>Show Rulers menu option.

GRID

The grid provides a visible structure against which you can arrange your media. You can use the grid solely as a visual guide, or you can set your media to 'snap' precisely to the grid lines as you drag. The default grid is set to 25-pixel units, but you can adjust the grid size and even the grid color to suit your project's needs through the Layout>Set Grid Size menu option. You can also hide the grid so that you can still snap to the grid without seeing it.

GUIDES

To snap items to exact positions that do not fall along pre-existing gridlines, try using guides. Through the Layout menu in the Menu Bar you can:

- turn guides on or off;
- show or hide them; or
- lock them in place.

With 'Show Guides' selected, simply click either of the rulers to pick up a guide and, with the mouse still down, drag the newly-created guide to the Stage and position it where you wish. Clicking and dragging right from the left ruler yields a vertical guide, clicking and dragging down from the top ruler yields a horizontal guide. Positioning information, providing the distance of the guides from the left/right or top/bottom bounds of the background, assist you in the precise placement of your guides. To remove a guide, simply drag it off the Stage.

The 'Enable Object Guides' option in the Layout menu allows you to snap selected media elements to other media elements. This allows you to align elements quickly and to easily position an element vertically or horizontally.

LAYERING

Layering is an important element when creating QuickTime movies with LiveStage Professional. You can move a selected item in front of or behind other elements on the Stage using the 'Move Forward', 'Move Backward', 'Move to Front' and 'Move to Back' options in the Layering section of the Layout menu. This changes the Drawing Layer of the selection. To learn more about Drawing Layers and how to work with them, see Chapter 14—Working with Multiple Tracks, on page 281.

GROUPING

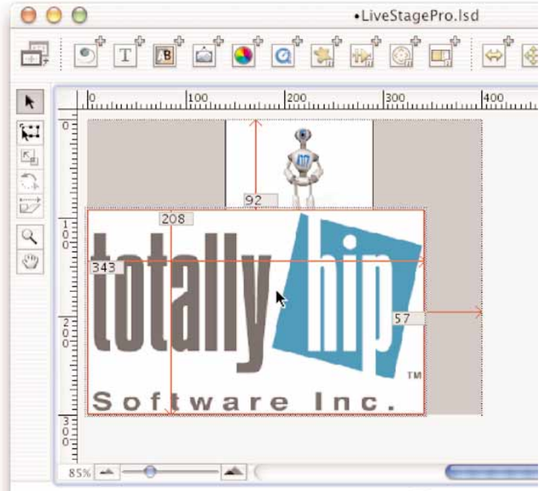
Grouping tracks together allows you to move and resize them as a single unit. On a Fixed Size Stage, individual tracks will only move as far as the Stage background bounds, which can change the relative positioning of the grouped tracks. Shift-click on the items in the

Stage you wish to group and then select ‘Group’ from the Layout Menu. Select Ungroup to break them apart again.

TRACK LOCATION COORDINATES

When a track’s position in relation to other objects in the movie must be precise, LiveStage Professional’s track location coordinates can be highly useful. At a glance, you can view the dimensions of a selected track as well as its distance from other objects on the Stage.

Using the arrow tool, select a track, hold down the Option key, and click on the track to view the track’s dimensions, as well as the horizontal and vertical distances to the edges of the Stage Background. Continue to hold the Option key and move your mouse over another object to view the distances from the selected track to the nearest boundaries of that object.



THE TIMELINE

The Timeline section of the Document Window is a ruler that measures time and displays a chronological view of your project’s tracks and samples. This is where you arrange your tracks and samples over time. Just as the Stage provides spatial information and tools to assist the physical layout of a project, the Timeline provides time-based information and tools to assist the sequential layout of a project.

Timeline Views

There are three alternatives for viewing your tracks in the Timeline’s Track List: Load Order, Layer Order, and Custom Order. Each view displays the track order according to these different construction considerations. To select the view you wish to access, click on the at the bottom left corner of the Document Window.

LOAD ORDER

Load Order will display your tracks in succession according to their Index Number, or the order in which they will be loaded into your QuickTime movie. Index Numbers are assigned to tracks automatically in the order in which the track is created, but can be changed by selecting and dragging a track to a different location in the Track List.

Load order is an important consideration in achieving optimum playback of your movies over the Web. For more information on utilizing Load Order to optimize your projects, go to Chapter 14—Working with Multiple Tracks on page 281.

LAYER ORDER

Another obvious consideration in designing and building your projects is layering your media visually. Moving media elements in front of or behind other elements is accomplished by changing the element's Drawing Layer (see page 281 in Chapter 14—Working with Multiple Tracks). Drawing Layer values are assigned to tracks automatically in descending order as each track is created, with the last track created having the lowest Drawing Layer value and appearing in the foreground. The Drawing Layer can be altered while in the Layer Order view by selecting and dragging tracks to different positions within the list.

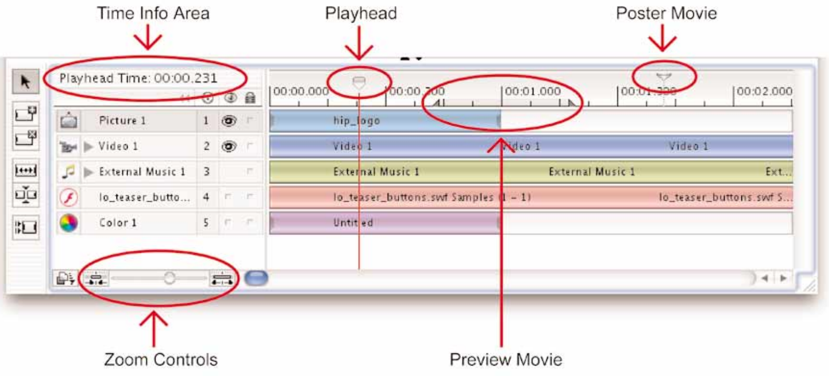
Layer Order view displays tracks in the foreground at the top of the Track List.

CUSTOM ORDER

The Custom Order view allows you to organize your tracks to suit your own project management needs, and has no visual or constructional effect on the movie.

TIMELINE AIDS

The Timeline has a host of tools and features that help pinpoint precise times in your movie sequence, and perform a variety of editing functions.



Playhead Time: 00:00.231

TIME INFO AREA

This area displays the current time in the movie according to the position along the Timeline of either the Playhead or the Mouse, if the Mouse is within the Timeline ruler. Time is specified as minutes, seconds, and 600ths of a second. So a full second would appear as 00:01.000, and a half second would be 00:00.300.

THE PLAYHEAD

The Playhead consists of a pointer and a vertical red line that you can drag throughout the duration of your movie. As the Playhead is dragged, the current time appears in the Time Info Area, and the Stage updates to reflect the movie's visual appearance at that point in time. Clicking the Mouse anywhere along the Timeline will move the Playhead to that precise moment in the movie.

PREVIEW MOVIE INDICATOR

A Preview Movie is a very short version of the full movie, usually less than 5 seconds in duration, which gives the user an idea of the movie's content. The Preview Movie will display when the user selects the movie file in an "Open File" dialog window or in other applications that provide thumbnail previews. To specify a brief segment of your movie to play as a Preview Movie, drag the start and finish indicators to the desired locations on the Timeline.

POSTER FRAME INDICATOR

Similar to the Preview Movie, a Poster Frame is a single frame of your movie that displays in open file dialog windows and other applications that provide thumbnail previews. Drag this indicator to the appropriate time in the Timeline to select the frame.

ZOOM CONTROLS

The Zoom Controls at the bottom left edge of the Document Window enable you to adjust the scale of the Timeline. You can either click the Zoom buttons or use the slider to achieve the desired view.

tip

Open a preview by double clicking on a video or audio track's media in the timeline.

ADD MARKER

You can add a marker to any track to provide a visual label or flag to assist you as you work in the Timeline. You can add a marker to a selected track by positioning the Playhead at the point in time you desire and selecting Movie>Add Marker (Option-Command-M). Markers can be also added 'on the fly' to Video Tracks while previewing a video sample by pressing the letter M on your keyboard as the preview progresses. You can also add a marker by control-clicking on the track and using the contextual menu that appears.

TIME

You can use the Time options in the Movie menu to move the Playhead to particular points in your Timeline.

Go To Start

This returns the Playhead to the beginning of your movie.

Go to End

This advances the Playhead to the end of your movie.

Go to Next Sample

This jumps the Playhead to the beginning of the next sample that appears in any track on the Timeline.

Go to Previous Sample

This moves the Playhead to the beginning of the previous sample that appears in any track on the Timeline.

Go to Time

Select this and enter the precise point in time that you wish the Playhead to move to.

Track and Sample Modification Tools

In addition to the Menu bar commands, LiveStage Professional provides a number of tools that can modify your samples and tracks with a click.

ICON	TOOL NAME	FUNCTION
↖	Selection Tool	Selects tracks and samples. Click-drag moves and resizes samples, and repositions tracks in the Track List.
	Sample Creation Tool	Click to create a sample one second in length at the Mouse's position in the selected track. Click and drag to create a sample of a specific duration.
	Sample Deletion Tool	Deletes the sample that is clicked.
|↔|	Sample Expansion Tool	Expands a sample to meet the nearest sample(s) in the track with a click. Clicking the left edge expands the sample to the left; right edge expands to the right; middle expands in both directions.
	Sample Split Tool	Splits a sample at the Mouse's position.
	Track Manipulation Tool	Enables modification of the track's offset or duration.

CONTEXTUAL MENUS

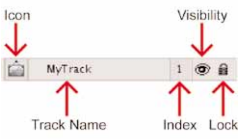
Two Contextual Menus are available in the Timeline to streamline workflow:

- Control-click or right-click in the Track Header to display a contextual menu which allows you to create a sample, marker, or any type of track. You can also disable, lock, or delete a track, as well as access the Track's Properties window through this menu.
- Control- or right-click in the Sample Bar to display a contextual menu which allows you to create and delete samples, or access the Sample Properties window. You can also disable, lock, edit and delete tracks and create markers through the contextual menu as well.

Tracks in the Timeline

Once you have created a track through the Track Creation Toolbar, Movie menu or by dragging media in, it appears in the Timeline in two segments: the Track Header and the Sample Bar.

TRACK HEADER



The Track Header is the portion of the track located to the left on the Timeline. It contains the track type icon, track name, Index Number, and Stage Visibility and Stage Lock controls. You can drag the separator bar between the Track Header and the Sample Bar to resize the Header. You can also click on the arrows at the top of the Track List to hide the visibility and lock controls, making more room to view your Sample Bars.

Stage Visibility

You can hide and show a track on the Stage by clicking on this icon. This only affects visibility on the Stage and does not affect the track in the final movie. This is useful when you want to work on a track that is concealed by other tracks.

Stage Lock

Clicking on this icon locks the track and prevents you from moving it on the Stage

SAMPLE BAR

The Sample Bar is where you manipulate the time-based properties of your track, as well as access and modify the media contained in the track.

TRACK LIST

The left side of the Timeline that contains all of the Track Headers is called the Track List. The tracks within your project are displayed according to the Timeline View you have selected. You can adjust the width of this list by dragging the bar that separates the Track List from the tracks. You can also reorder the Track List simply by selecting and dragging one or more tracks to a new location in the list. Dragging a track within the Track List will affect your project differently depending on which Timeline View you are currently working in.

WORKING WITH TRACKS AND SAMPLES

When you create a new track, an initial sample is created automatically. In most cases, you can modify and/or add to this sample. Editable tracks and samples are easily modified in the Timeline. Common editing operations, easy access Timeline tools, and dynamic time display assist you in selecting and modifying tracks and samples with pinpoint precision.

Using the Selection Tool ↖

The Selection Tool allows you to select and perform various operations on both tracks and samples. When a sample is selected, the color of the sample will change indicating that sample has been selected. A red bar will display across the time scale ruler in the Timeline indicating the time duration of the selected sample.

- Click anywhere on a track to select it. Click on individual samples to select them. Only samples with drag handles can be manipulated.
- Shift-click to select a range of tracks or samples. Select the first item, hold down the Shift key, and select the end item. The entire range is selected.
- Command-click to select non-contiguous tracks or samples. The Command key also toggles the selected state.

Track Properties Window

Double-clicking the Track Header opens the Track Properties window. This window is an alternate means of updating properties that are also found in the Track Inspector. In addition, there are a few additional functions that cannot be accessed through the Inspector, including setting Markers for use as a Chapter Track (Chapter 6—The Text Track, page 142), and Alternate Tracks (Chapter 14—Working with Multiple Tracks, page 285).

Sample Properties Window

Double-clicking an individual sample opens a window where you can view the data that is contained within that sample. If the track is editable or augmentable, the Sample Properties window will also allow you to make adjustments to the individual media components. Each track type has a specific Sample Properties window, the description of which can be found in the individual Track Chapters in this User Guide.

While each track type's Sample Properties window is unique, there are some common functions and features found in most Sample Properties windows to assist your workflow.

SAMPLE PROPERTIES WINDOW ICONS

These common icons allow you to perform basic functions and access tools right from most tracks' Sample Properties windows.

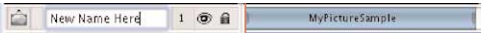
ICON	FUNCTION
←	Edits the previous sample in the track.
→	Edits the next sample in the track.
+	Adds a sample to the end of the last sample in the track, and changes the Sample Properties window to reflect the new sample.
+	Duplicates the current sample, adding it to the end of the track, and changes the Sample Properties window to reflect the new sample.

ICON	FUNCTION
	Deletes the current sample.
	Opens the Debugging Console (See Chapter 16—Introduction to QScript, page 329),
	Previews the movie.

Renaming Tracks and Samples

There are a few ways you can rename your tracks and samples. To change the name of a track, you can enter the name in the Inspector (page 106), the Track Properties window (Chapter 4—Introduction to Tracks, page 121), or use the in-place editing feature directly in the Timeline. To change a sample name, use the Sample Properties window or use the new in-place editing feature.

STEP-BY-STEP: IN-PLACE EDITOR



1. Using the Selection Tool, click to select a track.

2. Click once on the track or sample name. The in-place editor displays.

3. Type in the new track or sample name

Creating Samples

You can create samples via menu commands, by duplicating existing samples, or with the click of your Mouse.

SAMPLE CREATION TOOL

The Sample Creation Tool button allows you to create samples by clicking directly in an editable track.

- Clicking in an empty section within a track will create a sample that extends from the point where you clicked to the next sample in the track.

- Samples have a minimum duration of 1/6 of a second (00:00.100); if the gap in which you click is shorter than that duration, no sample will be created.

- If there are no samples following the insertion point, then a sample of one second in duration will be created.

- Customize the sample's duration as you create it by clicking and dragging the sample along the Timeline. The duration will be determined by where you stop dragging or when you come into contact with the next sample in the track. Provided there is sufficient room in the dragged sample, the start, end, and duration times will be displayed for you as you drag.

DUPLICATE

Use the Edit>Duplicate command (Command-D) to duplicate a selected track or sample. Duplicating will create a copy of the currently selected item and will place it in the Timeline. A duplicated track will appear at the end of the Track List.

- A duplicated sample will appear in the first available location in the same track.
- Duplicated tracks and samples maintain the same properties as the original.

CUT/COPY/PASTE

Edit>Copy (Command-C) and Edit>Paste (Command-V) work in much the same way as Duplicating. The difference is that you can paste a copied sample into a different track, provided the destination track can contain the type of sample that is being pasted. Cutting (Command-X) and Pasting also produces the same results, except the original selection is removed.

- Pasted tracks will appear at the end of the Track List.
- Pasted samples will appear in the first available location of the selected destination track.
- Pasted tracks and samples will have the same properties as their originals.

CREATE SAMPLE COMMAND

Movie>Create Sample (Command-K) will add a sample to the selected track at the closest available space to the right of the Playhead. To precisely position your new sample, move the Playhead to the exact time along the time scale and then perform the command. If there is insufficient room for the sample at the current time, the sample will be added to the end of the track.

Deleting Samples

- Press the delete key to delete one or more selected samples.
- Pressing the delete key when a selected track contains no samples will delete the whole track.

SAMPLE DELETION TOOL

Use this tool and click on individual samples to delete them.

CLEAR COMMAND

Selecting Edit>Clear from the Menu Bar will also delete a selected item.

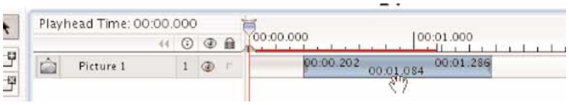
Modifying Samples

You can expand, split, and alter sample duration right in the Timeline with new tools and editing features.

MOVING SAMPLES

Using the Selection Tool, you can easily move samples around to change their start and end points, or to re-sequence a group of samples in a track.

- Select and drag a sample from any point other than its edges to move it. Space permitting, duration, start, and end times will display and adjust as the sample is dragged.



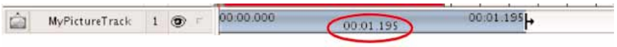
- You can select multiple samples and drag them all at once.
- Dragged samples snap to other samples.
- You can move samples around other samples to available spaces in a track by clicking and dragging. The dragged sample will snap to any sample it comes in contact with.
- You cannot move a sample into a space that is too short in duration to accommodate it.
- Holding down Option while moving a sample will push or pull all the samples following it.
- Use the arrow keys when you have one or more samples to nudge those samples incrementally. Hold down the Shift key in conjunction with the arrow keys to move the sample(s) in larger increments. If you have multiple samples selected and one of those samples no longer has room to move, it will stop moving but the remaining samples can continue to be nudged until each has no more room to move.
- Samples snap around other samples when you drag them around in a track.

CHANGING SAMPLE DURATION

Using the Selection Tool, you can alter the duration of samples within editable tracks. Editable samples have drag handles on their edges.

- Dragging a sample from its left or right edge will alter its duration and its start or end time, depending on which edge you drag.

- Start time, end time, and duration will be displayed when room permits.



- Holding down the Option key while dragging a sample's end point will push or pull all the samples before or after the manipulated sample.

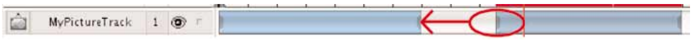
- The Start and End Times of individual samples can also be adjusted in the Sample Properties window.

- Grab handles are visible only when the sample is large enough. Adjust the zoom level of the Timeline to view a sample's grab handles, if they are not visible at first.

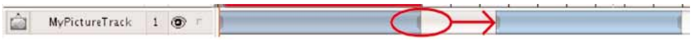
SAMPLE EXPANSION TOOL

The Sample Expansion tool allows you to expand a sample's duration to fill all available space to the left, right or in both directions.

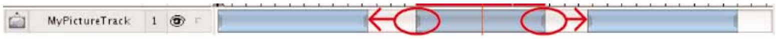
- Clicking on the left edge expands the sample to fill all available space to the sample's left, stopping at either the nearest sample or the beginning of the track if no additional sample is present.



- Clicking on the right edge expands the sample to fill all available space between the sample's right edge and the start of the next sample. The last sample in a track will not expand to the right.

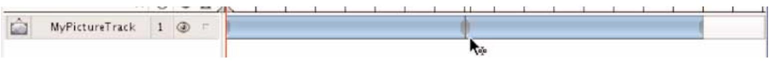


- Clicking in any part of the sample other than the edges will expand both ends to fill empty space to the left and right within the constraints mentioned above.



SAMPLE SPLIT TOOL

Use the Sample Split Tool to split a sample at the exact point where you clicked. The resulting samples will be exact copies of each other.

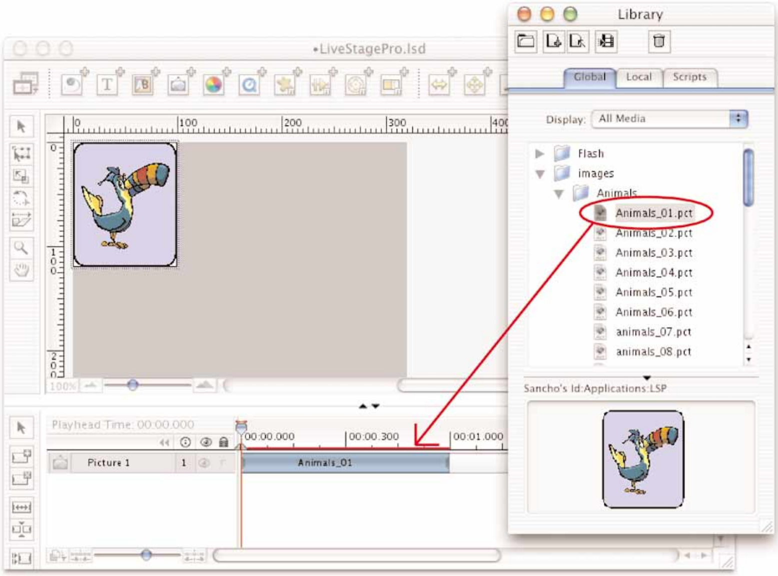


STEP-BY-STEP: WORKING WITH TRACKS AND SAMPLES

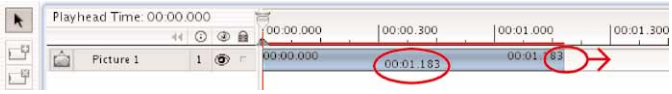
1. If it is not already visible, from the Window menu, select "Show Library Window" (Command-Y).

2. In the Global tab in the Library, open the Images folder. Open the Animals Folder and select one of the images.

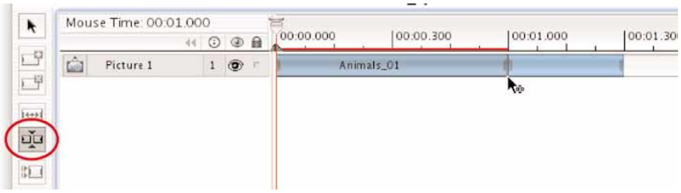
3. Drag the image directly to the Timeline. A Picture Track is created containing a one second sample and the image appears on the Stage.



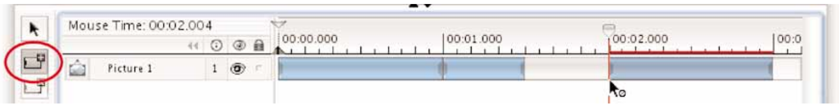
4. Click the Selection Tool and drag the right edge of the sample until the duration is roughly 00:01.300.



5. Click on the Sample Split Tool and move your Mouse over the first sample to 00:01.000. Click in place. The sample is split into two separate samples.



6. Click the Sample Creation Tool and move your mouse to 00:02:000. Click in place. A new empty one second sample is created.



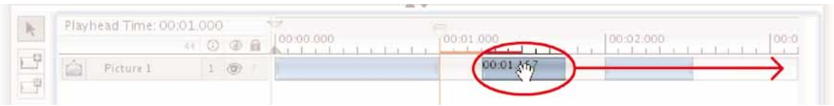
7. Select another image from the Animals folder in the Library's Global tab and drag it to the new empty sample in the Timeline. Three options appear—Replace/Insert/Chop.

8. Select Replace to replace the empty contents of the sample with the new image.

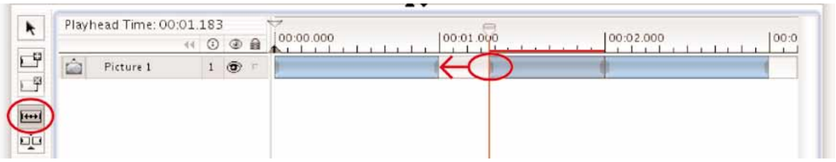
tip

Holding down the Option key while dragging media to the Stage forces the media to create a new track as opposed to being incorporated into an existing track.

9. Click the Selection Tool and clicking in the middle of the second sample, drag it to the end of the track.



10. Click on the Sample Expansion Tool and click the left edge of the middle Picture sample. The sample expands to the start of the second sample.

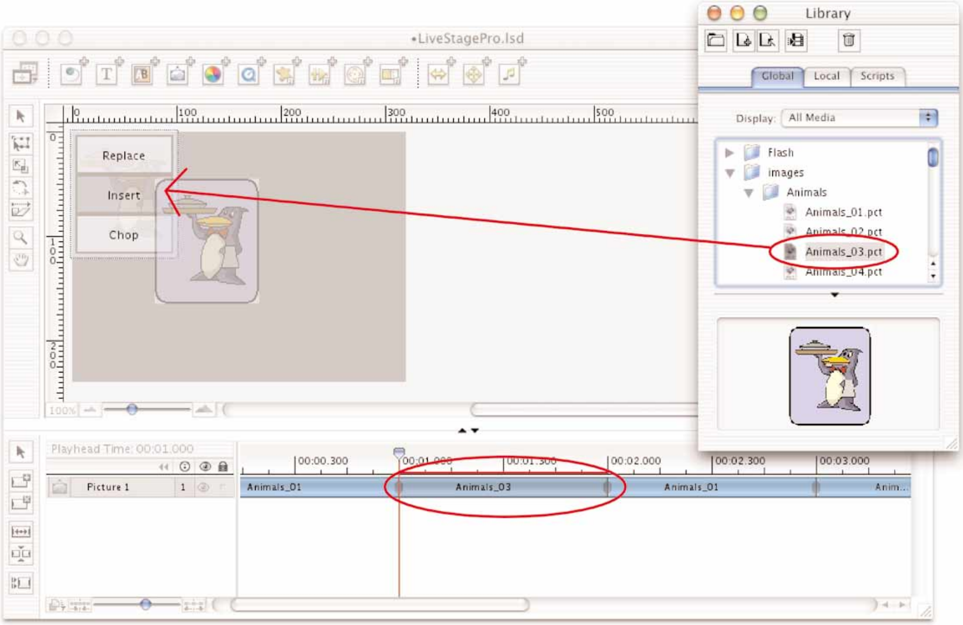


11. With the last sample in the track selected, choose another image from the Animals folder and drag it to the Picture Track on the Stage.

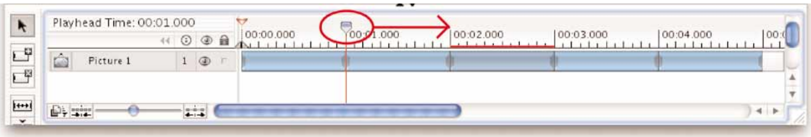
The Replace/Insert/Chop options appear.

12. Drop the image on Insert.

The new image appears on the Stage and a new sample is inserted directly before the last sample in the Timeline.



13. Move the Playhead in the Timeline to 00:02.000.

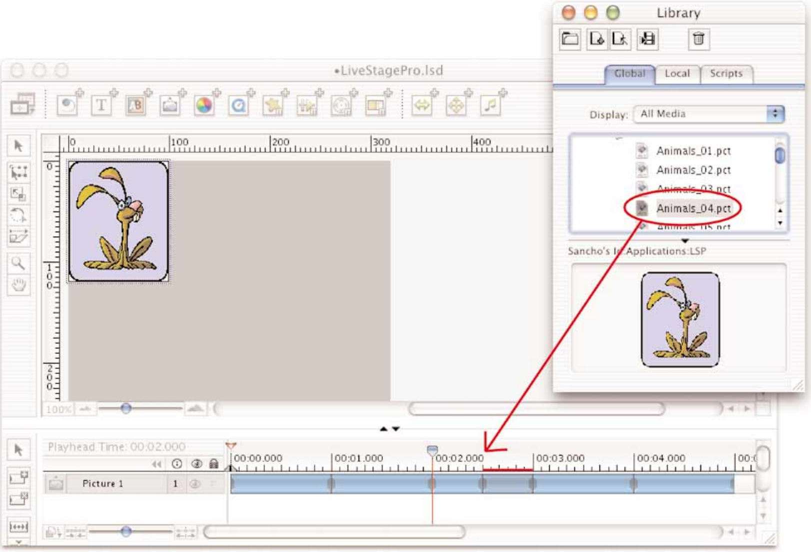


14. Drag one more image from the Animals folder in the Global Library to the sample on either the Stage or the Timeline.

    Again, the Replace/Insert/Chop options appear.

15. This time, select Chop.

    The sample in the Timeline splits into two separate samples. The first sample is the original image. The second contains the new image.



Track Duration and Offset

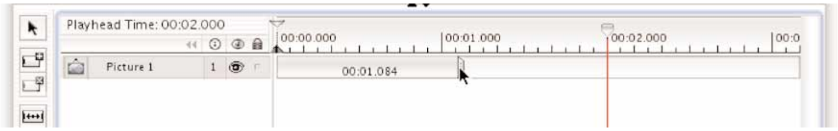
You can alter the duration of tracks as well as samples. The duration of tracks and samples is, for the most part, inter-dependent. A track's duration is determined by the end point of the last sample contained within the track. Moving this endpoint will automatically change the track duration. Likewise, altering a track's duration will affect the duration of the samples it contains.

It is also important to consider the end effects of modifying the duration of certain track types. Altering the duration of a Flash Track or a Video Track changes the sample's frame rate, and can cause it to play back in slow motion or at high speed. Streaming, Modifier, and VR Tracks containing Object Movies cannot have their duration modified.

You can also modify the offset of a track. Offsetting a track delays its loading. If your project contains a track that does not need to be played right at the beginning of the Timeline, offset it to the time when it will be needed. This intelligent project construction can ease the loading demands on QuickTime at the beginning of your movie.

TRACK MANIPULATION TOOL

Clicking on the Track Manipulation Tool switches the Timeline to Track Manipulation mode, showing just the tracks, and allowing you to adjust both their duration and offset.



- Click in a track and drag to change the offset time. The new offset appears at the left edge of the track as you drag.

- Click the resize handle that appears at the right edge of tracks that permit resizing and drag to adjust the track's duration. The duration is displayed within the track as you drag.

- Changing the track duration resizes the samples within the track proportionately.

Folder Drop [NEW in LiveStage Professional 4.0]

New folder functionality allows you to arrange your project media within folders outside of LiveStage Professional and then create the appropriate tracks, samples, and sprites instantly by dragging them to the Timeline or Stage. To build these folders, you will need to follow pre-defined naming and organization conventions:

Name your folder as follows:

```
TheName (TheType)
```

"TheName" can be any name you like.

"TheType" is one of the pre-defined object types.

Object Types

Pre-defined object types inform LiveStage Professional of the type of object to create.

(Sprite) or (Spr)

This object type folder should be dropped only onto sprite samples. When this folder is dragged to the Timeline:

- A sprite named "theName" will be added and assigned the images for button states.

- Image files using the naming convention "Up.xxx", "Down.xxx" and "Over.xxx" will be added to the sprite sample and named "theName" + "Up" etc. ("xxx" is a valid QuickTime Image format.)

- "Handler.txt" will have the text (QScript) added to the "Handler" (see the 'Event Handlers' section in Chapter 16—Introduction to QScript)

- "Behavior.bhv" will add the Behavior (see Chapter 17—Behaviors) to the sprite.

(Up), (Down), (Over)

These are optional for images in the sprite folder to force them to be the up, down, and over image.

(Image) or (Img)

These folders can be dropped onto sprite samples only. All image files in the folder will be added to the sprite sample, but no sprite will be created.

(PictureT) or (PT)

These folders can be dropped onto the Stage only. All image files in the folder will be added to a new Picture Track.

(SpriteT) or (ST)

These folders can only be dropped onto the Stage. They should contain (Sprite) folders.

(TextT) or (TT)

These folders can be dropped onto the Stage only. All text files within the folder will be added to a new Text Track.

(Group) or (Grp)

Dropping this folder onto the Stage or Timeline will create a Group Track and will group any tracks created within this group.

tip

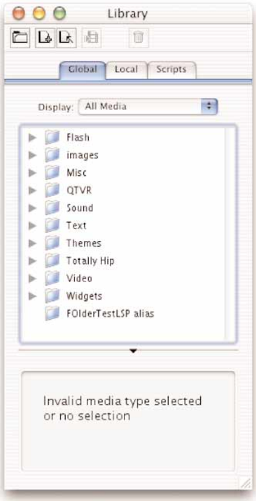
The default duration for samples created by Folder Drop is one second. To modify this duration, hold down the Option key while performing the drag-and-drop. A dialog will display asking you for the new sample duration. The value you enter will be used as the duration for any samples in the dropped folder.

THE LIBRARY

The media files you incorporate into a LiveStage Professional project are not stored within the project document. Instead, the media in the project is referenced from its location on your computer. The Library provides a means to organize and access the media used in your project. The Library can access items on media storage devices such as hard disks, CD-ROMs, Network Servers, and so on. For collaboration, Libraries can also be shared volumes.

To open the Library window, select Window>Show Library Window (Command-Y). The Library consists of three tabs: Global, Local, and Scripts. Each tab corresponds to a different folder that contains a specific category of media. The Global tab contains media that come with LiveStage Professional and are available to any project you create. The Scripts tab is also available to all LiveStage Professional projects and contains Scripts and Behaviors. Finally, the Local Tab contains all the media that you designate for a single project or group of related projects.

Global Tab



Invalid media type selected or no selection

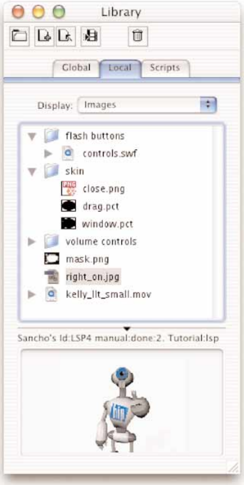
LiveStage Professional includes many different media elements that you can incorporate into your projects. This content is stored in a Library folder in the same directory as the LiveStage Professional application. All content within this folder is viewable and accessible to all LiveStage Professional projects through the Global tab. The media installed with LiveStage Professional is grouped into several folders. These include Images, Sound, Text, and Video, among others.

If you have media elements that you use repeatedly in your projects, it is a good idea to store them in your Global Library. This will enable you to access them in every project you create without having to drag or duplicate them into the Local Library. Simply drag your frequently used media files and folders into the Global Library, which is located in the LiveStage Professional Application folder.

tip

The Media in the Global Library is provided for your personal use in your LiveStage Professional projects. If you wish to incorporate any Global Library media elements for commercial purposes please contact: Totally Hip Software or the individual copyright owners (as listed in the READ ME file on the LiveStage professional CD-ROM) for written permission.

Local Tab



Sancho's Id:LSP4 manual:done:2. Tutorial:lsp

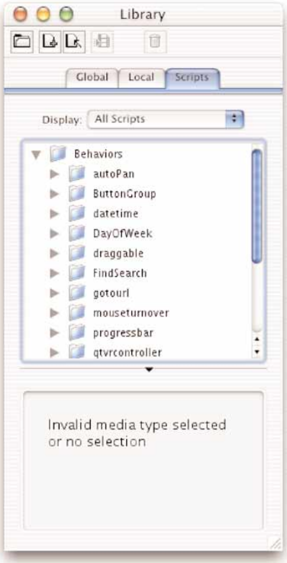
tip

Store related projects in the same directory so they can share the same Local Library folder.

The Local tab of the Library corresponds to the Library folder that LiveStage Professional automatically creates for you when you first save a new project file. Unlike the Scripts and Global tabs, media in the Local tab pertains to the current project only. Any media elements unique to the current project should be stored in the Local Library.

Always keep your project document and its associated Library folder in the same directory to ensure that LiveStage Professional can locate the media files your project requires when you reopen it.

Scripts Tab



Behaviors
autoPan
ButtonGroup
datetime
DayOfWeek
draggable
FindSearch
gotourl
mouseturnover
progressbar
qtvrcontroller

Invalid media type selected
or no selection

The Scripts tab displays a list of custom scripts that you can use in your projects. These commonly-used routines are stored in the Behaviors and QScripts folders. You can also store your own QScripts (Chapter 16—Introduction to QScript, page 311) and Behaviors (Chapter 17—Behaviors, page 335) in this tab for use in multiple projects. Just drag them into the Scripts folder located in the Library folder of the LiveStage Professional Application folder. If you like, you can also create sub-folders within the QScript and Behaviors folders to organize the scripts.

To utilize a script from the Scripts tab, drag the icon representing the script from the Library onto the script edit field of a scriptable track, such as Sprite, Flash, QTVR, or Text Track. In the case of Behaviors, drag them into the Behaviors tab of the Sprite.

Library Toolbar

The Library assists you in managing your project’s media and offers flexibility you would expect from a folder on your computer’s desktop, all without leaving LiveStage Professional.

ICON	FUNCTION
📁	Creates a new folder in the Library window and corresponding Library folder.
🗋	Creates a copy of a selected file and adds it to the Library.
🗋	Creates an shortcut, or alias, of a selected file and adds it to the Library.
🎞	Converts the selection to a QuickTime movie.
🗑	Deletes the selected file. If the item you are deleting is a shortcut or alias, only the shortcut will be deleted. The original file will remain untouched.

tip

Although you can drag media into your project from any location on your computer, it is important that you get in the habit of using the Library. Thus, throughout this User Guide you will be instructed to drag media only "from the Library."

Bringing Media into the Library

There are a few ways to bring media into the Local Library:

- use the Library Toolbar buttons to add an alias or copy to your folder.
- drag media directly into the Library window.
- drag media to the Library folder in your project file's directory.

ADD A COPY/ADD A SHORTCUT

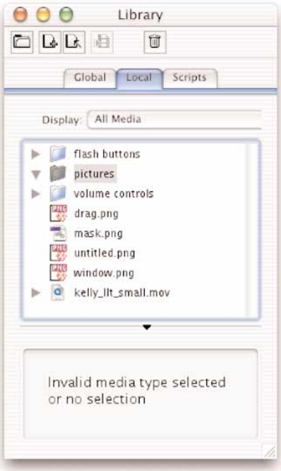
Files located outside of the Library folder can still be accessible through the Library. You can either add an actual copy of the file, or create a shortcut or alias to it. 'Add a Copy' copies the file and puts its duplicate into the Library. 'Add a Shortcut' creates a reference to the file, and then places that reference within the Library folder.

DRAG AND DROP

As is common throughout LiveStage Professional authoring, you can drag and drop media into the Library.

STEP-BY-STEP: ADDING MEDIA TO THE LIBRARY

1. Select the Library folder where you want the new item to be added.

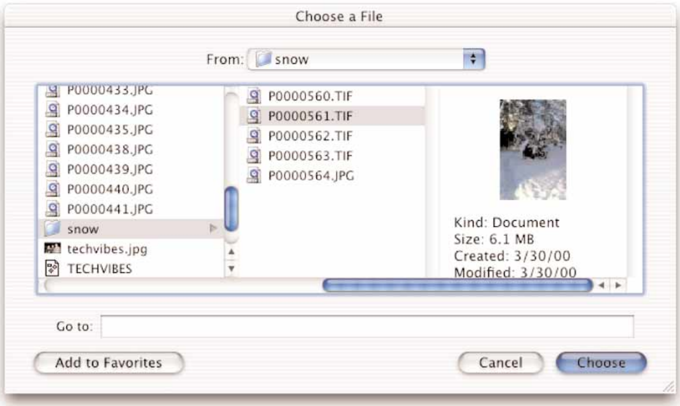


Display: All Media

Invalid media type selected or no selection

2. Click the Add A Copy of a File button. The “Choose a File” dialog will appear.

3. Select the file you want to add to the Library.



From: snow

Kind: Document
Size: 6.1 MB
Created: 3/30/00
Modified: 3/30/00

Go to:

4. Click ‘Open’ or ‘Choose’ to add the file to the Library. The new media asset will appear in the selected folder.

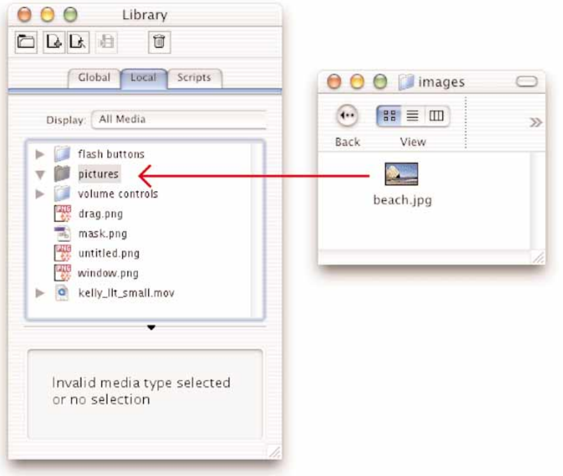
OR

5. Select a media file that LiveStage Professional supports from your computer, such as a .gif, .pict, .mov, .swf, MPEG etc.

tip

You can click on the names of the files in the Library to edit them in place.

6. Drag it directly to the Library window in LiveStage Professional. The item appears in the Library window as well as in the Library folder associated with your project.



Library Aids

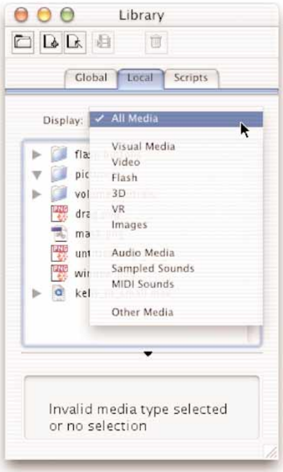
The Library provides you with a great deal of control over the media elements it houses. A variety of options allows you to view, access, and manipulate the items individually.

CONTEXTUAL MENU

Control-clicking or right-clicking any item in the Library brings up a contextual menu that enables you to:

- Open the folder on your computer that houses the individual media element
- View the path to the media element
- Open the element itself in the appropriate media editor (e.g., Photoshop(r), FireWorks(r))
- Move the element to the Trash. This removes the element from both the Library and the folder on your computer that contains it.

FILTERS



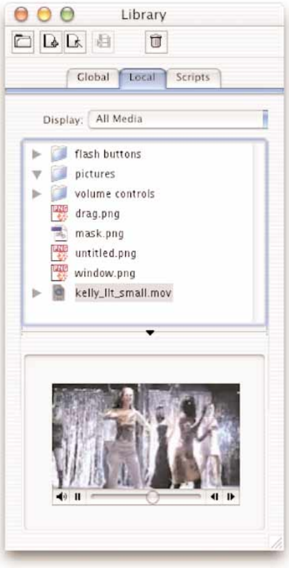
Display: All Media

Invalid media type selected or no selection

When you have a large number of files in your Library, you can use filters to simplify the process of searching for content. Access the filters in the 'Display' popup menu, at the top of each tab in the Library.

- The Global and Local tabs provide a variety of filters to display specific types of content within that particular tab. View only Visual Media or display only QuickTime VR media.
- The Scripts tab displays filters for AppleScripts and QScripts.
- Filtering does not affect the actual contents of your Library, only what is visible in the current tab.
- To view all of your Library's contents, select All Media from the Display popup menu.

MEDIA PREVIEW



Display: All Media

flash buttons
pictures
volume controls
drag.png
mask.png
untitled.png
window.png
kelly_llt_small.mov

The Media Preview pane located at the bottom of the Library window displays individual media elements within your Library.

- Scripts and any media format that is supported by LiveStage Professional can be displayed.
- Image types containing layers, such as Photoshop and GIF files can be opened to access their individual layers. Expand the image in the Library window by clicking on the expansion triangle and then clicking on the specific layer you wish to view.
- The Preview is fully functional, providing playback controls for video, audio, Flash, and QTVR elements.
- Double-clicking an element in the Media Preview pane will launch a full sized preview.

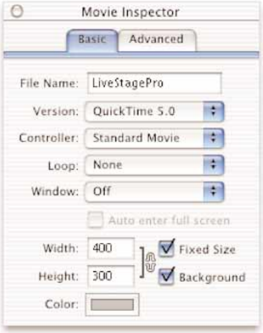
THE INSPECTOR

The Inspector is a new addition to LiveStage Professional. With the Inspector, you can access a track's most common properties without having to open the Track Header, as you would in previous versions. Through this floating window, you can quickly and easily adjust a variety of settings on any track you select. When no track is selected, the Inspector reflects the Movie settings.

Most tracks share common properties that appear in the Inspector. Additional track-specific fields can be found in chapters dedicated to individual tracks. To view the Inspector, select Window>Show Inspector (Command-I).

The Basic Tab

The Basic tab in the Inspector offers a number of fields to adjust the visual aspects of your track.



File Name: LiveStagePro
Version: QuickTime 5.0
Controller: Standard Movie
Loop: None
Window: Off
□ Auto enter full screen
Width: 400
Height: 300
☑ Fixed Size
☑ Background
Color:

NAME

This is the name of the selected track. This name is displayed in the track list and can be used to refer to the track in QScript.

LEFT, TOP

These are the X and Y coordinates on the Stage where the top left corner of the selected visual track is located. You can adjust the physical position of the track on the Stage by entering new coordinates in this field.

- If your Stage is in Fixed Size mode, you can only enter coordinates that will keep the track within the Stage dimensions.
- You cannot enter negative values in the Top and Left fields as the top left corner of Stage is itself always located at 0,0.

WIDTH, HEIGHT

These fields display the physical dimensions of the track in pixels. Alterations to track size can be made directly on the Stage with the resize tool, or you can enter the dimensions here. If you want to maintain the track's aspect ratio, click the link icon to the right of the dimension fields.

LAYER

This displays the Drawing Layer for the media in question. For more information on working with Drawing Layers, read Chapter 14—Working with Multiple Tracks, on page 281.

OFFSET

This displays and allows you to adjust the offset time for the selected track.

DRAW MODE

You can create various visual effects including Transparency through the Draw Mode drop-down menu. The different Drawing Modes available are described in Appendix 2—Draw Mode Reference, page 495.

tip

In the color picker, hold down the option key to pick up a color from anywhere on your screen.

OPCOLOR

The OpColor—short for Operation Color—is used by some Draw Modes to create a variety of visual results. For Transparency, the OpColor indicates the color that will be transparent. For example, if your track's background is green (R:0, G:100, B:0) and you wish it to be transparent, select 'Transparent' from the Draw Mode popup, click on the OpColor swatch, and select R:0 G:100 B:0 from the RGB Color Picker. Any instances of this exact green in your track will be rendered transparent.

TRACK ENABLED

Deselecting this option disables the track. Tracks are enabled by default. A disabled track is loaded and available in the movie, but is not visible and does not play. Disabling and enabling a track can be very useful when used in conjunction with QScript

tip

Why are there negative values on the volume slider? In scripting, you can quickly toggle sound on or off by multiplying the volume by -1. To do this, use the following line of script: ThisMovie.SetVolumeTo(-ThisMovie.MovieVolume)

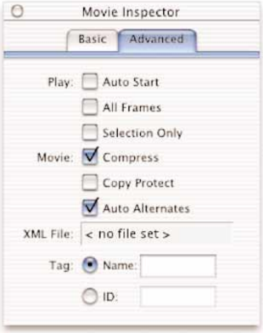
VOLUME

For tracks that include sound or MIDI, the Inspector displays a Volume control. You can use the slider to assign the volume output of the track or you can enter a value between -255 and 255. Values from -255 to 0 are mute, while a value of 255 would be the loudest.

BALANCE

Again for audio tracks, you can shift the audio balance to the Left or Right speakers. This too, can be adjusted via QScript.

The Advanced Tab



Movie Inspector

Play: □ Auto Start
□ All Frames
□ Selection Only
Movie: ☑ Compress
□ Copy Protect
☑ Auto Alternates
XML File: < no file set >
Tag: ● Name:
○ ID:

tip

You can adjust the settings for multiple tracks simultaneously in the Inspector. Shift-click to select consecutive tracks in the Timeline, or Command/CTRL-click to select non-consecutive tracks. The Inspector will change to reflect the common properties that can be altered.

USED IN

Used In Movie indicates in what capacity the track will be used in the movie. The Used In option defaults to 'Movie', meaning that the track will appear in the overall movie you are building. If you will be creating a Poster Frame or a Preview Movie, this is where you specify the tracks that you want included. For instance, if you have a graphic you have designed for the Poster Frame but you do not actually want the track to appear in the movie, you would select 'Poster'.

MATRIX SOURCE

If you create a Tween or a Modifier track (Chapter 13—Additional LiveStage Professional Tracks, pages 259 and 273), and you want it to affect the matrix of the current track (e.g. cause the track to rotate, to move around, or to distort), this is where you would select the appropriate Tween or Modifier track you have created for this purpose.

DRAW SOURCE

If you create a Tween or a Modifier track and you want it to affect the drawing mode of the current track (e.g. fade it to transparent over time), select the relevant Tween or Modifier track here.

XML FILE

Dragging an XML file to this location attaches it to the track for use with QTLists (see Chapter 18—XML and QTLists, page 356). To remove an attached XML file, select it and then press the Delete key.

MATTE FILE

A gray scale 8-bit image dragged into this field acts as a mask for the track. The image can specify various degrees of transparency and uses an Alpha Channel to mask out and display only areas where the Matte image overlaps the original image. Double clicking in the Matte field displays the image at full size. Anything less than 8-bit will be converted to 1-bit and used as a mask.

CACHE

QuickTime will generally remove media samples from RAM soon after playing them. This check box tells QuickTime that it should try to cache the media for this track on the end user's computer instead of automatically purging it. This can improve performance on certain detailed or complex sections of your movie. However, even with this option selected, if QuickTime needs the memory of the cached track to play another part of a movie, it will purge the cache. The track will only remain cached for as long as QuickTime does not require the memory that the cached track has taken up.

HIGH QUALITY

Selecting this check box indicates that the track should be displayed at the highest possible quality, regardless of real time performance considerations. This CPU-intensive operation does not work with all codecs, and may cause your movie to play slower on lower performance computers.

PRELOAD

Selecting this preloads the track into memory. When this option is checked, your movie will not start playing until the entire track is loaded. You should preload Flash Tracks, Skin Tracks, and tracks that contain your interface for the smoothest visual start to your movie.

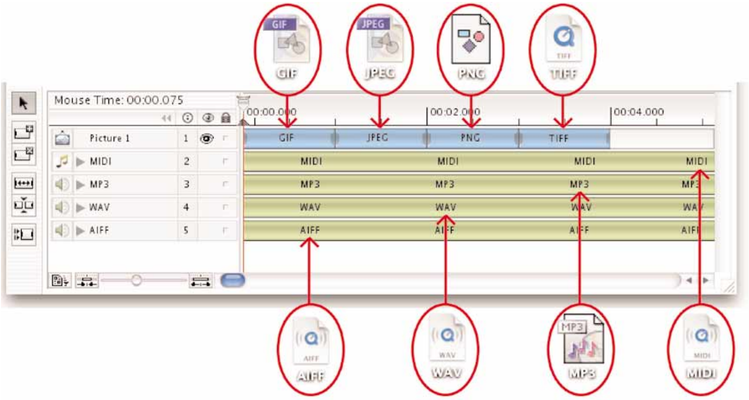
Chapter 4

Introduction to Tracks

WHAT ARE TRACKS?

Tracks are the basic building blocks for QuickTime movies and, by extension, for your LiveStage Professional projects. Most movies we watch today are comprised of only two tracks: a video track with a corresponding audio track. QuickTime movies can have many more than just two tracks. QuickTime movies can have Text Tracks, Flash™ media tracks, Sprite Tracks, and VR Tracks, just to name a few.

Tracks are made up of one or more Media Samples. Media samples contain individual media elements. LiveStage Professional supports a variety of media types and formats that can be used in conjunction with one another to produce a diverse range of interactive content. Each track type is associated with the kind of media it houses. A Picture Track contains one or more still images while an Audio Track is comprised of audio files, and so on. You can have multiple tracks of the same type within a project, but each track can only contain the type of media it is designed to house. For instance, you will not be able to bring an MP3 file into a Flash Track, but a Sound Track will handle it just fine.



WHAT ARE MEDIA SAMPLES?

Media samples are the individual containers of media elements within a track. Some tracks can only contain a single sample, while others allow multiple samples, each with their own individual starting points and duration. The type of data within a sample is determined by the type of track that houses it. For example, a sample in an instrument track will contain a list of instruments while a sample in a picture track will contain a single picture.

TRACK TYPES

The different tracks in LiveStage Professional correspond to the type of media they house. Picture Tracks hold still images, Sound Tracks contain digital audio files, and so on.

FastTracks™ [NEW in LiveStage Professional 4.0!]

FastTracks™ contain simple editors that allow you to create common interactive interface elements without entering a single line of QScript. FastTracks™ are outlined in Chapter 2—FastTracks™, page 43.

MEDIA SKIN FASTTRACK™

The Media Skin FastTrack™ determines the shape of your overall movie much like a cookie cutter determines the shape of a gingerbread man. The Media Skin FastTrack™ allows you to specify the area in which a user can click and drag your finished movie when it is on their desktop. See Chapter 2—FastTracks™, page 51.

PLAYBACK CONTROLS FASTTRACK™

Through drag-and-drop, you can automatically create fully interactive buttons that allow your users to control the playback of audio, video, and other time-based elements in your movie. See Chapter 2—FastTracks™, page 59.

VR CONTROLS FASTTRACK™

The VR Controls FastTrack™ allows you to easily provide buttons that your audience can use to pan, tilt, and zoom your QTVR. See Chapter 2—FastTracks™, page 63.

STATUS BAR FASTTRACK™

The Status Bar FastTrack™ is an easy, script-free way to add a download and progress bar to your movie. See Chapter 2—FastTracks™, page 65.

External Tracks

External tracks are media elements that can be incorporated into LiveStage Professional but cannot be edited directly. External Tracks can only be added by dragging them into the Stage or Timeline. Sound, Video, and Zoomify are examples of External Tracks. See Chapter 8—External Tracks on page 181.

VIDEO TRACKS

Video Tracks are external visual tracks created by dragging a moving image file such as MOVS, AVIs, or MPEG4s directly to the Stage or Timeline. LiveStage Professional is not a video editor, so you will not be able to edit your video inside LiveStage Professional. However, you can add effects such as fade in and fade out. See Chapter 8—External Tracks on page 183.

SOUND TRACKS

Sound Tracks are external tracks that consist of digital audio files such as MP3, AIFF, and WAV files. Sound Tracks are created by dragging audio media to the Timeline or Stage. See Chapter 8—External Tracks on page 183.

Sprite Tracks

The Sprite Track is the most powerful track in LiveStage Professional where interactivity is concerned. Sprite Tracks house individual sprites that can be used to create everything from buttons to complex interactive operations. The Sprite Track can act on itself as well as other tracks, the movie, and even completely separate QuickTime movies. The Sprite Track is detailed in Chapter 7—The Sprite Track on page 155.

Text Tracks

The Text Track is a scriptable track that stores text samples. The Text Track can either be displayed within the movie as just text or be used as a Chapter or HREF (text that references a Web Page) track. Learn more in Chapter 6—The Text Track on page 135.

CHAPTER TRACKS

A Chapter Track is created using a Text Track to provide an index of sections within your movie that is displayed in the QuickTime Player. Your users can click on any of the displayed headings to jump to that section in the movie. For more on Chapter Tracks, see Chapter 6—The Text Track, page 150.

HREF TRACKS

Constructed in a similar manner to a Chapter Track, the HREF track loads URLs into web-based frameset at specified points in time or according to user input. See Chapter 6—The Text Track, page 152.

Effect Tracks

Effect Tracks are used to add transitions like cross fades as well as filters and special QuickTime effects to other tracks. The Effect Track is detailed in Chapter 13—Additional LiveStage Professional Tracks on page 145.

Picture Tracks

Picture Tracks contain still images such as GIFs, JPEGs, PNGs, and TIFFs, amongst others. For more information on working with Picture Tracks, please refer to Chapter 5—The Picture Track on page 125.

Color Tracks

Color Tracks house solid or gradient colors. To learn how to use Color Tracks, refer to Chapter 13—Additional LiveStage Professional Tracks on page 243.

Movie Tracks

Movie Tracks are made up of multiple 'movies' that can be any type of media that QuickTime recognizes. Each movie has its own individual timeline and can be independently loaded and controlled from the main movie's timeline. Movie Tracks are an effective means of delivering multiple video elements. Learn all about Movie Tracks in Chapter 11—The Movie Track on page 213.

Flash Tracks

Flash is another media type that QuickTime understands. Flash Tracks allow you to incorporate Macromedia Flash's vector-based media (.swfs) into your LiveStage Professional project. Through the Flash Track you can access and script Flash buttons to control QuickTime functions. Flash Tracks are detailed in Chapter 9—The Flash Track on page 189.

VR Tracks

The VR Track houses QuickTime VR media. QTVR media consists of 360º panoramas, Cubic (spherical) VR's, and 3D object movies. You can also use QScript to add interactivity to your VR Tracks. QTVR is covered in Chapter 10—The VR Track, on page 201.

Streaming Tracks

The Streaming Tracks link to on-demand and/or live streaming media. Used in conjunction with other scriptable tracks, streaming movies can become highly engaging experiences. For more information, read Chapter 12—The Streaming Track, on page 231.

Instrument Tracks

The Instrument Track allows you to specify MIDI instruments or digital audio files that can then be used in scripts to create audio effects in your movies. This is a great way to add sound to your movie without adding much to its file size. For more information please refer to Chapter13—Additional LiveStage Professional Tracks, on page 255.

Tween Tracks

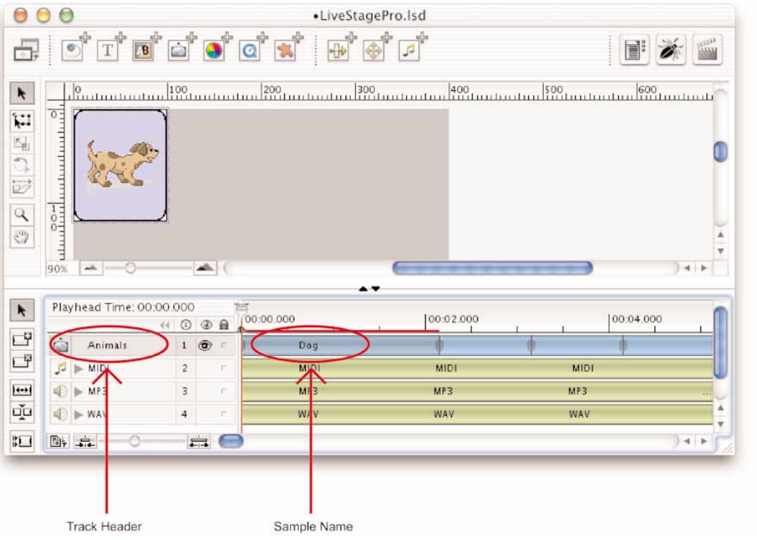
Tween Tracks allow you to enter values to affect properties such as rotation, skew, scale, volume, and drawing layer smoothly over time. Tween Tracks are applied to tracks or individual sprites. This advanced track is covered in Chapter 13—Additional LiveStage Professional Tracks, on page 259.

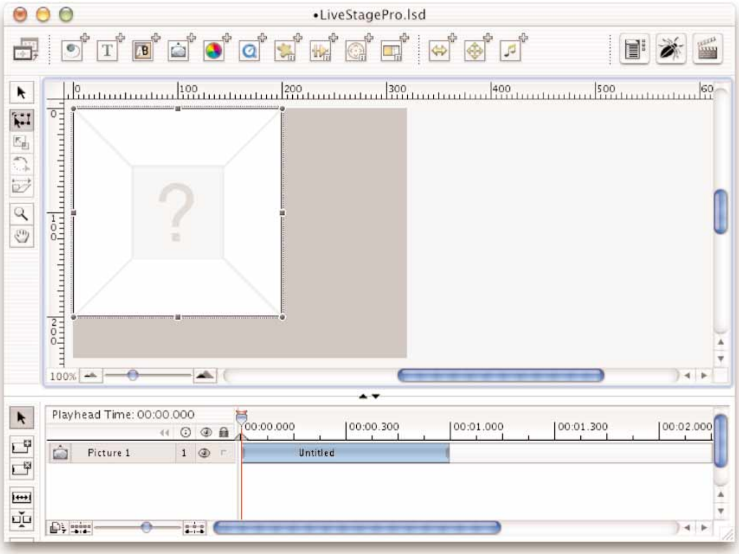
Modifier Tracks

Modifier Tracks are like Tween Tracks in that they effect change over time and are applied to other tracks in the QuickTime movie. Modifier Tracks are best used for repetitive events like simple looping animations, or to trigger events synchronized to your movie's playback. Learn more in Chapter 13—Additional LiveStage Professional Tracks, on page 273.

CREATING TRACKS

When you first start a project, the Stage and Timeline are empty. When you create a track, the Track Header appears in the timeline showing the track type's icon and the track name, along with a media sample. Visual tracks will show up in the Stage as either a visual representation of the media contained in the sample or, if no media is present yet, an empty square with a question mark.



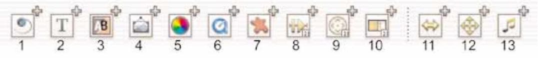


There are several different ways to create a track in LiveStage Professional 4.0:

Track Creation Toolbar

[NEW IN LIVESTAGE PROFESSIONAL 4.0!]

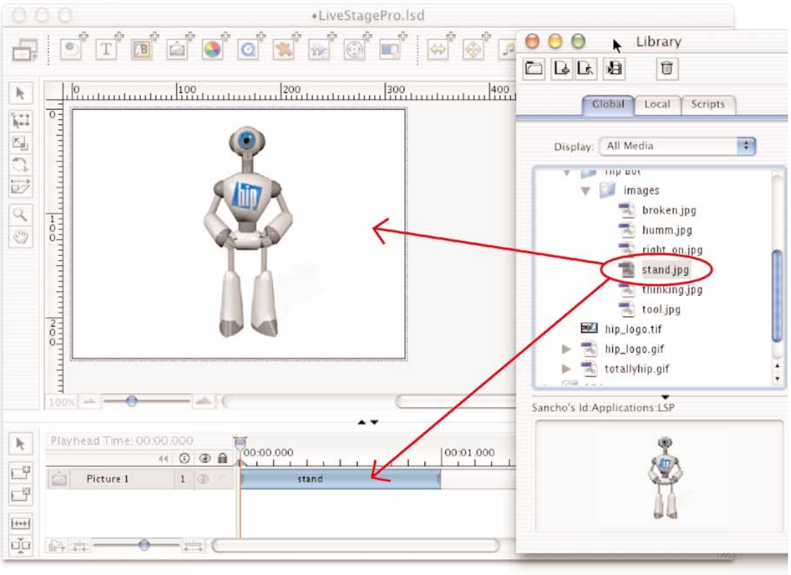
You can create most tracks in LiveStage Professional with a simple click on one of the buttons in the toolbar located at the top of the Document Window. When a new visual track (for example, a Picture Track) is created, an empty track measuring 200x200 will appear on the Stage. Any media that is dragged into the track will be resized to fit the track's dimensions.



1. Sprite Track
2. Text Track
3. Effect Track
4. Picture Track
5. Color Track
6. Movie Track
7. Skin Track
8. Linear Transport Track
9. VR Transport Track
10. Progress Bar Track
11. Tween Track
12. Modifier Track
13. Instrument Track

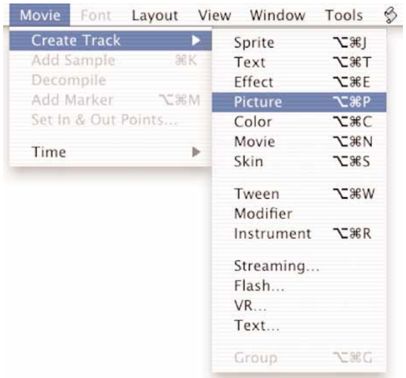
Drag and Drop

You can simply drag media—from either your desktop or the Library window—directly to the timeline or Stage and the appropriate track containing that media is automatically created for you. The original dimensions of your media will be preserved if possible when you create a track with Drag and Drop.



Menu Bar

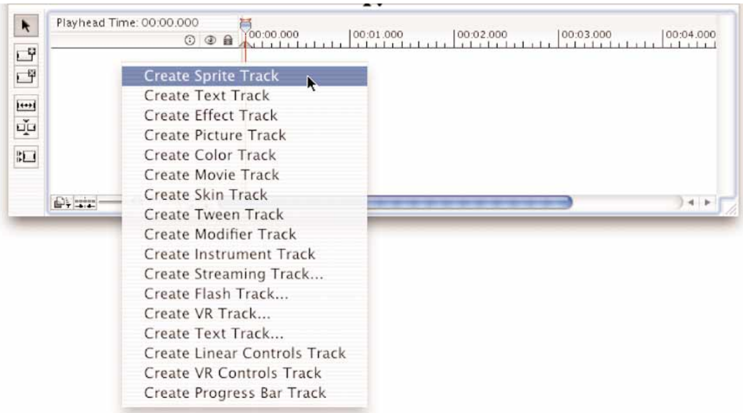
You can also create tracks through the Menu Bar. Select Movie>Create Track and choose the track you wish to create. As with the Track Creation toolbar, any media you drag into a visual track created through the Menu Bar will be resized to fit the dimensions of the track.



Contextual Menus

[NEW IN LIVESTAGE PROFESSIONAL 4.0!]

Tracks can also be created through a handy contextual menu in the Timeline. Simply hold down the control button while clicking in the Track Header area.



EDITING TRACKS

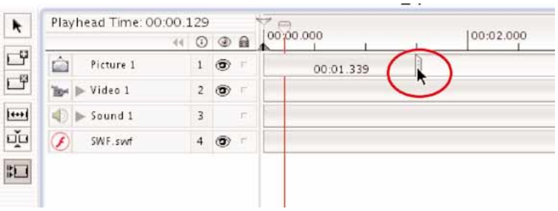
You can select a track and utilize the same familiar Cut (Command-X), Copy (Command-C), Paste (Command-V), and Duplicate (Command-D) commands found in other software to edit it. Both Pasting and Duplicating will add a track to the end of the Track List in the Timeline. To delete a selected track, either hit the Delete key on your keyboard or select Clear from the Edit Menu.

Duration/Offset

To adjust the duration of your track or to offset its start time to a point later in the timeline, click on the Track Manipulation button. The Timeline changes to the Track Manipulation mode.

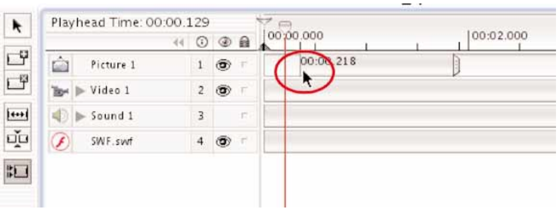
CHANGING A TRACK'S DURATION

Drag the handle on the right edge of the track that you wish to alter. As you drag the handle, you will notice that time information is displayed to aid in precise timing. The samples within the track will expand or contract uniformly according to the change in duration.



CHANGING A TRACK'S OFFSET

Click within the track and drag it to the desired time. This changes the start time of the track, but does not change its duration. Again, time information is provided to help you accurately position the track.



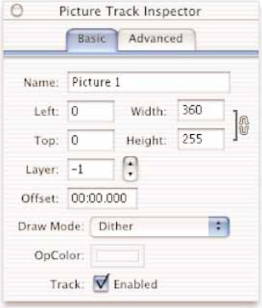
SETTING TRACK PROPERTIES

You can set the properties of your tracks in one of two ways—through the Track Properties window or, more simply, through the Track Inspector.

The Inspector

[NEW IN LIVESTAGE PROFESSIONAL 4.0!]

The Track Inspector is a floating window that displays the properties of whichever track or group of tracks is currently selected. To view the Track Inspector, select Show Inspector from the Window menu on the Menu bar. With the Selection Tool in the Timeline active, click once anywhere on the track you wish to edit and the Inspector will display that particular track's properties. For a breakdown of the different fields shown in the Track Inspector, refer to Chapter 3—Working in the LiveStage Professional Environment on page 106.



Picture Track Inspector

Basic | Advanced

Name: Picture 1

Left: 0 Width: 360

Top: 0 Height: 255

Layer: -1

Offset: 00:00.000

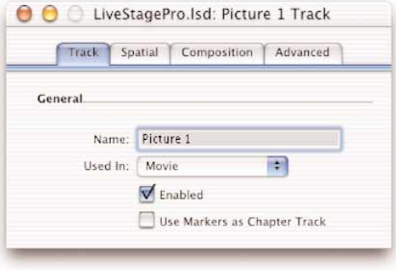
Draw Mode: Dither

OpColor:

Track: ☑ Enabled

Track Properties Window

The Track Properties window provides the same editing options as the Inspector. Access the Track Properties window by double clicking on the Track Header.



LiveStagePro.lsd: Picture 1 Track

Track | Spatial | Composition | Advanced

General

Name: Picture 1

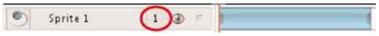
Used In: Movie

☑ Enabled

□ Use Markers as Chapter Track

TRACK ORDERING

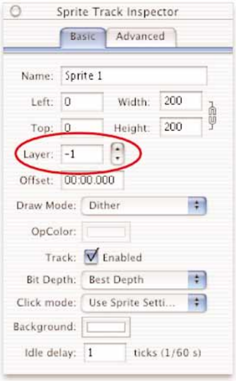
QuickTime is a track-based architecture. LiveStage Professional's timeline reflects this structure. The way tracks are ordered takes two different forms. The first is by Index Number.



Sprite 1 1

This pertains to the order that tracks are loaded into a movie. You can alter a track's index by selecting and dragging the track to the desired location within the track list while in Load Order View. For more information on Views, and how they affect the arrangement of tracks on the Timeline, see Chapter3— The LiveStage Professional Environment page 84.

Tracks are also ordered according to Drawing Layer, which determines how visual elements will appear in relation to one another in your movie.



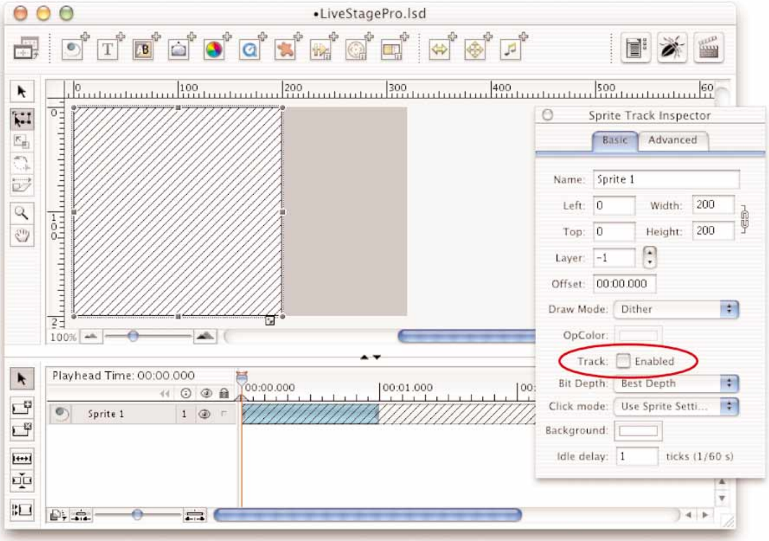
Sprite Track Inspector

Name: Sprite 1
Left: 0 Width: 200
Top: 0 Height: 200
Layer: -1
Offset: 00:00.000
Draw Mode: Dither
OpColor:
Track: ☑ Enabled
Bit Depth: Best Depth
Click mode: Use Sprite Setti...
Background:
Idle delay: 1 ticks (1/60 s)

It is important to consider both points when you are constructing your projects. You can view your tracks by Index Number, by Layer, or by your own custom order by changing the Timeline View. Working with Track Layers and Index Numbers is covered in detail in Chapter 14—Working with Multiple Tracks, page 279.

TRACK ENABLING AND DISABLING

You can disable a track to prevent it from playing in your movie by deselecting the Track: Enabled checkbox in the Track Inspector. You can also use the Contextual Menu.



•LiveStagePro.lsd

Sprite Track Inspector

Name: Sprite 1
Left: 0 Width: 200
Top: 0 Height: 200
Layer: -1
Offset: 00:00.000
Draw Mode: Dither
OpColor:
Track: □ Enabled
Bit Depth: Best Depth
Click mode: Use Sprite Setti...
Background:
Idle delay: 1 ticks (1/60 s)

While the track will still load into the movie, it will not be visible and will not play. This is especially useful when creating a Chapter Track (see Chapter 6—The Text Track, page 150).

You can enable and disable tracks through QScript by using the following syntax:

```
TrackNamed(“myTrack”).SetEnabled(TRUE or FALSE)
```

When a track is disabled, any scripts that are attached to the track will not run until it is enabled.

Chapter 5

The Picture Track

PICTURE TRACK OVERVIEW

Picture Tracks contain still images. They can be used very effectively as slide shows or background images. Any image format that QuickTime understands can be used, making QuickTime and LiveStage Professional the best tool for displaying high quality images over the Internet. Supported images include TIFF's, PICT's, BMP's, PNG's, GIF's, JPEG's, and even individual Photoshop layers. Furthermore, QuickTime can maintain the integrity of the image format. In other words, images in the Picture Track can be set so they will not be recompressed or transcoded; instead they maintain their original media format.

Using a Picture Track as a background image for your project can reduce file size considerably compared to video, since the Picture Track can be a single image file with its duration stretched to fill the movie's timeline.

WHEN TO USE THE PICTURE TRACK

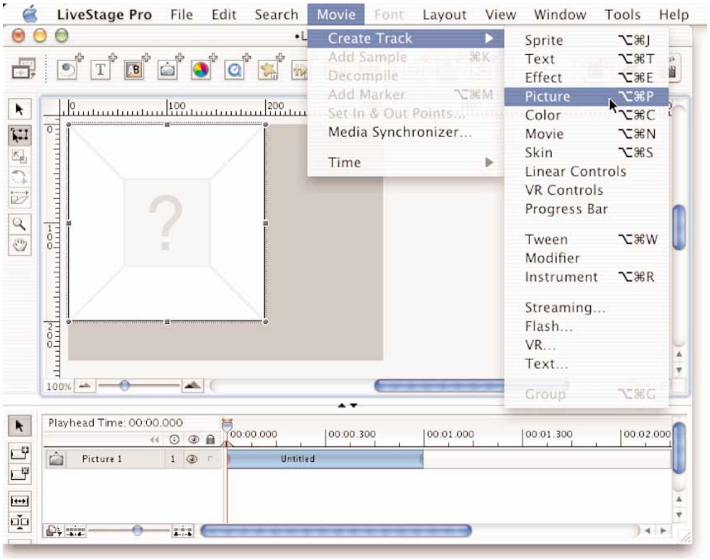
Picture Tracks are a staple of many projects, commonly used as interface backgrounds and slide shows. Use the Picture Track when:

- You want to have one or more background images in your movie.
- You want to create a slide show.
- You want to frame another visual element (see Chapter 1—LiveStage Professional Tutorial, page 22).
- You have PowerPoint slides that you want to put online.

WORKING WITH THE PICTURE TRACK

The Picture Track can contain as many images as you like. Each image is contained in its own sample. You can create a Picture Track in several ways:

- Select Movie>Create Track>Picture (Option-Command-P). The Picture Track appears in the Timeline and on the Stage.



- Click on the in the Track Creation Toolbar. The Picture Track appears in the Timeline and an empty track appears on the Stage.

- Drag an image from the Library directly to the Stage or the Timeline. The Picture Track appears in the Timeline and the image shows up on the Stage.

- Control-click in the Track List area of the Timeline to bring up a contextual menu and select “Create Picture Track” from there.

Adding Picture Samples

Once you have created a Picture Track, you can add image samples. You can do this in a number of ways:

ADDING SAMPLES THROUGH THE SAMPLE PROPERTIES WINDOW

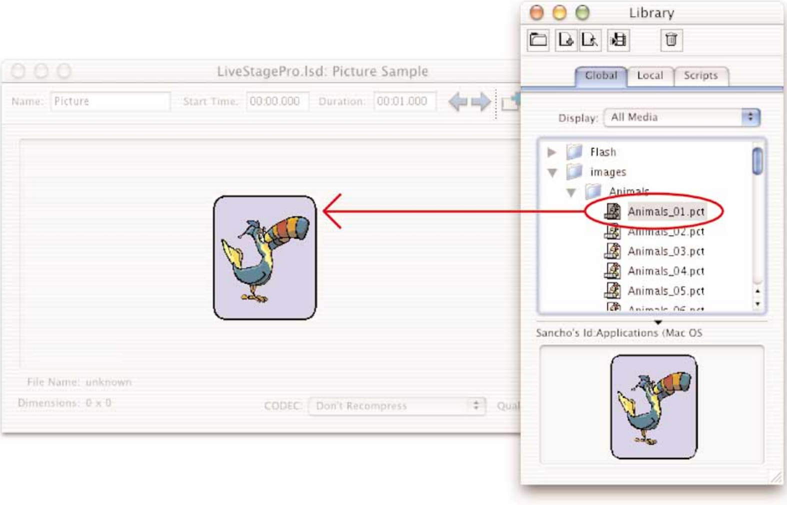
tip

If you want to incorporate a layered Photoshop 6 or higher file into LiveStage Professional, you will need to select the “always maximize compatibility for PSD files” in Photoshop’s preferences.

1. Create a Picture Track using one of the methods previously described.
2. Select the Create Sample tool located near the Timeline
3. In the Timeline, click in an empty area of the Picture Track where you would like the sample to be placed.

   You can drag the sample to position it on the Timeline once it has been created.
4. Double click the sample on either the Stage or the Timeline.

   The Sample Properties window opens.
5. Drag your image from the Library to the Sample Properties window.



LiveStagePro.lsd: Picture Sample

Name: Picture Start Time: 00:00.000 Duration: 00:01.000

File Name: unknown

Dimensions: 0 x 0

CODEC: Don't Recompress

Library

Display: All Media

Sancho's Id:Applications (Mac OS

ADDING SAMPLES THROUGH DRAG AND DROP

Hold down Option while you drag your image to the Stage or Timeline to specify a duration for your new sample other than the default one second.

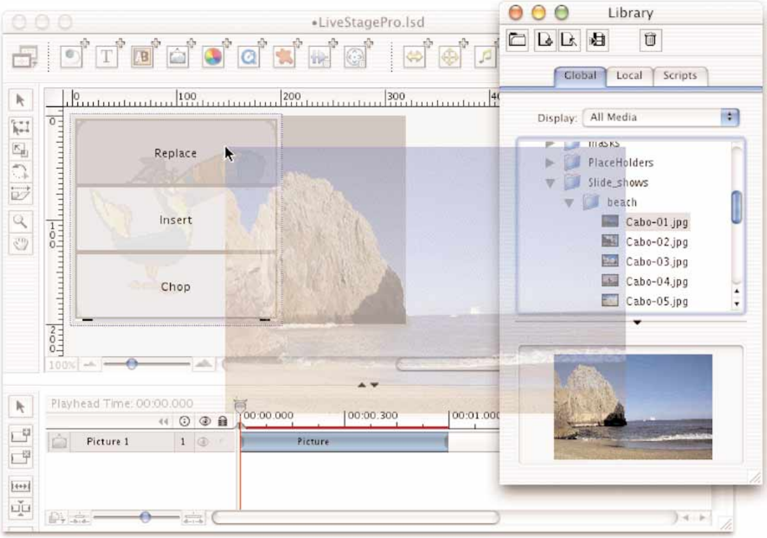
1. Create a Picture Track.

2. Drag an image from your Library to the empty area following the Picture Track in the Timeline.

   The sample is added to the end of the track.

Drag another image from your Library over an existing Picture Track sample on the Stage or in the Timeline.

Three options appear:



Replace

This replaces the contents of the current Picture Track sample with the dragged image. The current sample is determined by the position of the Playhead and is reflected on the Stage.

Insert

This inserts the dragged image immediately before the start of the current sample. Samples will be pushed to the right in the Timeline to make space.

Chop

This splits the current sample at the current position of the Playhead (or right at the half of the sample if the Playhead is not positioned inside it). The first segment of this split sample is replaced by the dragged image and the second retains the current sample’s original image.

ADDING A SAMPLE THROUGH THE CONTEXTUAL MENU

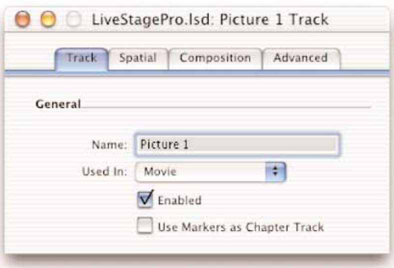
In addition to these options, you can use the contextual menu to add samples. Control-click on the track in the Timeline and the contextual menu will open. Select “Add Sample” to add a new sample to the end of the track in the Timeline.

Drag a folder of images to the Stage or Timeline to create a different track for each image. Shift-select multiple images and drag them to the Stage or Timeline to create a single track made up of samples containing all the images. The track's dimensions will be those of the largest image dragged in.

Picture Track Properties Window/Inspector

You can access the Picture Track Properties window by double clicking on the Track Header or you can access the Picture Track Inspector by selecting Window>Show Inspector (Command-I). Through either of these windows, you can modify various property settings for the Picture Track.

Both the Picture Track Properties window and the Picture Track Inspector contain the same standard set of properties found in most tracks. For a breakdown of the specific options available, see Chapter 3—Working in the LiveStage Professional Environment, page 106.



LiveStagePro.lsd: Picture 1 Track

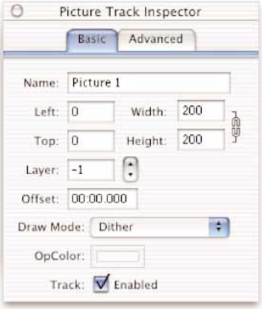
General

Name: Picture 1

Used In: Movie

☑ Enabled

☐ Use Markers as Chapter Track



Picture Track Inspector

Name: Picture 1

Left: 0

Width: 200

Top: 0

Height: 200

Layer: -1

Offset: 00:00.000

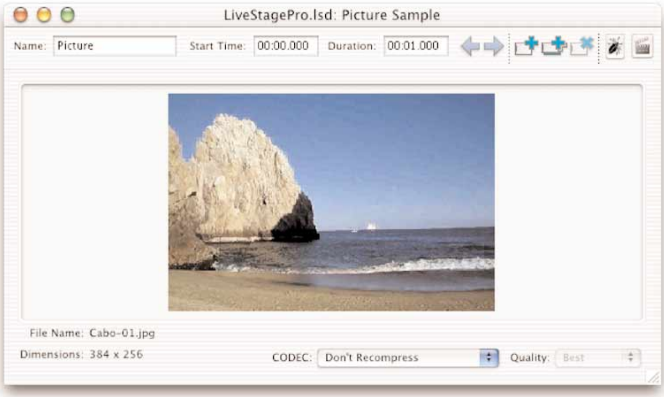
Draw Mode: Dither

OpColor:

Track: ☑ Enabled

Picture Sample Properties Window

Double-click on a sample to open its Sample Properties window. In this window you can view and adjust the particular sample's properties. The Picture Sample Properties window offers tools that allow you to Create, Duplicate, Delete, and navigate through samples in the Picture Track (see Chapter 3—Working in the LiveStage Professional Environment, page 89), as well as the following Picture Track-specific features.



LiveStagePro.lsd: Picture Sample

Name: Picture

Start Time: 00:00.000

Duration: 00:01.000

File Name: Cabo-01.jpg

Dimensions: 384 x 256

CODEC: Don't Recompress

Quality: Best

NAME

This is the sample's name. When you drag in an image to a sample, this name will reflect the image name. When you create a new empty sample, the name will default to "Untitled." You can edit the sample's name in this field or you can click on the sample name in the Timeline to edit it in place.

START TIME

This is the starting time of the sample. If you drag the sample to a new position in the Timeline, this field will reflect the change. You can enter a specific time in this field to move the sample to that starting point, provided no other sample is located at that time.

DURATION

This field indicates the sample's duration. This value will change if you drag the edge of the sample in the Timeline. You may also enter a time value in this field. This value can be changed only if the new duration does not overlap an existing sample in the track.

IMAGE AREA

This area shows a preview of the image that is contained in the picture sample. If the sample is empty, no image appears. You can add an image to the sample by dragging it from the Library or your computer's Finder/Explorer to the Image Area. You can also double-click the image to view it at full size in its own window.

FILE NAME/DIMENSIONS

This displays the file name and actual dimensions of the image that is contained within the sample.

CODEC

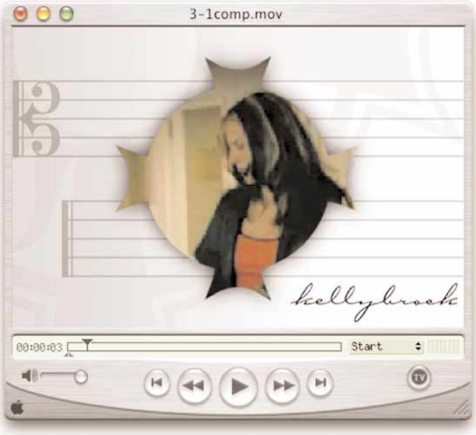
The CODEC popup menu offers different CODECs (Compressor/Decompressors) that you can use on the image in your sample. This option can be convenient when the image in your sample has not yet been compressed and you want to reduce overall file size. If your image has already been compressed, select 'Don't Recompress', as compressing a previously compressed image can result in poor image quality and may even increase file size.

QUALITY [NEW! IN LIVESTAGE PROFESSIONAL 4.0]

This popup gives you a higher degree of control over the quality of an image when you apply a CODEC in the CODEC menu. You can select from Best, High, Medium, and Low to balance quality with file size.

ADVANCED USE OF THE PICTURE TRACK

- Picture Tracks do not have to just be in the background. One of the more interesting implementations of the Picture Track is to use it to frame or partially cover a video as described in detail in the tutorial in Chapter 1—LiveStage Professional Tutorial, on page 22. This is accomplished by activating the Alpha Channel Drawing Mode on a 32-bit image.



- Use the Picture Track in conjunction with the Effect Track to create transitions and other effects.
- Create a Chapter Track for a slide show that allows your user to jump to particular images or sections on demand. Use the "Use Markers as Chapter Track" option in the Track tab of the Picture Track Properties window as detailed in Chapter 12—The Streaming Track, page 238.

tip

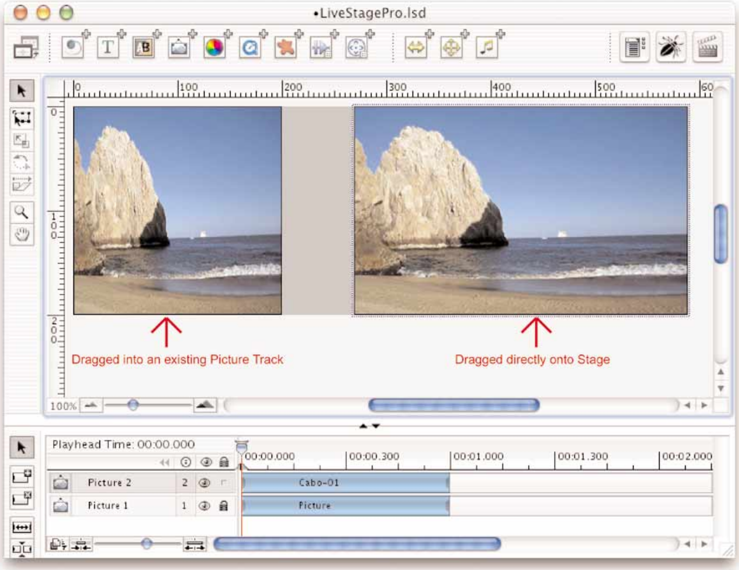
A Picture track can also be used as an image source for a sprite (see Chapter 7—The Sprite Track, page 178). How about a draggable or animated slide show?

TIPS, TRICKS AND TROUBLESHOOTING

Track Size vs. Image Size

It is important to understand how the Picture Track itself affects the images that it contains. The dimensions of a Picture Track determine the dimensions of the images that are added to it. In other words, images brought into a Picture Track will be scaled and their aspect ratio resized as needed to fit the track itself.

The method you choose to incorporate an image into your project will determine how its dimensions and aspect ratio are maintained. When you create a new Picture Track using the Track Creation Toolbar or through the menu bar, the dimensions of the track default to 200x200. An image dragged into this track will resize to fit 200x200. Conversely, an image dragged to the Stage or Timeline will create a Picture Track that matches the image's dimensions.

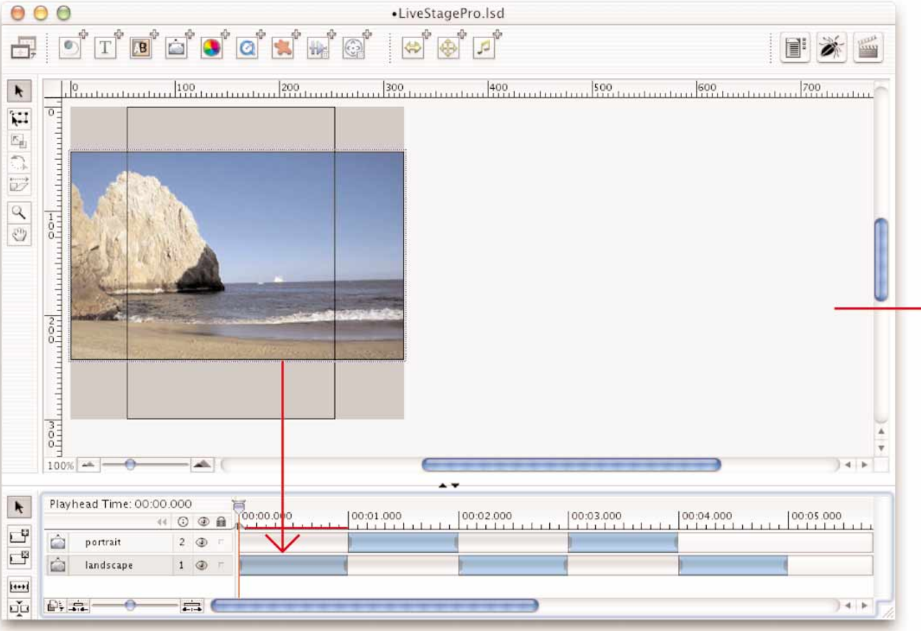


Dragged into an existing Picture Track

Dragged directly onto Stage

There are a couple other ways you can avoid undesirable resizing:

- Place images with different sizes and/or orientation on separate Picture Tracks. This is the best solution for creating slide shows that contain images in both landscape and portrait orientation. Avoid creating too many tracks as this requires more computer resources.



- Adjust the Width and Height in the Picture Track Inspector from the default 200x200 to match the dimensions of your image. You can determine the dimensions of your image in the Sample Properties window.

PICTURE TRACK WRAP UP

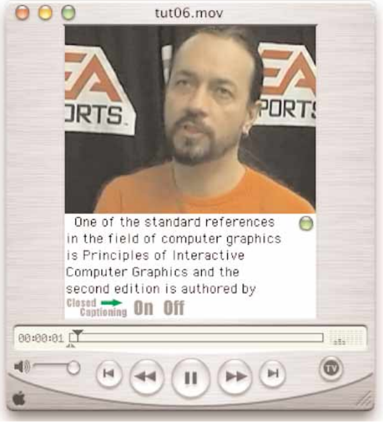
Picture Tracks are virtually a standard in most LiveStage Professional projects, providing backgrounds, interface elements, and slide shows. Make sure to read through Chapter 14—Working with Multiple Tracks to learn how to synchronize a Picture Track with a Video or Sound Track, and visit StageDoor to see the Picture Track in action.

Chapter 6

The Text Track

TEXT TRACK OVERVIEW

At its most basic, a Text Track allows you to present text. Text can be formatted, animated, and synchronized with other elements in your movie. Beyond the basics, Text Tracks are powerful scriptable tracks. They are searchable, editable, and allow user input. Text Tracks can act as ‘Chapter Tracks’, assisting the user in navigating your project, as well as ‘HREF’ tracks that can load URLs into frames at specified points in time.



One of the standard references in the field of computer graphics is Principles of Interactive Computer Graphics and the second edition is authored by

WHEN TO USE THE TEXT TRACK

Text Tracks can be used simply to display text in your movies. Using Text Tracks, you can also add captions to your movies and provide multiple language versions to cover the widest possible audience. In combination with QScript, the Text Track takes on greater functionality. Users can interact with the Text Track, viewing informative content, filling out forms, and even searching for words and phrases in the track.

Examples of when to use the Text Track:

- You want informative text to appear when a user interacts with an object in your movie.
- You want to provide your audience with multiple language options.
- You have a project that needs captions in order to comply with the “Americans with Disabilities Act.”
- You want to create your own karaoke movie!
- You want to keep file size to an absolute minimum and avoid using graphics to represent text.
- You want an easy and file-size efficient means of adding scrolling credits at the end of a movie
- You want to provide user-editable text input fields, such as a form.

- You want the user to be able to search for words or phrases in your movie, and have the movie jump to the point where the text appears.

- You want to load a sequence of URLs into an HTML frameset at timed intervals or to coincide with user interaction.

WORKING WITH THE TEXT TRACK

You can also bring in QuickTime marked-up text and LiveStage Professional will recognize it. Free applications like Magpie from pbs.org can create “QT Text”, from line 21 captioning. Simply drag the marked up text file or text movie to the Stage or Timeline.

The Text Track can contain as many individual samples of text as you like. Individual samples can be set at precise times in your movie to synchronize text with other elements. You can create a Text Track in several different ways:

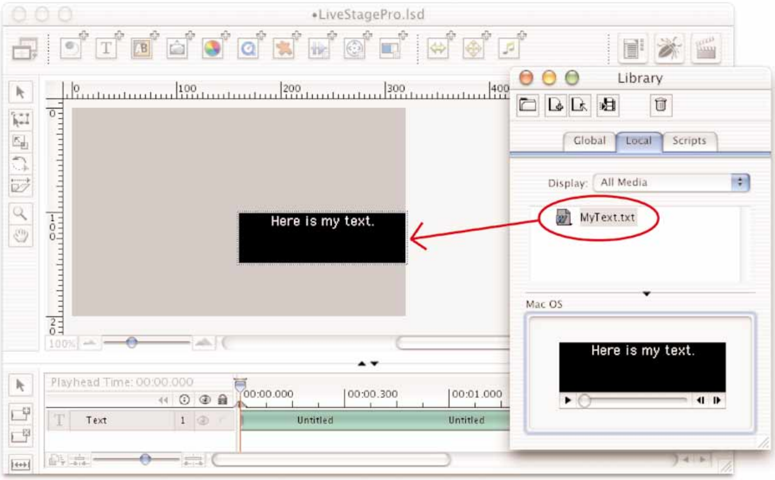
- Select Movie>Create Track>Text... (Option-Command-T) from the Main Menu Bar. The Text Track appears in the Timeline and an empty 200x200 track appears on the Stage.

- Select Movie>Create Track>Text... also on the Main Menu Bar. A dialog window will open allowing you to navigate to a text file on your computer. Select the .txt file of your choice and choose ‘Open’ to add it to your project. A different sample will be created for each paragraph contained in the file.

- Click on the T in the Track Creation Toolbar. Again, the Text Track appears in the Timeline and an empty 200x200 track appears on the Stage.

- Drag a text file (.txt only) from the Library or your computer’s Finder/Explorer directly to the Stage or the Timeline. The Text Track appears in the Timeline and on the Stage. A different sample will be created for each paragraph contained in the file.

- Control-click on the Timeline to open the track contextual menu and select Create Text Track.



Formatting Considerations

Text tracks offer significantly lower file size and increased readability over text represented graphically in a bitmap or other image formats. However, you give up some control over how the text formatting will appear on the end user's computer.

FONT DISPLAY

The fonts you use to format your text are not embedded into the project. In order for the user to view the text as you intended, they will need to have the font installed on their computer. If the font is not present, the user's system will attempt to provide a similar font, or replace it with a common font such as Helvetica or Times New Roman. If the appearance of the text font is critical to your project, then it makes sense to "burn" your text. This creates an image of the text instead, though this will cause you to lose some of the Text Track's functionality. Text burning is described later in this chapter in the Sample Properties Window section under Properties tab.

tip

To avoid cross-platform dpi problems, you can use QScript to detect the user's platform and resize the text accordingly. A complete tutorial covering this procedure can be found on Backstage at http://backstage.totallyhip.com

CROSS-PLATFORM DISPLAY

Text is rendered differently on Macs and Windows machines. Macs render text at 72dpi, while Windows machines render text at 96dpi. Text on Windows machines will therefore appear larger than on Macs.

ADDING SAMPLES

Once you have created a Text Track, you can add text samples. You can do this in several ways:

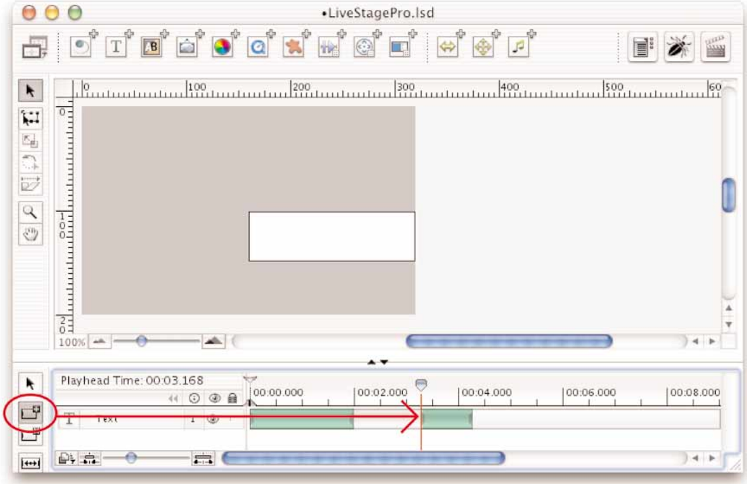
Sample Creation Tool/Sample Properties Window

1. Create a Text Track using one of the methods previously described.

2. Select the Create Sample tool located near the Timeline

3. In the Timeline, click in an empty area of the Text Track where you would like the sample to be placed.

   You can drag the sample to position it on the Timeline once it has been created.



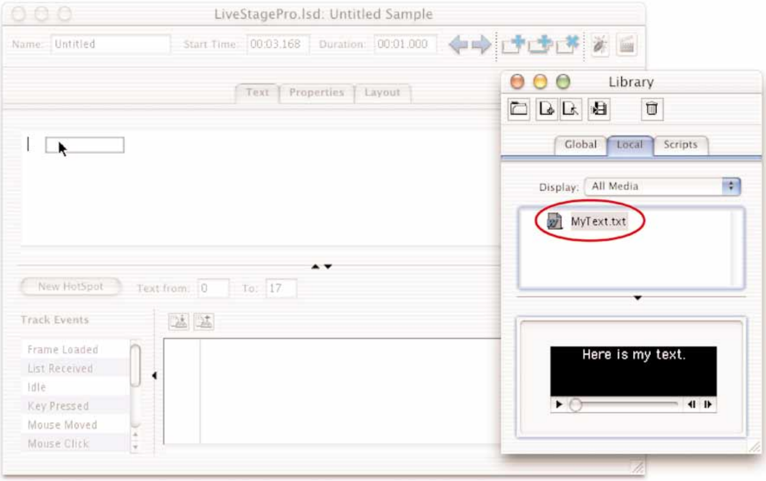
•LiveStagePro.lsd

Playhead Time: 00:03.168

4. Double click the sample on either the Stage or the Timeline.

   The Sample Properties window opens.

5. Enter text into the Text Edit field. You can also drag a text file (.txt) from the Library to the Text Edit field.



LiveStagePro.lsd: Untitled Sample

Name: Untitled
Start Time: 00:03.168
Duration: 00:01.000

Text from: 0 To: 17

Display: All Media

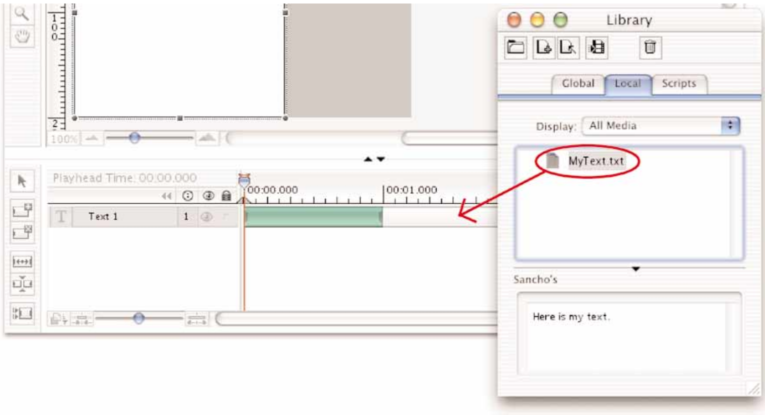
Here is my text.

Drag and Drop

1. Create a Text Track.

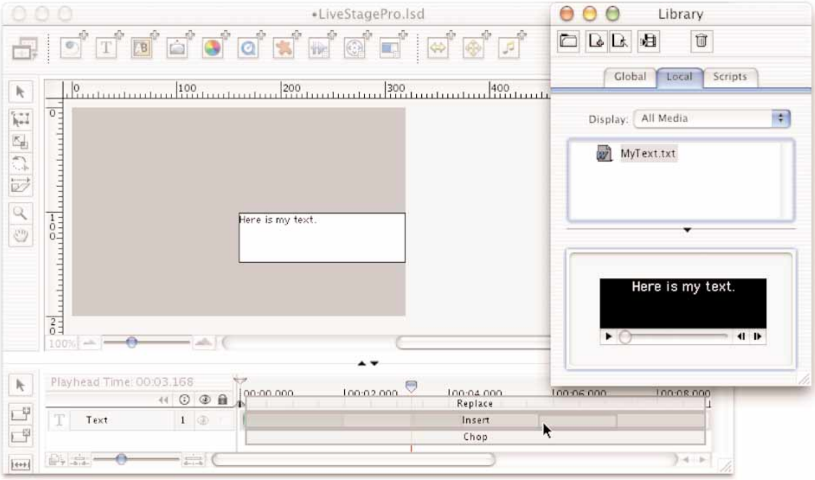
2. Drag a text file from your Library to the empty area following the Text Track in the Timeline.

   The sample(s) are added to the end of the track.



3. Drag another text file from your Library over an existing Text Track sample on the Stage or in the Timeline.

Three options appear:



Replace

This replaces the contents of the current Text Track sample with the dragged text file. The current sample is determined by the position of the Playhead and is reflected on the Stage.

Insert

This inserts the dragged text file immediately before the start of the current sample, pushing existing samples to the right in the Timeline.

Chop

This splits the current sample at the current position of the Playhead or at the sample's halfway point if the Playhead is not positioned within it. The first segment of this split sample is replaced by the dragged text file and the second retains the current sample's original text.

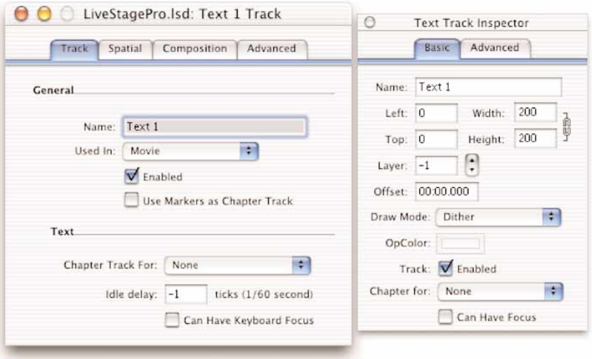
Contextual Menu

You can also create new samples through a Contextual Menu by control clicking in the track.

TEXT TRACK INSPECTOR/PROPERTIES WINDOW

Depending on your preference, you can change the Text Track properties through either the Track Properties window or the Text Track Inspector. Access the Text Track Properties window by double clicking on the Track Header or view the Text Track Inspector by selecting Window>Show Inspector (Command-I) with the track selected in the Timeline.

Both the Text Track Properties window and the Text Track Inspector contain the same standard set of properties found in most tracks. For a breakdown of the specific options available, see Chapter 3—Working in the LiveStage Professional Environment, page 106. In addition to the standard properties, the Text Track offers three additional options:



General

Name: Text 1

Used In: Movie

☑ Enabled

□ Use Markers as Chapter Track

Text

Chapter Track For: None

Idle delay: -1 ticks (1/60 second)

□ Can Have Keyboard Focus

Name: Text 1

Left: 0 Width: 200

Top: 0 Height: 200

Layer: -1

Offset: 00:00.000

Draw Mode: Dither

OpColor:

Track: ☑ Enabled

Chapter for: None

□ Can Have Focus

CHAPTER TRACK FOR

This menu allows you to specify the track for which you would like the Text Track to act as a Chapter Track. For a complete explanation of Chapter Tracks and how to create them, see the Chapter Track section on page 150. The Chapter Track For option is found in the Track tab of the Text Track Properties window, and the Basic tab of the Text Track Inspector.

CAN HAVE (KEYBOARD) FOCUS

Selecting this option allows the Text Track to receive keyboard input from the user. You will want this option selected if you have a text-input field. Can Have (Keyboard) Focus is found in the Track tab of the Text Track Properties window, and the Basic tab of the Text Track Inspector.

You can create a text-input field by making the text sample editable. In order for a track to be made editable, it must contain at least one character, so you will need to enter either a blank space or some text into the text edit field. Then add the following QScript to the text sample's FrameLoaded event:

ThisTrack.SetTextEditState(kDirectEditing)

ThisTrack.SetFocus

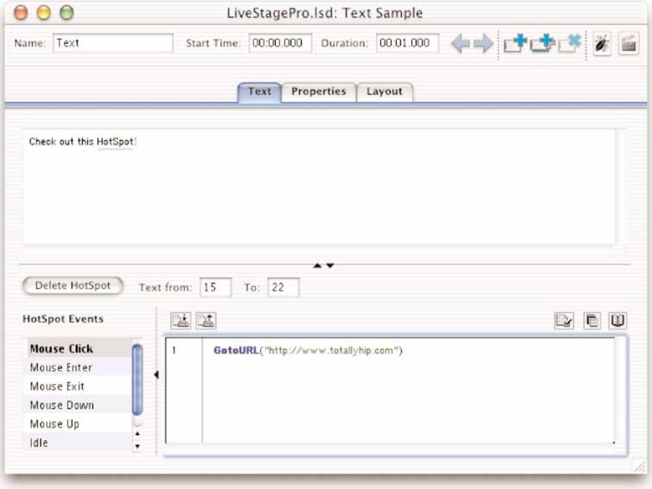
This allows the user to enter text in the text field and ensures the Text Track is active and able to accept user input when the movie starts.

IDLE DELAY

The Text Track, being a scriptable track, can receive Idle Events. The Idle Delay rate set here determines the minimum delay between when the Text Track will receive these events. The default value is 1/60 of a second—or one 'tick'. You can set a value of zero, which is the fastest, or you can set a value of -1 to turn it off. The Idle Delay is found in the Track Properties window in the Track tab and in the Text Track Inspector in the Basic tab. For more information on Idle Delay, see Chapter 6—Introduction to QScript, page 309. The Text Track's default value for Idle Delay is -1.

TEXT SAMPLE PROPERTIES WINDOW

To access the text sample Properties window, double-click on the text sample. In this window you can enter your text, assign appearance and layout properties, and add scripting to your text samples. You will find the same fields and tools that are standard to most Sample Properties windows along the top. For a description of each of these tools, refer to Chapter 3—Working in the LiveStage Professional Environment, page 89. The Sample Properties window is separated into three tabs.



LiveStagePro.lsd: Text Sample

Name: Text

Start Time: 00:00.000

Duration: 00:01.000

Check out this HotSpot!

Text from: 15

To: 22

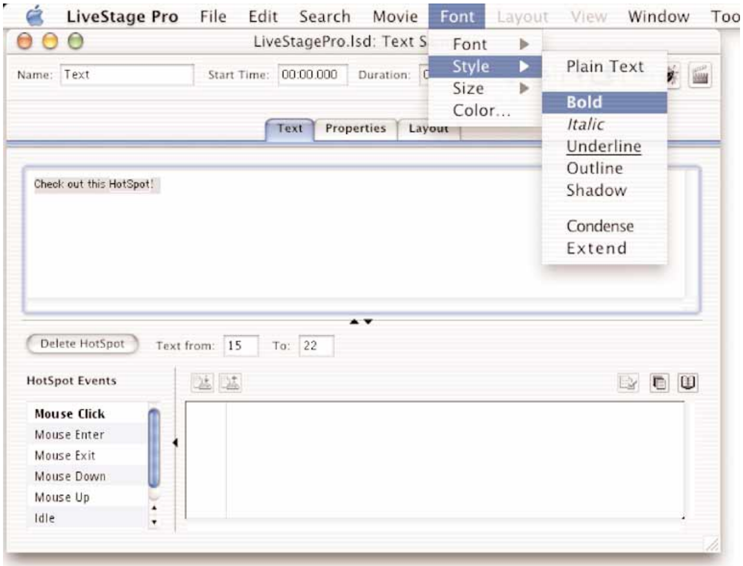
```
GotoURL("http://www.totallyhip.com")
```

THE TEXT TAB

Text content is entered and scripting is assigned in the Text tab. The top portion of the Text tab is the Text Edit Field and the bottom contains an Event Handlers List and a Script Editor, much like those in other scriptable tracks. You can Save and Load scripts, check syntax, and access the Defines and QScript Reference windows through the icons along the top of the Script Edit Field. For a breakdown of each of these icons, see Chapter 16—Introduction to QScript, page 315. There is also a Separator Bar provided for you to maximize either portion of the Text tab.

Text Edit Field

The Text Edit Field is the primary field in the Text tab. In this field, enter the text that you want displayed in the text sample, format it, and determine selections for HotSpots. You can format your text's Font, Style, Size, and Color through the Font menu on the Menu Bar.



LiveStagePro.lsd: Text S

Name: Text Start Time: 00:00.000 Duration:

Check out this HotSpot!

Text from: 15 To: 22

1. Create a Text Track by clicking on the Text Track icon on the Track Creation Toolbar.

   The Text Sample Properties window opens.

2. Double-click the text sample, either on the Stage or in the Timeline.

3. In the Text Edit Field, enter "This is a Text Sample."

4. Select all of the text.

5. From the Font menu of the Menu Bar, choose a font and a color you like, and set the font size to 14 pt.

   The changes are reflected in the Sample Properties window and in the Stage.

HotSpot Editor

HotSpots are sections of text that will perform actions when the user interacts with them. When you select a range of text in the Text Edit Field, the 'New HotSpot' button becomes active. Clicking on the button designates the selection as a HotSpot. This activates the HotSpot Event Handler List below and allows you to add scripting to the HotSpot. You can also enter numeric values in the Text From and To fields to specify the range of characters and spaces you want selected.

Example: To select the words "This is" in the above exercise, the range would be Text From: 0 To: 7.

You can have multiple HotSpots in a single sample, but you cannot have the same text selected in two HotSpots (i.e., HotSpots cannot overlap). When you select an existing HotSpot, the HotSpot Editor button changes to “Delete HotSpot.” Clicking this button will remove the HotSpot and any QScripts attached to it.

Event List

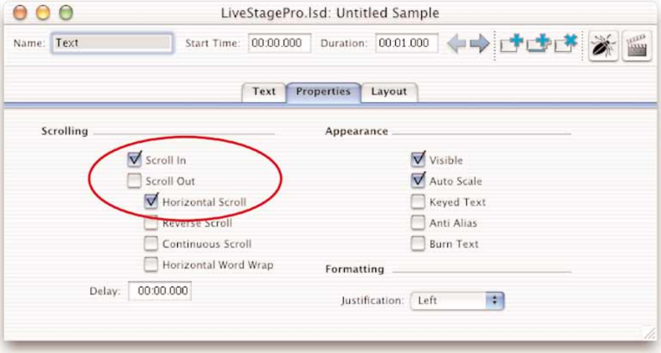
The Event List displays the available Event Handlers. Track Event Handlers are displayed unless a HotSpot is selected; at which point the HotSpot Event Handlers are activated. The Event Handlers operate in much the same way as in other scriptable tracks. Clicking on an item in the Event List selects the Event Handler. You can then enter a script in the Script Edit Field to the right. For more information on Event Handlers, see Chapter 16—Introduction to QScript, page 312.

Script Edit Field

As with all other scriptable tracks, the Script Edit Field is where you enter the QScript statements you want assigned to the selected Event Handler. For more information on QScript and the Script Edit Field, see Chapter 16—Introduction to QScript, page 311. You can use the resize bar to hide the event list and maximize the Script Edit Field.

THE PROPERTIES TAB

The Properties tab contains options that affect the display characteristics of the text in the sample. In this tab you can determine if and how your text will scroll as well as how it will display in the track.



LiveStagePro.lsd: Untitled Sample

Name: Text Start Time: 00:00.000 Duration: 00:01.000

Scrolling
☑ Scroll In
□ Scroll Out
☑ Horizontal Scroll
□ Reverse Scroll
□ Continuous Scroll
□ Horizontal Word Wrap
Delay: 00:00.000

Appearance
☑ Visible
☑ Auto Scale
□ Keyed Text
□ Anti Alias
□ Burn Text

Formatting
Justification: Left

Scrolling

This group of options allows you to scroll your text across the Text Track in various ways. When you enable scrolling, the text will scroll vertically from the bottom of the Text Track to the top by default. You can alter the direction, timing, and other scroll properties by selecting from the options shown.

Scroll In

When this is checked, scrolling is enabled and the text will scroll into view over the length of time that the text sample plays. The text will scroll from the bottom of the Text Track (or Text Box, if it is set) to the top, or from right to left if the Horizontal Scroll option is selected. The duration of the text sample determines the speed at which the text will scroll in.

Scroll Out

When this is checked, scrolling is enabled and the text will scroll up and out of view, or to the left and out of view if horizontal scroll is selected, over the length of time that the text sample plays. The duration of the text sample determines the speed at which the text will scroll out. If both the Scroll In and Scroll Out options are selected, the scrolling time is divided between both scroll actions, with the text first scrolling in and then scrolling out over the duration of the text sample.

Horizontal Scroll

This option changes the scrolling action from vertical to horizontal. Each line of text in the sample will occupy a single line in the horizontal scroll.

Reverse Scroll

The default scrolling is from bottom to top in a vertical scroll and from right to left in a horizontal scroll. Selecting this option will reverse these directions.

Continuous Scroll

This option will cause the next sample in the Text Track to scroll the current sample out. Both samples will be displayed as this occurs, provided both samples have this option checked.

Horizontal Word Wrap

If you have Horizontal Scroll selected and would like to avoid a single long line of text scrolling by, choose this option. This will fit the text into the track bounds or text box (if it is set) and will scroll it with this formatting.

Delay

The delay field allows you to set the timing of your scrolls. The delay time is the duration that the text will be 'held' in the sample without scrolling. Any remaining time in the sample will be used by the scroll function.

Example 1:

```
Sample Duration = 2 seconds
Delay = 1 second
Scroll In = 1 second
```

If you specify a one-second delay in a sample that is two seconds long and have a Scroll In function set, the scroll will take one second to complete.

Example 2:

```
Sample Duration = 2 seconds
Delay = 1 second
Scroll In = 1/2 second
Scroll Out = 1/2 second
```

If you specify a one-second delay in a sample that is two seconds long and have both Scroll In and Scroll Out functions set, each scroll will take 1/2 second to complete.

Appearance

Appearance affects how the text itself, all formatting aside, will be displayed.

Visible

Deselecting this will hide the text in the sample.

Auto Scale

This will cause the text in the sample to scale if the movie or track is resized. This is selected by default.

Keyed Text

Selecting this will render the background for the text transparent.

Anti-Alias

QuickTime will anti-alias the text when this option is selected. While this can make larger type faces look cleaner and more readable, it can result in poor quality when the text is keyed and is placed over a Video Track.

Burn Text

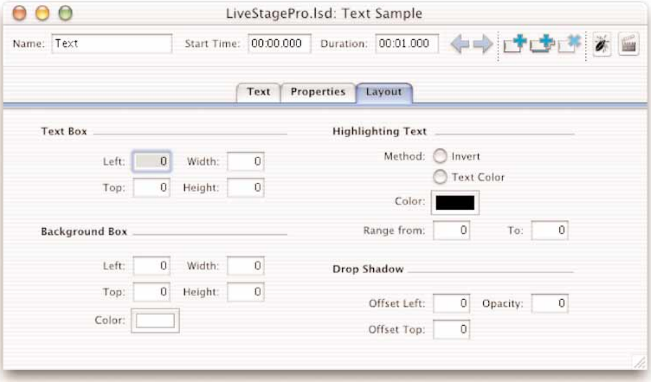
This option effectively turns the text sample into an image. While this can solve the formatting problems of using a font that the user might not have on their computer, it does present some drawbacks. First, as the sample in effect becomes an image, file size increases accordingly. Second, most functionality afforded a text sample, such as scrolling, interactivity, and highlighting is disabled, though the text is still searchable. Use this when you prefer an alternative to creating a graphic image of your text in an external application such as Photoshop.

Justification

The selection in the popup menu will determine how the text is justified in the Text Track or the text box. You can choose from Right, Left, and Center.

LAYOUT TAB

The Layout tab allows you to set a text box and a background box as well as add effects to the text itself.



Name: Text
Start Time: 00:00.000
Duration: 00:01.000

Text Box

Left: 0 Width: 0
Top: 0 Height: 0

Background Box

Left: 0 Width: 0
Top: 0 Height: 0
Color:

Highlighting Text

Method: ○ Invert
○ Text Color
Color: ■
Range from: 0 To: 0

Drop Shadow

Offset Left: 0 Opacity: 0
Offset Top: 0

Text Box

The Text Box is a bounding box for the text. The text you enter in to the sample will be constrained to the dimensions you specify in the Width and Height fields. The Text Box will be positioned at the coordinates you define in the Left and Top fields. Entering zero in all of these fields disables the Text Box. Make sure that your Text Box is always within the boundaries of the Text Track and Background Box (if set).

Background Box

The Background Box provides a colored background for your text. Set the Background Box's dimensions in the Width and Height fields and its position in the Left and Top fields. Again, you should ensure that your text box is within the boundaries of the Background Box and that the Background Box is within the boundaries of the Text Track itself.

Color

Click on the Color chip to select a background color. The background color will be constrained to the Background Box if you have one specified. If not, the background will fill the entire Text Track.

Highlighting Text

You can highlight sections of your text by changing or inverting the text color through the options here in the Highlighting Text section.

Invert

This inverts the current color of the text. Black text will be displayed as white and so on.

Text Color

Select this option and choose the color from the Color Chip to change the color of a selected range of text.

Color

This color will be used to highlight the text when you use the highlighting options.

Range From, To

Enter the numeric range of characters you wish to highlight, including spaces.

Drop Shadow

Through these fields, you can add a drop shadow to your sample's text to make it appear as though your text is casting a shadow.

Offset Left, Top

Specify in pixels how far to the right and how far down you want to offset the drop shadow.

Opacity

Specify how dark you want the drop shadow to be by entering a value from 0-255. Zero is off and 255 is black.

Scripting your Text Track

QScripts are added to the Text Track in the Text tab of the Sample Properties window.

You can assign scripting to the track as a whole, or to individual HotSpots. Track Event Handlers include the Frame Loaded, List Received, Idle, KeyPressed, and Mouse Moved Event Handlers, and do not require a HotSpot to be created. Simply select the Handler and add your script to the Script Edit window. The other option is to create and add scripting to HotSpots in your text.

1. Create a Text Track by clicking on the Text Track icon on the Track Creation Toolbar. T

   The Text Sample Properties window opens.

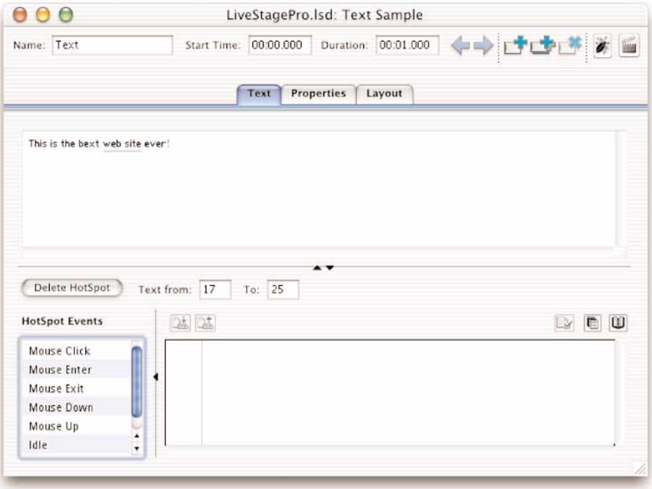
2. Double-click the text sample, either on the Stage or in the Timeline.

   The Sample Properties window opens.

3. In the Text Edit Field, enter “This is the best web site ever.”

4. Select the words “web site” and click on the ‘New HotSpot’ button.

   The words “web site” are outlined by a box in the Text Edit field and the HotSpot Events are displayed in the Event List.



LiveStagePro.lsd: Text Sample

Name: Text

Start Time: 00:00.000

Duration: 00:01.000

This is the bext web site ever!

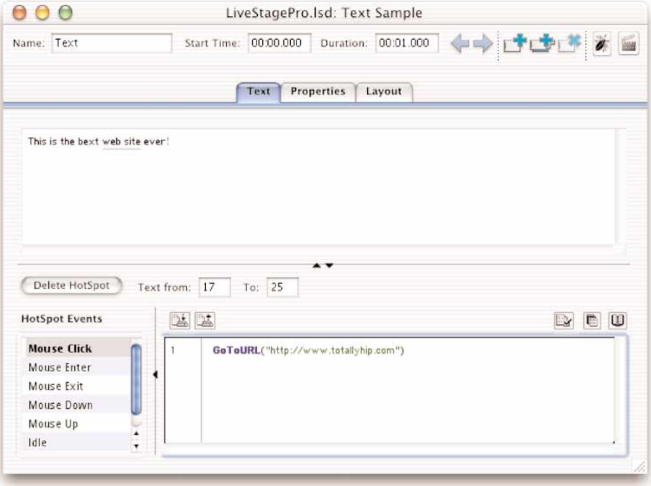
Text from: 17

To: 25

5. Click on the Mouse Click Event Handler.

6. Enter the following script in the Script Edit Field:

   GoToURL(“http://www.totallyhip.com”)



LiveStagePro.lsd: Text Sample

Name: Text   Start Time: 00:00.000   Duration: 00:01.000

This is the bext web site ever!

Text from: 17   To: 25

HotSpot Events

- Mouse Click
- Mouse Enter
- Mouse Exit
- Mouse Down
- Mouse Up
- Idle

```
GoToURL("http://www.totallyhip.com")
```

7. Click the 🎬 in the top right corner of the Sample Properties window to Preview your movie.

8. Click on the words “web site” in the preview movie.

   If you are connected to the Internet, your default browser will launch and load the Totally Hip web site.

Not only can you use HotSpots to create hot links as you can in web pages, but you can also build scripts that interact with sprites and other elements in your movie. See Chapter 16—Introduction to QScript to learn more about how to implement QScripts in scriptable tracks.

ADVANCED USE OF THE TEXT TRACK

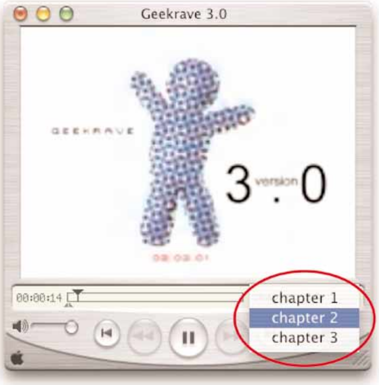
Alternate Languages

Text Tracks are often used very effectively in projects that want to offer more than one language version. Using Alternate Tracks (see Chapter 14—Working with Multiple Tracks, page 285), you can create several translations of your text track and QuickTime will automatically pick the version that is appropriate for the particular audience.

Chapter Tracks

Aside from captioning, one of the Text Track’s more powerful functions is in providing Chapter Tracks. Chapter Tracks provide labels—or “Chapters” that are displayed in the QuickTime player controller. These Chapters allow the user to jump to specific points in the movie’s timeline. Chapter Tracks can be created using Markers as described in detail in

Chapter 12—The Streaming Track, on page 236. However, Chapter Tracks created through the Text Track can also be attached to different Alternate Tracks, so that your French version contains French chapters, your German version contains German chapters, and so on.



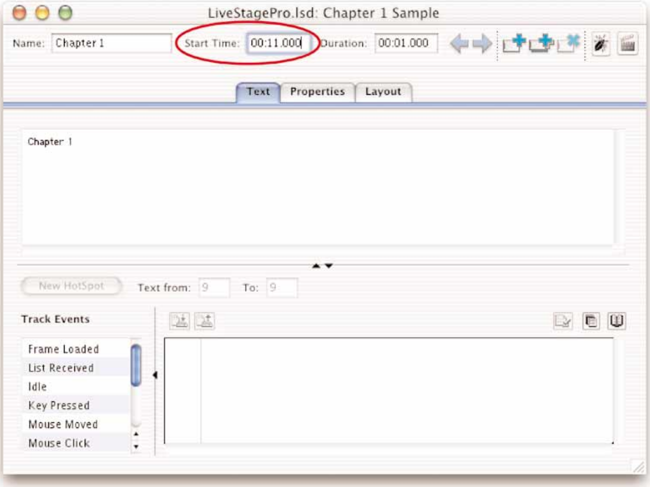
1. Create a Text Track by clicking on the Text Track icon on the Track Creation Toolbar. T

   The Text Sample Properties window opens.

2. In the Text Edit field in the text sample Properties window, enter the title of the first Chapter you want in your movie.

   The title is what the user will see in the QuickTime controller.

3. In the Start Time field, enter the point in time that you want to coincide with your Chapter.



LiveStagePro.lsd: Chapter 1 Sample

Name: Chapter 1 Start Time: 00:11.000 Duration: 00:01.000

Chapter 1

Text from: 9 To: 9

4. To create the next chapter, add a new sample to your Text Track by clicking on the button at the top right of the Text Sample Properties window.

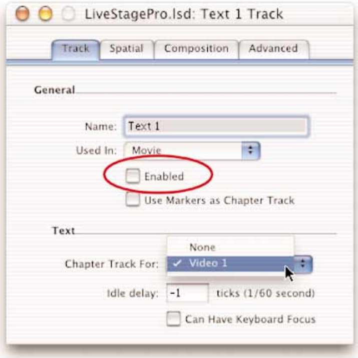
5. Repeat steps 2-4 for each Chapter you want to add to your movie.

   Close the Sample Properties window when you are done.

6. Select Window>Show Inspector (Command-I) to open the Text Track Inspector.

7. Uncheck Enabled.

   Disabling the Text Track will prevent the Chapter Track titles from showing up in the movie itself.



General

Name: Text 1

Used In: Movie

□ Enabled

□ Use Markers as Chapter Track

Text

Chapter Track For: None / ✓ Video 1

Idle delay: -1 ticks (1/60 second)

□ Can Have Keyboard Focus

8. From the “Chapter Track For:” popup menu, select the track to which you want to attach this Chapter Track.

HREF Tracks

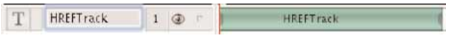
HREF Tracks are simply Chapter Tracks that instruct QuickTime to load URLs into an HTML frame. URLs can be loaded according to specific points in your movie’s timeline or user interaction. Generally, the QuickTime movie should be embedded in an HTML page when using this function.

1. Create an HTML frameset and individual frames.

   Make a note of the frame name(s) into which you will want to load the URLs.

2. Create a Text Track and change its name to “HREFTrack.”

   In order for the HREF Track to function, the name must be HREFTrack or QuickTime will consider the track to be just a Text Track.



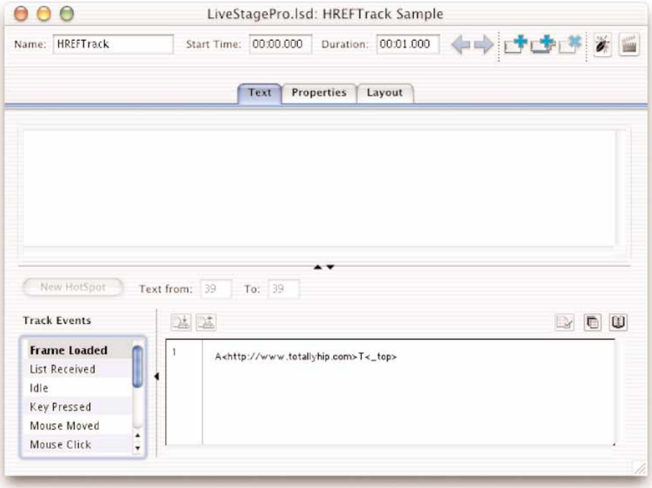
3. Create samples at each point in time that you wish to load a URL.

4. In the Text Edit Field of each sample, enter the URL you would like to load using the following syntax:

   ```
   A<http://myURL.com>T<Frame_Name>
   ```

   This instructs QuickTime to automatically load the URL into the target frame in the frameset.

Example: `A<http://totallyhip.com>T<_top>`



LiveStagePro.lsd: HREFTrack Sample

Name: HREFTrack Start Time: 00:00.000 Duration: 00:01.000

Text from: 39 To: 39

```
A<http://www.totallyhip.com>T<_top>
```

5. Export your movie and embed it into the appropriate frame.

TEXT TRACK WRAP UP

From Alternate Tracks, to Chapter Tracks to HREF Tracks, Text Tracks are obviously versatile and powerful. On the LiveStage Professional 4.0 CD-ROM, you will find a QuickTime movie called Text_Tester.mov, which will provide you with a taste of the effect certain QScript constants can have on text. Additionally, make sure to read Chapter 18—QTLists and XML to learn how text tracks can be dynamically updated with content from external XML files. Also, make sure to visit StageDoor (http://stagedoor.totallyhip.com) to view featured sites and tutorials that take advantage of the functionality the Text Track has to offer.

Chapter 7

The Sprite Track

SPRITE TRACK OVERVIEW

The Sprite Track is a scriptable track that contains individual interactive objects called sprites. It is generally considered the most important track for interactive QuickTime movies. Sprites are elements that can be layered, positioned, scaled, rotated, hidden, and animated. Each sprite sample can contain numerous sprites; each of which can have different interactive or animated functions attached to it. A sprite's visual properties can change according to the movie's time or through scripting. Some sprites are designed for user interaction while others work in the background performing functions and examining results in the movie.

WHEN TO USE THE SPRITE TRACK

A Sprite Track is useful when you want your user to interact with your movie, when you want elements of your movie to interact with one another, or perform functions. In short, when you want your movie to "do" something, create a Sprite Track.

Use a Sprite Track when you want to:

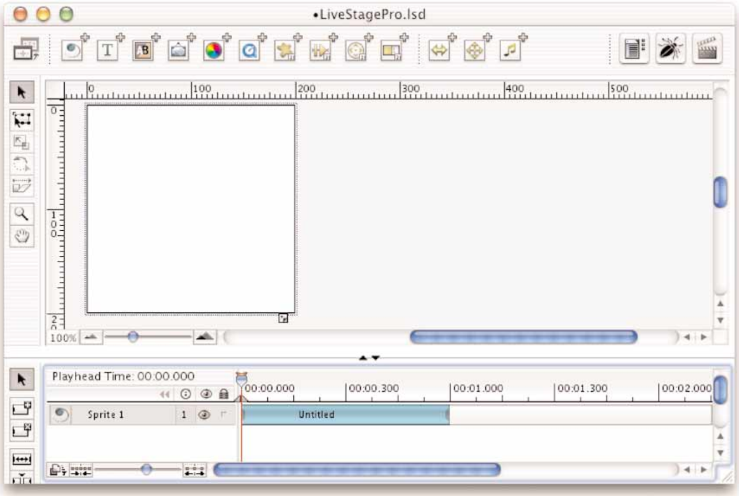
- Have buttons control or affect elements in your movie, such as playing, rewinding or fast-forwarding a video.
- Create simple animations.
- Have different elements in your movie, or even different movies, to communicate with, interact with, or control one another.
- Use a Behavior (see Chapter 17—Behaviors, page 333) to affect elements in your movie.
- Have your movie to perform calculations, test for conditions, or react to user input.
- Provide your audience with keyboard control.

WORKING WITH THE SPRITE TRACK

Sprite Tracks are created in one of three ways:

- Click the in the Track Creation Toolbar
- Select Movie>Create Track>Sprite (Option/Command-J).
- Control-click in the Track List area of the Timeline to open the track contextual menu, then select Create Sprite Track.

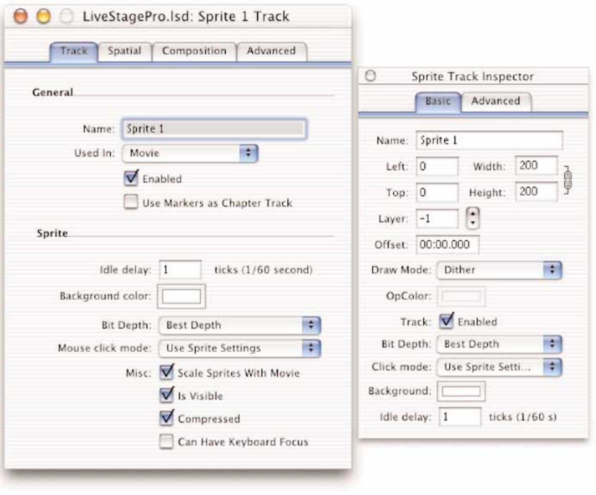
The new Sprite Track with dimensions of 200x200 appears with an empty sample in the Timeline and on the Stage.



Sprite Track Inspector / Properties Window

Depending on your preference, you can modify Sprite Track properties through either the Track Properties window or the Sprite Track Inspector. Access the Sprite Track Inspector by selecting Window>Show Inspector (Command-I), or view the Sprite Track Properties window by double clicking on the Sprite Track Header.

Both the Sprite Track Properties window and the Sprite Track Inspector contain the same standard set of properties found in most tracks. For a breakdown of the specific options available, see Chapter 3—Working in the Livestage Professional Environment, page 106. The Sprite Track's Inspector and Properties window also contain visual settings that are unique to Sprite Tracks, as well as settings that determine how the Sprite Track will play in your movie:



LiveStagePro.lsd: Sprite 1 Track

Track | Spatial | Composition | Advanced

General

Name: Sprite 1
Used In: Movie
☑ Enabled
☐ Use Markers as Chapter Track

Sprite

Idle delay: 1 ticks (1/60 second)
Background color: □
Bit Depth: Best Depth
Mouse click mode: Use Sprite Settings
Misc: ☑ Scale Sprites With Movie
☑ Is Visible
☑ Compressed
☐ Can Have Keyboard Focus

Sprite Track Inspector

Basic | Advanced

Name: Sprite 1
Left: 0 Width: 200
Top: 0 Height: 200
Layer: -1
Offset: 00:00.000
Draw Mode: Dither
OpColor: □
Track: ☑ Enabled
Bit Depth: Best Depth
Click mode: Use Sprite Setti...
Background: □
Idle delay: 1 ticks (1/60 s)

IDLE DELAY

The Sprite Track can receive Idle Events. The Idle Delay rate set in this field will determine how often the Text Track receives Idle Events. These are set in increments of 1/60th of a second. The default value is one tick, or 1/60 of a second. You can set a value of zero, which is the fastest, or you can set a value of -1 to turn it off. The Idle Delay is found in the Track Properties window in the Track tab and in the Sprite Track Inspector in the Basic tab. For more information on Idle Delay, see Chapter 16—Introduction to QScript, page 312.

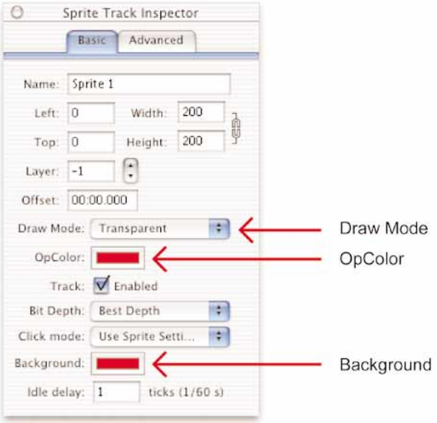
BACKGROUND COLOR

You can select a background color for your Sprite Track with this swatch. Click it to access the standard color picker. You will find this option in the Basic tab of the Sprite Track Inspector and in the Track tab of the Sprite Track Properties window.

Making a Sprite Track Transparent

- In the Background swatch, select a background color that does not appear in any of the sprites in your Sprite Track.

- Select 'Transparent' for the Draw Mode.



Sprite Track Inspector

Basic | Advanced

Name: Sprite 1
Left: 0 Width: 200
Top: 0 Height: 200
Layer: -1
Offset: 00:00.000
Draw Mode: Transparent
OpColor: 🟥
Track: ☑ Enabled
Bit Depth: Best Depth
Click mode: Use Sprite Setti...
Background: 🟥
Idle delay: 1 ticks (1/60 s)

If you're using the Sprite Track Properties window, this option is located in the Composition tab.

- In OpColor, select the exact color you selected as your Sprite Track's background in Step 1.

BIT DEPTH

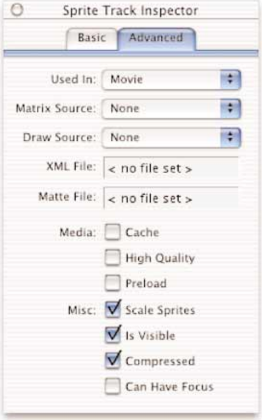
You can select the bit depth quality for your Sprite Track here. This can be set to 8-bit (256 colors), 16-bit (thousands of colors), 32-bit (millions of colors + Alpha Channel), or Best. When the 'Best' setting is selected, QuickTime selects the best bit depth based on the media in the Sprite Track and the user's monitor. The lower the bit depth, the less of a demand there will be on the memory and CPU of your user's computer. The Bit Depth setting is found in the Track tab of the Sprite Track Properties window and in the Basic tab of the Sprite Track Inspector.

MOUSE CLICK MODE

This menu allows you to designate which sprites in your Sprite Track will be able to receive mouse clicks. You can elect to have all your sprites clickable, none clickable, or allow the individual sprite's settings to determine the clickable status. This setting is found in the Track tab of the Sprite Track Properties window and in the Basic tab of the Sprite Track Inspector as 'Click mode'.

SCALE SPRITES WITH MOVIE

This checkbox affects vector images used in your Sprite Track. If it is checked, vector images will scale smoothly as the movie is scaled. If it is disabled, vector images will have rough edges and will display artifacts when scaled. You can find this option in the Track tab of the Sprite Track Properties window and in the Advanced tab of the Sprite Track Inspector.



Sprite Track Inspector

Used In: Movie
Matrix Source: None
Draw Source: None
XML File: < no file set >
Matte File: < no file set >
Media: □ Cache
□ High Quality
□ Preload
Misc: ☑ Scale Sprites
☑ Is Visible
☑ Compressed
□ Can Have Focus

IS VISIBLE

This option indicates whether the Sprite Track will be visible when the movie is first launched. Choose this option in the Track tab of the Sprite Track Properties window or in the Advanced tab of the Sprite Track Inspector. The visible state can be toggled via QScript.

COMPRESSED

Selecting this option compresses the Sprite Track's sample data, resulting in smaller file size. The Compressed option can be found in the Track tab of the Sprite Track Properties window, or in the Advanced tab of the Sprite Track Inspector.

tip

Decompressing compressed samples takes a little time and QuickTime not only loads the compressed sample, but also holds the decompressed sample in RAM simultaneously. If you experience drawing issues during playback of your movie, deselecting the 'Compressed' setting may help.

CAN HAVE (KEYBOARD) FOCUS

Selecting this option allows the Sprite Track to receive keyboard input from the user. Can Have (Keyboard) Focus is found in the Track tab of the Sprite Track Properties window, and in the Advanced tab of the Sprite Track Inspector.

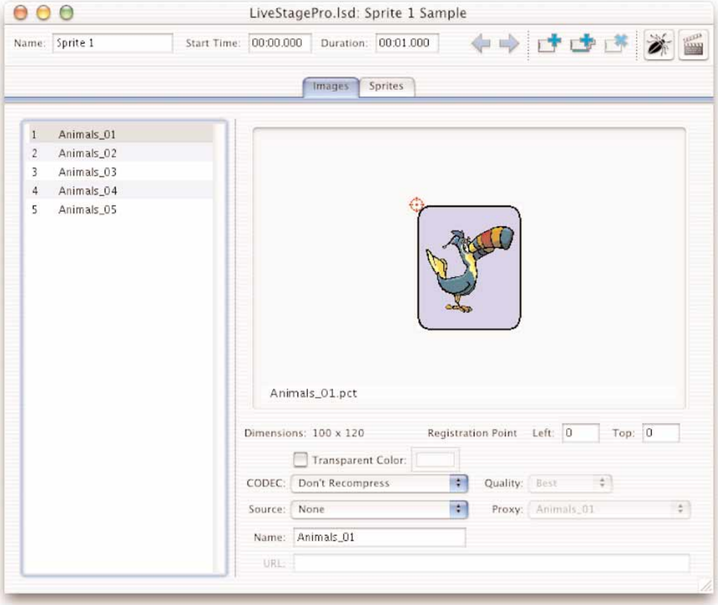
Sprite Sample Properties Window

Access the Sprite Sample Properties window by double clicking on a Sprite Track sample. The Sprite Sample Properties window contains standard fields and tools along the top of the window as described in Chapter 3—Working in the Livestage Professional Environment, page 90. These tools allow you to name the sample, adjust its start time and duration as well as navigate through, add, and delete samples.

Two tabs make up the Sprite Sample Properties window: the Images tab and the Sprites tab. As the primary component of a sprite is an image, the Images tab is what first appears when you open an empty Sprite Track's Sample Properties window.

THE IMAGES TAB

Sprites are often referred to as actors on a stage. As actors can have many costumes, sprites can have many images. The Images tab houses all the images for the sprites in the sample. To add images to the sprite sample, drag them from the Library into the Image List, located on the left side of the window.



LiveStagePro.lsd: Sprite 1 Sample

Name: Sprite 1 Start Time: 00:00.000 Duration: 00:01.000

1 Animals_01
2 Animals_02
3 Animals_03
4 Animals_04
5 Animals_05

Animals_01.pct

Dimensions: 100 x 120 Registration Point Left: 0 Top: 0

□ Transparent Color:

CODEC: Don't Recompress Quality: Best

Source: None Proxy: Animals_01

Name: Animals_01

URL:

IMAGE LIST

When you drag images to the Image List, they are assigned consecutive Index Numbers. You can use these numbers in your QScript to refer to the images in this list.

Example: `SpriteNamed(“MySprite”).SetImageIndexTo(3)`

This will change the image associated with the sprite called “MySprite” to the third image in the sprite sample’s Image List. You can also refer to the image’s name in your QScript:

Example: `SpriteNamed(“MySprite”).SetImageIndexTo($”MyImage”)`

You can drag to reorder your images in the Image List and therefore change their Index Numbers. You can also edit the image names in the Name field near the bottom of the window or in place by clicking on them in the Image List.

Preview Window

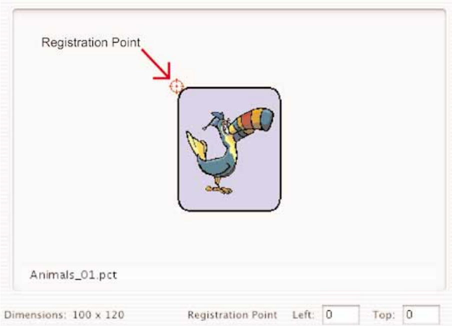
When an image in the Image List is selected, it appears in the Preview window to the right. The image file’s name and dimensions are displayed below. To replace the graphic used in an image, drag and drop a new image from the Library or your computer into the Preview window. The name and Index in the Image List remain the same, but the associated image is changed.

tip

You may also want to adjust a Registration Point if you are rotating a sprite as sprites rotate around the Registration Point of their images.

Registration Point

The Registration Point is the positioning anchor for an image. This is the point of the image that sits on the sprite’s Top and Left coordinates. The Registration Point is initially located at an image’s top left corner. If you change the sprite’s coordinates to Top: 10, Left: 20, the image will be repositioned with its top left corner at 10, 20. Alternately, if you place a sprite image’s registration point at the center of the image, the image will be centered over any coordinates you specify.



Animals_01.pct

Dimensions: 100 x 120 Registration Point Left: 0 Top: 0

You can alter an image’s Registration Point in a few ways:

- Edit the Left and Top values in the Registration Point fields.
- Click directly on the preview image. The Registration Point will move to the point where you clicked.
- Click on the Preview Pane and then use the arrow keys to nudge the registration point. Holding down Shift while nudging will move the Registration Point in 10-pixel increments.

Transparent Color

You can specify a color to appear transparent in an image. Enable transparency by clicking on the Transparent Color check box, select the color from the box, and then choose ‘Animation’ from the codec popup. Other codecs do not support transparency. The image will be transparent in any sprite in which it is used, and any mouse events will only work on its non-transparent areas.

Codec

The codec popup menu offers different codecs (Compressor/Decompressor) that you can use on the images in your sprites. This option can be convenient when the image has not yet been compressed and you want to reduce overall file size. If your image has already been compressed, select ‘Don’t Recompress’, as compressing a previously compressed image can result in poor image quality and can even increase file size. See Appendix 3—QuickTime Codec Reference, for complete information on the codecs available.

Quality [New! in LiveStage Professional 4.0]

This popup gives you a higher degree of control over the quality of an image when you apply a codec in the codec menu. You can select from Best, High, Medium, and Low to balance quality with file size.

Source

From this popup menu, you can specify an alternate source to override the image’s data. Alternative image sources can come from another track like a Video Track. Thus, you can have sprites that are videos. Image overrides can even come from an external URL. See ‘Image Override” in the Advanced Use of the Sprite Track section, page 178.

Proxy

If you select a ‘Get From URL’ source in the Source popup menu, choose one of the static images in this popup for the sprite to display until the data is available from the URL. The images in this popup are those stored in the current sprite sample.

Name

The image’s original file name can be edited in this field. Use this name in quotation marks preceded by a “$” sign when referring to the image by name in your QScripts.

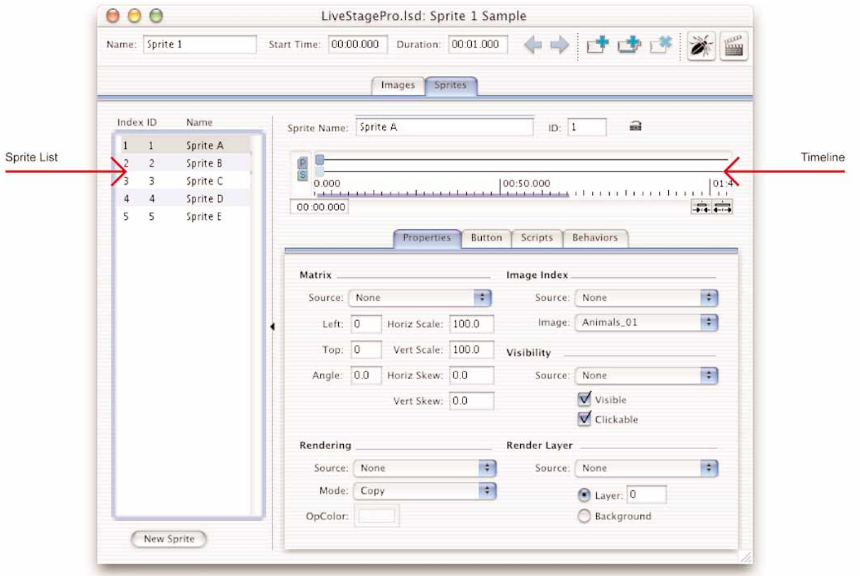
Example: `$“ImageName”`

URL

This is where you specify the URL to be used when the ‘Get from URL’ setting is selected in the Source Popup Menu.

SPRITES TAB

Sprites are created and augmented in the Sprites tab. The Sprites tab is made up of three sections; the Sprite List, the Timeline and a section containing four tabbed categories: Properties, Button, Scripts, and Behaviors.



LiveStagePro.lsd: Sprite 1 Sample

Name: Sprite 1 Start Time: 00:00.000 Duration: 00:01.000

Index	ID	Name
1	1	Sprite A
2	2	Sprite B
3	3	Sprite C
4	4	Sprite D
5	5	Sprite E

Sprite Name: Sprite A ID: 1

00:00.000

Matrix
Source: None
Left: 0 Horiz Scale: 100.0
Top: 0 Vert Scale: 100.0
Angle: 0.0 Horiz Skew: 0.0
Vert Skew: 0.0

Image Index
Source: None
Image: Animals_01

Visibility
Source: None
☑ Visible
☑ Clickable

Rendering
Source: None
Mode: Copy
OpColor:

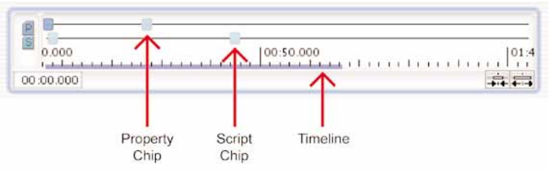
Render Layer
Source: None
● Layer: 0
○ Background

To have LiveStage Professional automatically place chips on the Sprite Timeline when you modify a sprite on the Stage at various points in time, select "Auto create new Sprite properties" in the Misc tab of the Preferences Window. To access the Preferences, select LiveStage Pro> Preferences through the Main Menu Bar.

Sprite Timeline

The Sprite Timeline allows you to change the properties and scripts of your sprite at different points in time. Through 'chips' laid out along a sprite-specific Timeline, this area provides a graphic means to access and edit properties and scripts that affect the sprite over time. You can use the chips in the Sprite Timeline when you want:

- to create a simple animation;
- a sprite to jump to different areas in the sprite track;
- to change the interactivity of a sprite at particular points in time; or
- to have specific changes made to a sprite at certain times in your movie.



Timeline

The Sprite Timeline is similar to the Timeline found in the Document Window, but is specific to the current sprite sample. The blue bar along the bottom of the timeline depicts the start time and the duration of the sample. The current time in the movie set by the chip that is currently selected, is displayed in the bottom left corner. The familiar Zoom In and Zoom Out buttons allow you to view the sample duration in its entirety.

Property and Script Chips

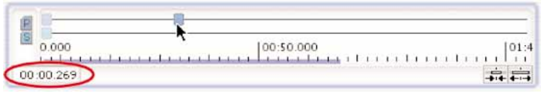
To the left of the Sprite Timeline there are two ‘chips’ labeled with a P and an S respectively. These markers represent changes in the sprite’s Properties and Scripts.

To create a new Property chip:

1. Select a sprite in the Sprite List.
2. Click on the P chip in the Sprite Timeline.

   A new chip appears in the timeline partially over the starting chip.
3. Drag the new chip to the point in time where you wish to modify the sprite.

   The time display changes to reflect the chip’s time.



4. With the new chip selected, enter any new properties that you wish to assign to your sprite for this particular point in time in the Properties tab.
5. Repeat steps 1 through 4 for any additional chips you wish to create.

Script chips are created in the same manner, except that you click on the S chip and make alterations in the Scripts tab of the Sprite Sample Properties window. You can also Cut, Copy, Paste, and Duplicate chips through the Edit menu. Just select the chip and choose Copy and then Paste, or select Duplicate. The copied chip will show up slightly offset from the original chip. You can then drag it to the desired point in the Timeline. To delete a chip, select it and then hit the delete key. The first chip cannot be deleted.

When you click on a chip in the Sprite Timeline, the sprite’s settings and scripts at that particular time will display in the Sprite Sample Properties window. The Stage and main Timeline will also update to reflect the movie at that point in time.

Sprite List

The Sprite List holds all the sprites available in the sprite sample. To create a new sprite, click the New Sprite button located below the Sprite List. Sprites in the Sprite List are assigned sequential Index and ID Numbers that can be referred to in your QScripts. You can reorder sprites in the Sprite List by dragging them to new positions in the list. This will alter the sprite’s Index number, but not ID. To edit a sprite’s name or ID number, click on the sprite in the Sprite List and then click the name and ID to edit them in place. You can also edit a sprite’s Name and ID through the fields at the top of the Sprites tab.

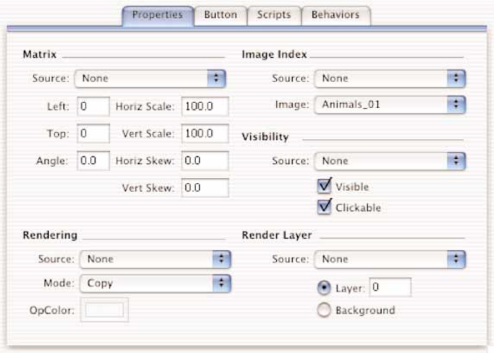
Clicking on a sprite selects it and activates the tabbed subsections to the right of the Sprite List so that you can alter the sprite’s properties and enter QScripts.

tip

Dragging sprites to a new position in the Sprite List changes the Index Number for the sprite, so keep that in mind if you are using Index Numbers in your QScript. You can avoid this problem by referring to sprites in your scripts by their name or ID instead.

Properties Tab

The Properties tab allows you to view and modify the visual aspects of each of your sprites.



Matrix
Source: None
Left: 0 Horiz Scale: 100.0
Top: 0 Vert Scale: 100.0
Angle: 0.0 Horiz Skew: 0.0
Vert Skew: 0.0

Image Index
Source: None
Image: Animals_01

Visibility
Source: None
☑ Visible
☑ Clickable

Rendering
Source: None
Mode: Copy
OpColor:

Render Layer
Source: None
● Layer: 0
○ Background

Matrix

The Matrix determines the position, angle, scale, and skew of a sprite.

- *Source:* If you would like the Matrix values of your sprite to change over time, you can create a Tween or Modifier Track and then attach it to the sprite through the Source popup.

- *Left/Top:* You can specify the sprite's coordinates in the Sprite Track by adjusting these values or by clicking and dragging the sprite on the Stage. You can enter either positive or negative values for coordinates. If the left/top coordinates are such that the Sprite will fall outside of the bounds of your Sprite Track, the sprite will not be visible, nor will your user be able to interact with it.

- *Angle:* Angle determines the degrees of rotation of the sprite. All content displayed in the sprite will be rotated accordingly. You can rotate your sprite to a particular angle by entering a value in degrees here, or by rotating it physically on the Stage (see Sub-Object Editing Mode in this chapter). A sprite will rotate around its Registration Point.

- *Horizontal Scale:* You can increase or decrease the horizontal scale (width) or your sprite by entering percentage values in this field. Entering a value of 25.0 will shrink the sprite's horizontal size to 25% of its normal size. Entering a value of 150.0 will enlarge the horizontal size of the sprite to 150% of its original size. You can also adjust the scale using the Sub-Object editing tools (See Sub-Object Editing Mode).

- *Vertical Scale:* You can increase or decrease the vertical size (height) of your sprite by entering percentage values in this field. A value of 75.0 will shrink the sprite's vertical size to 75° of its normal size. A value of 100.0 will display the vertical size of the sprite at its original size. You can also adjust the scale using the Sub-Object editing tools (See Sub-Object Editing Mode).

- *Horizontal Skew:* This slants the sprite along its horizontal axis according to a percentage value. You can also use the Sub-Object Editing Mode to affect these changes directly on the Stage. The skew values range from -500 to 500, negative numbers skewing to the left, positive to the right, and 0 being neutral (no skewing).

tip

Double clicking a sprite in the Sprite List opens it in its own editing window. This can come in handy when you want to compare two sprites side by side.

- *Vertical Skew:* This slants the sprite along its vertical axis according to a percentage value. Again, you can also use the Sub-Object Editing Mode to affect these changes directly on the Stage. The skew values range from -500 to 500, negative numbers skewing upwards, positive downwards, and 0 being neutral (no skewing).

Rendering

This section is where you can modify how the sprite will display its content.

- *Source:* As with the Matrix, you can select a Modifier or Tween Track to handle how the sprite will display. You can create interesting visual effects using this method (see Chapter 13—Additional LiveStage Professional Tracks).

- *Mode:* This is how the sprite will draw its content. The default mode is Copy, which simply 'copies' the sprite's visual data and draws it onto the screen. The image will be dithered as necessary to accommodate the bit depth specified in the Sprite Track Properties window/Inspector. For more information on the Drawing Modes available in this popup, go to Appendix 2—Drawing Mode Reference.

- OpColor: Modes, such as Transparent among others, will use a color value to alter how the sprite's image is drawn. Use this in much the same way as in the 'Making a Sprite Track Transparent' section.

Image Index

This is where you specify the image data to be used by the sprite.

- *Source:* You can create a Tween or Modifier Track that will change the Image Index property over time. Select that track here. For more information, see Chapter 13—Additional LiveStage Professional Tracks, pages 259 and 273.

- *Image:* In this popup menu select the image you would like the sprite to display initially. The images listed here are those that are housed in the sprite sample's Images tab. The sprite's image can be changed from this initial image by QScript or through the Property chip in the Sprite Timeline (see Sprite Timeline on page 164 of this chapter).

tip

If you have a sprite that does not move or change, select the background option for it. This allows QuickTime to optimize the sprite's drawing.

Visibility

This section allows you to determine your sprite's visibility as well as whether or not it will be clickable.

- *Source:* You can select a Tween or Modifier Track source to alter the sprite's visibility over time, turning it on or off.

- *Visible:* Deselecting this checkbox will render the sprite invisible. Visibility can also be altered through QScript. Invisible sprites can still receive mouse events, even if the user cannot see them.

- *Clickable:* If you do not want the sprite to be able to receive mouse events, deselect this option.

Render Layer
This determines the drawing order of the individual sprites in the Sprite Track.

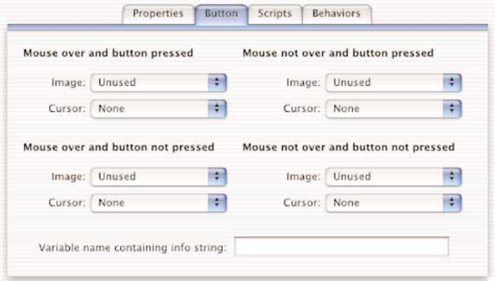
- *Source:* This specifies a Tween or Modifier Track as a Source for the Layer property. This can come in handy for moving the sprite in front of or behind other sprites within the same Sprite Track over time.

- *Layer:* This is the numeric value representing the sprite's layer. Multiple sprites in a Sprite Track have drawing layers just as tracks do. Sprites within the same sprite track that have lower layer numbers will display in front of those with higher layer numbers. You can specify positive or negative values ranging from -999 to 9999 in this field.

- *Background:* Selecting the Background option will position the sprite behind all other sprites in the same track.

Button Tab

Through the Button tab, you can make your sprite act like a button. A button appears in different visual states according to how the Mouse is interacting with it.



Mouse over and button pressed
Image: Unused
Cursor: None

Mouse not over and button pressed
Image: Unused
Cursor: None

Mouse over and button not pressed
Image: Unused
Cursor: None

Mouse not over and button not pressed
Image: Unused
Cursor: None

Variable name containing info string:

The Button tab offers four different Mouse interaction scenarios, each of which has Image and Cursor options:

Mouse over and button pressed
The user is clicking on the sprite. (Mouse Click or Mouse Down event).

Mouse over and button not pressed
The cursor is over the sprite but no Mouse down action has taken place. (Mouse Over Event).

Mouse not over and button pressed
This interaction occurs when the user clicks on the sprite and drags off with the mouse still pressed.

Mouse not over and button not pressed
The user is not interacting with the sprite and the Mouse is not pressed down.

In the Button tab, you can set these different visual states as well as change how the cursor will appear when interacting with your sprite.

Image

You can create rollover effects for your buttons by selecting images from this popup menu to correspond with each of the different Mouse interactions. The popup displays the images found in your sprite sample Images tab. If you select “None” from this list, the sprite will appear with the default initial image you specified in the Properties tab.

Cursor

You can select an alternate cursor to display for each mouse event. This can give further feedback to the user that the sprite/button is active. If “None” is selected, the cursor will not change with the Mouse action. The available cursors to choose from are:

- None
- Open Hand
- Closed Hand
- Pointing Hand
- Right Arrow
- Left Arrow
- Down Arrow
- Up Arrow

tip

QuickTime 3 will not display any alternate cursors.

Variable name containing info string:

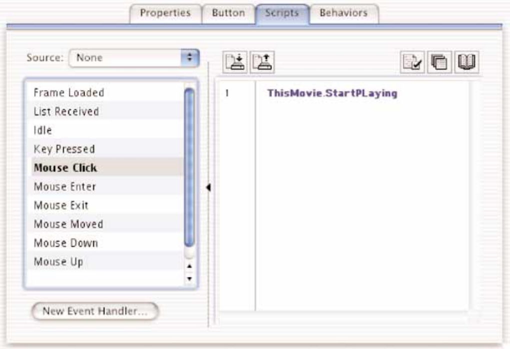
This field allows you to display information in a web browser’s status bar or a popup window when you are delivering your movie in a web browser. Declare a Global Variable somewhere in your script:

```
GlobalVars myStatusStringVar

SetString(myStatusStringVar, “This is a sample status
string”)
```

Then, enter the Global Variable name into the ‘Variable name containing info string’ field in the Button tab.

Scripts Tab

Add QScript to your sprite in the Scripts tab. The Scripts tab has the same layout and features that are found in other scriptable tracks’ Script Edit Windows (for detailed information, see Chapter 16—Introduction to QScript, page 311). The Scripts tab is separated into the Event Handler List and the Script Editor window by a Resize Bar. Clicking on the Resize Bar’s arrow maximizes the scripting area. You can also resize the area by dragging the Resize Bar.



```
ThisMovie.StartPLaying
```

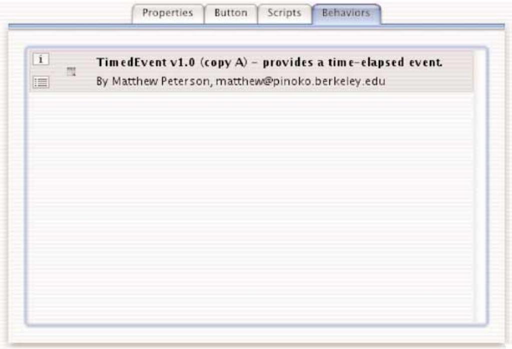
Source
You can create a Modifier or Tween Track that can trigger a Custom Event in a particular sequence or at a certain time. Select the Tween or Modifier using this dropdown menu.

Event Handler List
The Event Handler List displays all the Event Handlers for the selected Sprite. For a breakdown of the different Event Handlers, see Chapter 16—Introduction to QScript, page 312. When an Event Handler name appears in boldface, it means a script is attached to it.

New Event Handler Button
Clicking the New Event Handler creates a Custom Event. Custom Events and how to create them are described in detail in Chapter 16—Introduction to QScript, on page 313. When a Custom Event is created, it will appear in boldface in the Event Handler List. To remove a Custom Event, select it and press the delete key.

Behaviors Tab

You can add Behaviors to your sprite through the Behavior tab. Behaviors are pre-made scripts that you drag from the Scripts tab of the Library Window into the Behavior tab to easily add interactivity and functionality to your sprites. You can add multiple Behaviors to a single sprite. Behaviors appear in the Behavior tab displaying an identifying icon, as well as the Behavior name, a short description, and information on the author. Most Behaviors, once dragged, also provide icons that allow you to access additional information and assign parameters as well. Behaviors are covered in detail in Chapter 17—Behaviors, starting on page 335.



TimedEvent v1.0 (copy A) – provides a time-elapsed event.
By Matthew Peterson, matthew@pinoko.berkeley.edu

Creating Sprites

First, create a Sprite Track using one of the methods described at the beginning of the Working with the Sprite Track section on page 158.

CREATING SPRITES THROUGH THE SAMPLE PROPERTIES WINDOW

1. Double-click the sprite sample.

   The Sprite Sample Properties window opens.

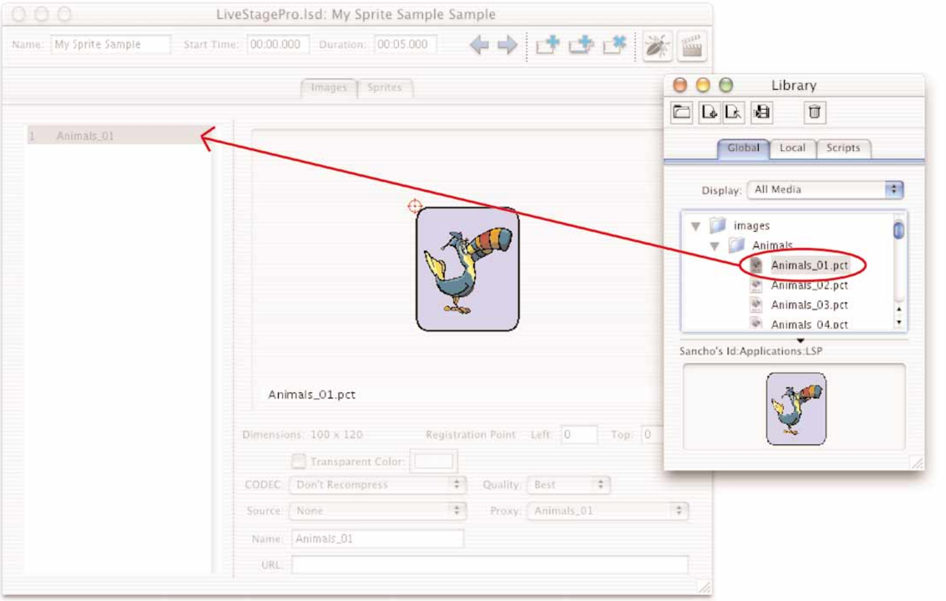
2. In the ‘Name’ field at the top of the Images tab in the Sample Properties window, enter “My Sprite Sample”.

3. In the Images Folder in the Global tab of the Library window, open the Animals folder.

   Drag one of the Animal images to the Images tab of the Sprite Sample Properties window. A sprite must be always be associated with an image, though it does not necessarily need to be visible.

tip

All sprites require an image, but sometimes you will want to use a sprite to perform functions that do not require it to be visible or clickable. Keep file size and resource usage to a minimum by using a 1-pixel by 1-pixel solid color GIF file for such sprites.



LiveStagePro.lsd: My Sprite Sample Sample

Name: My Sprite Sample
Start Time: 00:00.000
Duration: 00:05.000

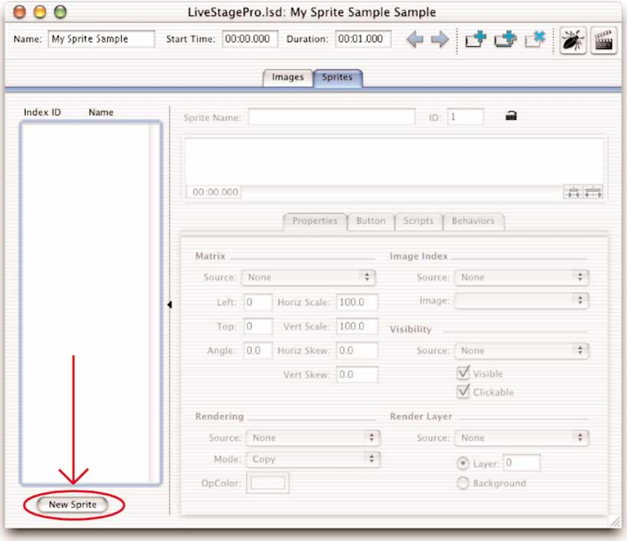
1 Animals_01

Animals_01.pct

Dimensions: 100 x 120
Registration Point Left: 0 Top: 0
□ Transparent Color:
CODEC: Don't Recompress
Quality: Best
Source: None
Proxy: Animals_01
Name: Animals_01
URL:

4. Click the Sprites tab.

5. Click the ‘New Sprite Button’.



LiveStagePro.lsd: My Sprite Sample Sample

Name: My Sprite Sample Start Time: 00:00.000 Duration: 00:01.000

Sprite Name: ID: 1

00:00.000

Matrix

Source: None

Left: 0 Horiz Scale: 100.0

Top: 0 Vert Scale: 100.0

Angle: 0.0 Horiz Skew: 0.0

Vert Skew: 0.0

Image Index

Source: None

Image:

Visibility

Source: None

☑ Visible

☑ Clickable

Rendering

Source: None

Mode: Copy

OpColor:

Render Layer

Source: None

● Layer: 0

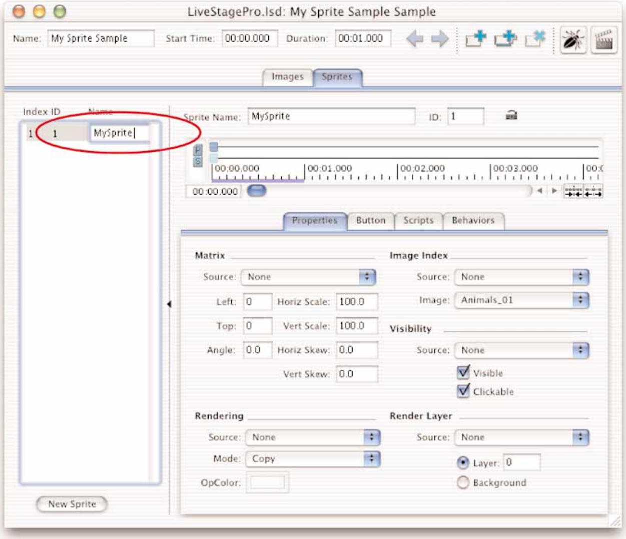
○ Background

A new sprite called ‘Untitled’ appears in the Sprite List and on the Stage.

6. Click on the ‘Untitled’ name in the Sprite List.

The in-place editor becomes active.

7. Change the name of the sprite to “MySprite”



LiveStagePro.lsd: My Sprite Sample Sample

Name: My Sprite Sample Start Time: 00:00.000 Duration: 00:01.000

1 1 MySprite

Sprite Name: MySprite ID: 1

00:00.000 00:01.000 00:02.000 00:03.000

00:00.000

Matrix

Source: None

Left: 0 Horiz Scale: 100.0

Top: 0 Vert Scale: 100.0

Angle: 0.0 Horiz Skew: 0.0

Vert Skew: 0.0

Image Index

Source: None

Image: Animals_01

Visibility

Source: None

☑ Visible

☑ Clickable

Rendering

Source: None

Mode: Copy

OpColor:

Render Layer

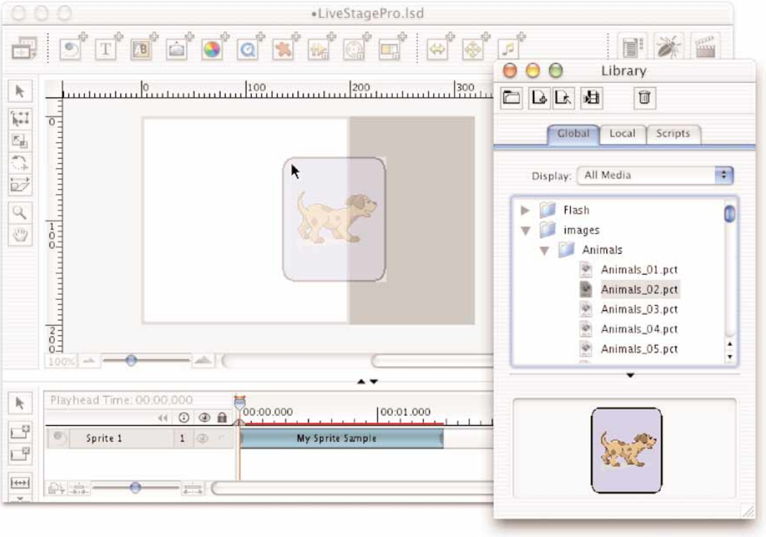
Source: None

● Layer: 0

○ Background

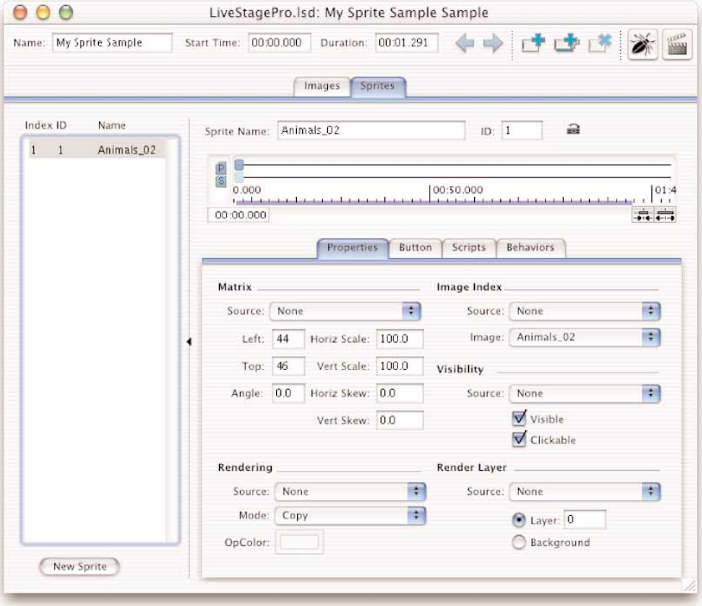
CREATING SPRITES THROUGH DRAG AND DROP

1 From the Library window, drag an image directly to the Stage or over the sprite sample in the Timeline. The sprite and its image appear on the Stage.



2. Double-click the sprite sample to open the Sample Properties window.

   The Sprites tab displays and the image name appears in the Sprite List.



Name: My Sprite Sample
Start Time: 00:00.000
Duration: 00:01.291

Index	ID	Name
1	1	Animals_02

Sprite Name: Animals_02
ID: 1

Matrix
Source: None
Left: 44 Horiz Scale: 100.0
Top: 46 Vert Scale: 100.0
Angle: 0.0 Horiz Skew: 0.0
Vert Skew: 0.0

Image Index
Source: None
Image: Animals_02

Visibility
Source: None
☑ Visible
☑ Clickable

Rendering
Source: None
Mode: Copy
OpColor:

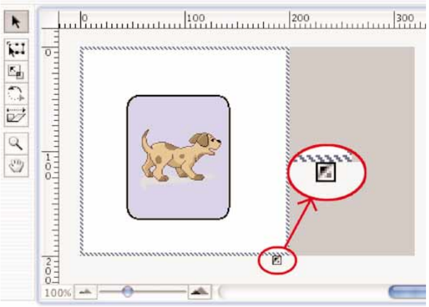
Render Layer
Source: None
● Layer: 0
○ Background

Sub-Object Editing Mode

You can modify the physical properties of your sprite either through the Properties tab of the Sprite Sample Properties window, or directly on the Stage using the Sub-Object Editing Mode. The Sub-Object Editing Mode allows you to manipulate individual sprites within the Sprite Track.

Once you have created a Sprite Track and a sprite:

1. Click the Sub-Object Editing Mode button at the bottom right hand corner of the Sprite Track on the Stage.



The Sprite Track becomes active.

2 Click on the sprite within the Sprite Track.

A bounding box appears.

3. Use the Transformation tools to the left of the Stage to modify the position, size, rotation, and skew of your sprite. Click the tool in the toolbar first and then click the sprite to modify it.

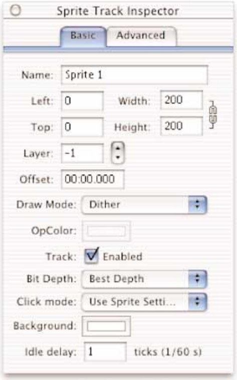
ICON	FUNCTION
↖	Selects and positions the sprite within the Sprite Track. Select multiple sprites by Shift-clicking.
⬚	Provides all transformation handles (scale, skew, and rotate) on the sprite simultaneously.
⧉	Scale the sprite by click-dragging on the handles.
↻	Rotate the sprite by click-dragging on the rotate control handle.
▱	Skew the sprite by click-dragging on the handles.

tip

With the positioning tool selected, hold down the Option key when you have a sprite selected in the Sprite Track in Sub-Object Editing Mode. This will display its dimensions in pixels. Hold down the Option key while moving your mouse in the Sprite Track to reveal the distances from the selected sprite to other sprites in the track as well as to the track boundaries.

Sprite Inspector

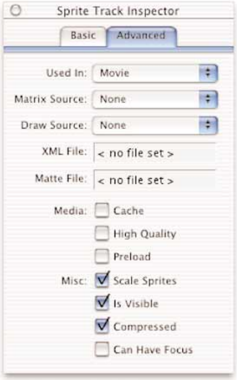
The Sprite Inspector is accessed by clicking on an individual sprite on the Stage while the Sprite Track's Sub-Object Editing Mode is activated. The Sprite Inspector includes most of the fields that are found in the Properties and Button tabs of the Sprite Sample Properties window (see the Sprite Sample Properties Window section earlier in this chapter). This makes it easy for you to view and modify the properties of your sprites without ever leaving the Stage. To view the Inspector, select Show Inspector (Command-I) from the Window menu.



Sprite Track Inspector

Basic | Advanced

Name: Sprite 1
Left: 0 Width: 200
Top: 0 Height: 200
Layer: -1
Offset: 00:00.000
Draw Mode: Dither
OpColor:
Track: ☑ Enabled
Bit Depth: Best Depth
Click mode: Use Sprite Setti...
Background:
Idle delay: 1 ticks (1/60 s)



Sprite Track Inspector

Basic | Advanced

Used In: Movie
Matrix Source: None
Draw Source: None
XML File: < no file set >
Matte File: < no file set >
Media: □ Cache
□ High Quality
□ Preload
Misc: ☑ Scale Sprites
☑ Is Visible
☑ Compressed
□ Can Have Focus

Scripting your Sprite Track

The Sprite Track is the most flexible of all the tracks when it comes to scripting. In addition to the Event Handlers found in other scriptable tracks, you can create and execute Custom Events through the Sprite Track (see Chapter 16—Introduction to QScript, page 313). As demonstrated in the Sprite Timeline section, scripts can also be added at particular points in time to any number of Event Handlers.

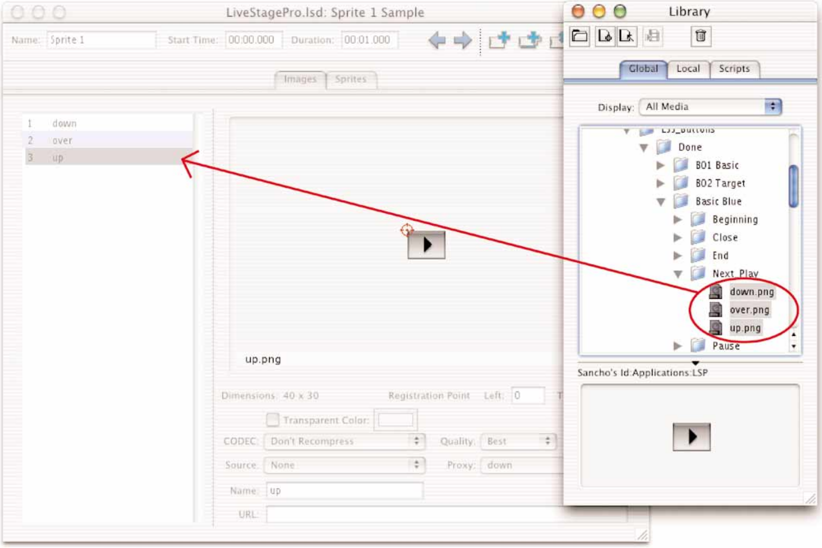
Scripts are added to Sprite Tracks in the Scripts tab located in the Sprites tab of the Sample Properties window.

1. Create a Sprite Track using one of the methods described at the beginning of the 'Working with the Sprite Track' section in this chapter on page 158.

2. Double-click the sprite sample to open the Sample Properties window.

3. From the Global tab of the Library window, click the triangles next to Images>Interface Elements>Linear Control Buttons>Basic Blue>Next_Play.

4. Select and drag the 'down.png', 'up.png', and 'over.png' images to the Images List in the Sample Properties window's Images tab.



LiveStagePro.lsd: Sprite 1 Sample

Name: Sprite 1 Start Time: 00:00.000 Duration: 00:01.000

Images | Sprites

1 down
2 over
3 up

up.png

Dimensions: 40 x 30 Registration Point Left: 0
□ Transparent Color:
CODEC: Don't Recompress Quality: Best
Source: None Proxy: down
Name: up
URL:

Library

Global | Local | Scripts

Display: All Media

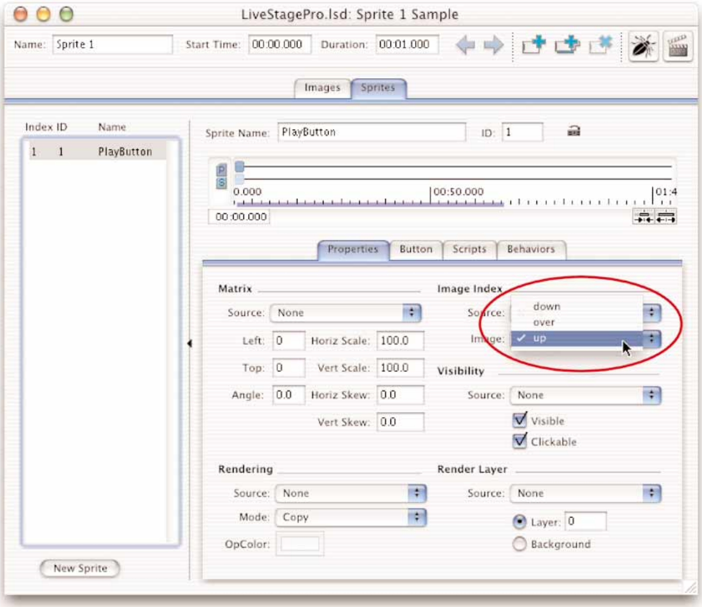
tip

The Photoshop .psd file format is a media type that QuickTime understands. Therefore, when you bring a .psd file complete with layers into LiveStage Professional, you can access the individual layers as well as the flattened image. This can be useful for creating and easily incorporating button states. The layered .psd file will appear in the Library as a single image with an expansion arrow. Click the arrow to reveal the individual layers.

5. Click on the Sprites tab.

6. Click on the ‘New Sprite’ button and in the Sprite Name field above the Sprite Timeline change the name to PlayButton.

7. From the Image Index popup menu in the Properties tab, make sure “up” is selected as the Image.


LiveStagePro.lsd: Sprite 1 Sample

Name: Sprite 1
Start Time: 00:00.000
Duration: 00:01.000

Index ID | Name: 1 | 1 | PlayButton

Sprite Name: PlayButton
ID: 1

Matrix
Source: None
Left: 0
Top: 0
Angle: 0.0
Horiz Scale: 100.0
Vert Scale: 100.0
Horiz Skew: 0.0
Vert Skew: 0.0

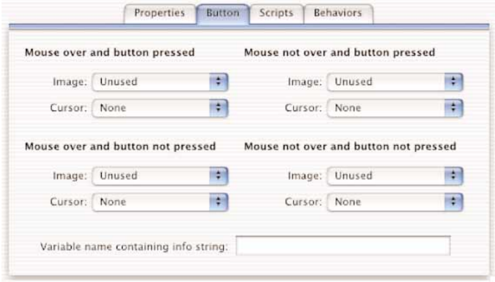
Image Index
Image: ✓ up

Visibility
Source: None
☑ Visible
☑ Clickable

Rendering
Source: None
Mode: Copy
OpColor:

Render Layer
Source: None
● Layer: 0
○ Background

8. Click the Button tab.


Mouse over and button pressed
Image: Unused
Cursor: None

Mouse not over and button pressed
Image: Unused
Cursor: None

Mouse over and button not pressed
Image: Unused
Cursor: None

Mouse not over and button not pressed
Image: Unused
Cursor: None

Variable name containing info string:

9. Select the appropriate button image for each Mouse over/pressed state:

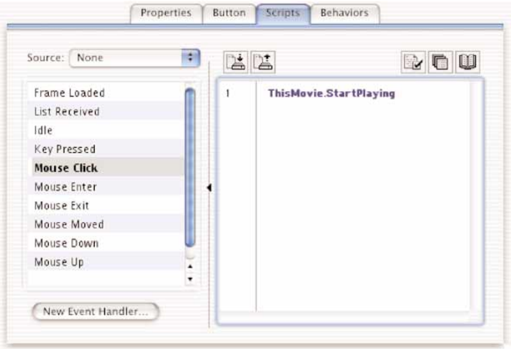
MOUSE STATE	BUTTON IMAGE	CURSOR
Mouse over and button pressed	down	Closed Hand
Mouse over and button not pressed	over	Pointing Hand
Mouse not over and button pressed	up	None
Mouse not over and button not pressed	up	None

10. Click on the Scripts tab

11. Select the Mouse Click Event Handler.

12. In the Script Editor Window, enter the following QScript:

    ThisMovie.StartPlaying

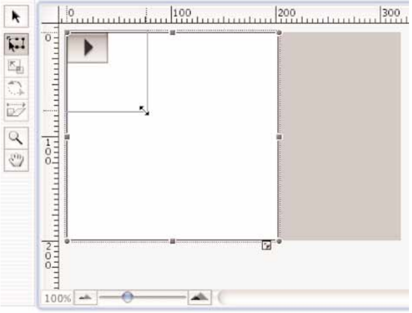


```
ThisMovie.StartPlaying
```

Close the Sprite Sample Properties window.

13. On the Stage, click on the tool.

    Click on the bottom right handle and resize the Sprite Track to fit the button.



14. From the Global tab of the Library window, open the Video folder.

15. Drag ‘bot.mov’ to the Stage.

    Position the video on the Stage, making sure not to cover the button.

16. Click the to preview your movie.

    Click your button to start the movie.

ADVANCED USE OF THE SPRITE TRACK

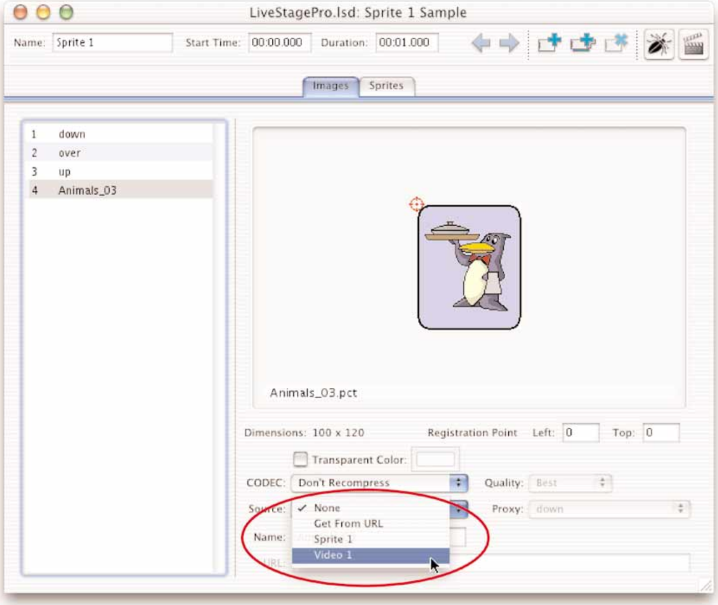
Image Override

Generally, sprites display their own images. You can, however, achieve interesting effects by replacing the image with visual data from another source. This is called Image Override. The Source popup menu in the Images tab of the Sprite Sample Properties window allows you to specify an alternate track or even a URL to replace the image of a sprite. For instance, you could turn a video into a clickable button, or make a slideshow draggable.

1. Open the Images tab of the Sprite Sample Properties window.

2. Select the image in the Image List that you would like to replace.

3. In the Source popup menu, select the track that contains the data you would like the image to display.



LiveStagePro.lsd: Sprite 1 Sample

Name: Sprite 1 Start Time: 00:00.000 Duration: 00:01.000

1	down
2	over
3	up
4	Animals_03

Animals_03.pct

Dimensions: 100 x 120 Registration Point Left: 0 Top: 0

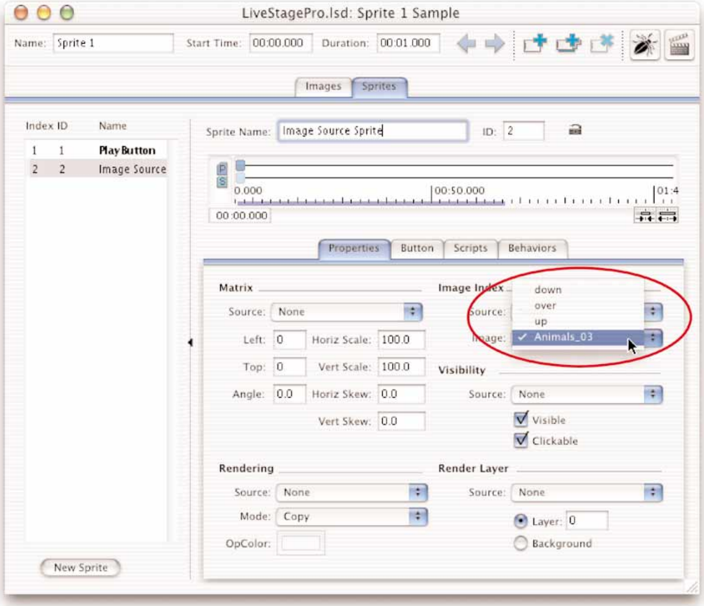
□ Transparent Color:

CODEC: Don't Recompress Quality: Best

Source: Video 1 Proxy: down

4. Click on the Sprites tab and create a sprite.

5. From the Image popup in the Image Index section, select the image you have just altered.



LiveStagePro.lsd: Sprite 1 Sample

Name: Sprite 1 Start Time: 00:00.000 Duration: 00:01.000

Index	ID	Name
1	1	Play Button
2	2	Image Source

Sprite Name: Image Source Sprite ID: 2

Matrix
Source: None
Left: 0 Horiz Scale: 100.0
Top: 0 Vert Scale: 100.0
Angle: 0.0 Horiz Skew: 0.0
Vert Skew: 0.0

Image Index
Source:
Image: ✓ Animals_03

Visibility
Source: None
☑ Visible
☑ Clickable

Rendering
Source: None
Mode: Copy
OpColor:

Render Layer
Source: None
● Layer: 0
○ Background

6. Add scripts and/or Behaviors to the sprite (see Chapter 16—Introduction to QScript, page 309).

You can also update an Internet URL with a new image every day so your sprite will display whatever image is at that URL on that particular day.

1. Open the Images tab of the Sprite Sample Properties window.

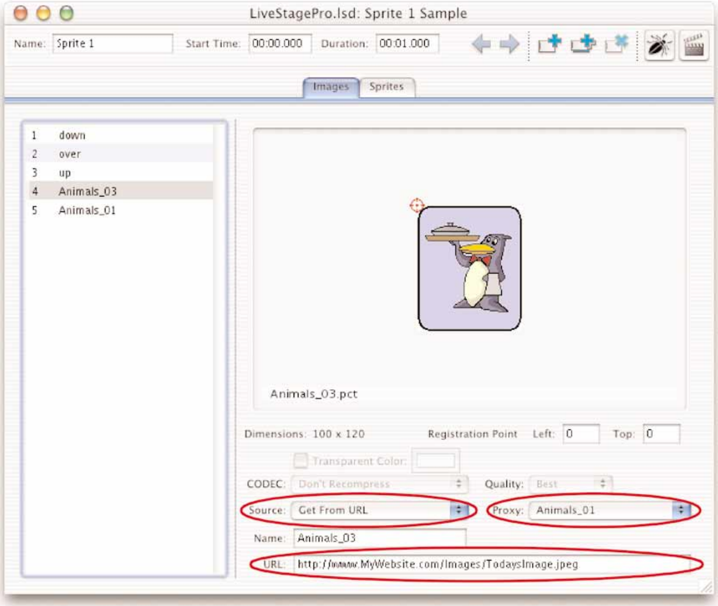
2. Select ‘Get From URL’ from the Source popup menu.

3. In the URL field at the bottom of the Images tab, enter the URL that contains the data you would like the sprite to display.

   Example: `http://www.MyWebsite.com/Images/TodaysImage.jpeg`

4. Complete Steps 4 through 6 above.

5. From the Proxy popup menu, select an image that will appear while the URL content is loading.



LiveStagePro.lsd: Sprite 1 Sample

Name: Sprite 1 Start Time: 00:00.000 Duration: 00:01.000

Images Sprites

1 down
2 over
3 up
4 Animals_03
5 Animals_01

Animals_03.pct

Dimensions: 100 x 120 Registration Point Left: 0 Top: 0

Transparent Color:

CODEC: Don't Recompress Quality: Best

Source: Get From URL Proxy: Animals_01

Name: Animals_03

URL: http://www.MyWebsite.com/Images/TodaysImage.jpeg

tip

LiveStage Professional comes complete with pre-made button sets located in the Global tab of the Library. You can drag these directly to your Sprite Track to create fully scripted buttons with up, down and over states.

SPRITE TRACK WRAP UP

The Sprite Track is sure to become the cornerstone of your interactive LiveStage Professional projects. You will rely on it to be everything from a simple button to the entire 'brain' of your QuickTime movie. This chapter just scratches the surface of all that can be accomplished with the Sprite Track. Chapter 16—Introduction to QScript and Chapter 17 - Behaviors will provide you with some more tools to assist you in taking advantage of the Sprite Track. Furthermore, in the Project Samples folder on the LiveStage Professional 4.0 CD-ROM, you will find a folder called "Sprite". This folder contains several QuickTime movies and the associated LiveStage Professional projects that highlight some of the ways the Sprite Track can be used in real-world situations. You will also find a wealth of information and inspiration on StageDoor (http://stagedoor.totallyhip.com).

Chapter 8

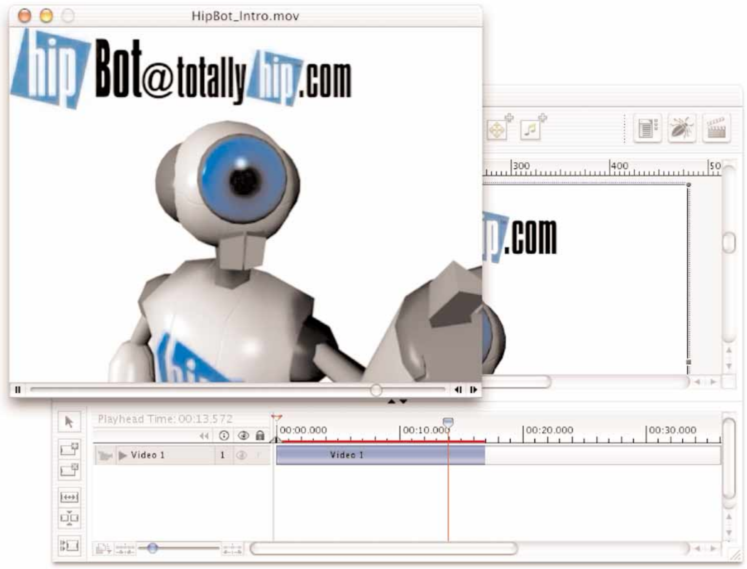
External Tracks

EXTERNAL TRACK OVERVIEW

External Tracks such as Video and Sound are the cornerstone of most LiveStage Professional projects. While they are often an integral part of LiveStage Professional projects, Video and Sound Tracks can be modified, but the media is not editable and the tracks cannot contain scripts. The External Track family also includes the Zoomify Track. Zoomify is a third party application that enables viewing of high resolution images over the Internet.

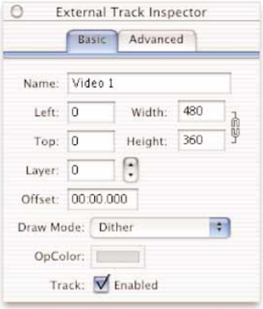
WORKING WITH EXTERNAL TRACKS

External Tracks can only be added to your LiveStage Professional project by dragging them to the Stage or Timeline from the Library. To preview an External Track, double click on the sample in the Timeline or on the Stage.



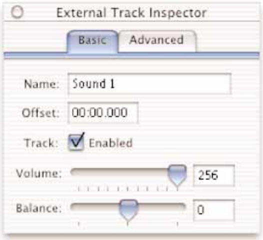
External Track Inspector/Properties Window

External Tracks have standard properties that are covered in detail in Chapter 3—Working in the Livestage Professional Environment, page 106. The Inspector and Track Properties Window differentiate between the Video, Zoomify, and the audible Sound/MIDI Tracks and provide the relevant options for each.


External Track Inspector

Basic | Advanced

Name: Video 1
Left: 0 Width: 480
Top: 0 Height: 360
Layer: 0
Offset: 00:00.000
Draw Mode: Dither
OpColor:
Track: ☑ Enabled


External Track Inspector

Basic | Advanced

Name: Sound 1
Offset: 00:00.000
Track: ☑ Enabled
Volume: 256
Balance: 0

ADVANCED USE OF EXTERNAL TRACKS

While External Tracks cannot be edited *per se*, they can be modified in a limited fashion through new tools common to Non-Linear Editing applications.

Set In & Out Points

In and out points can be thought of as the start and end points of the track. Setting them is the same as trimming some media from both ends of the track. You can assign the in and out points with precise timing in your External Video or Audio Track's timeline.

tip

You can also access the In & Out Point Editor through a contextual menu in the Timeline. Control-click on the Video track to open the menu, and select "set in your points" to open the Editor.

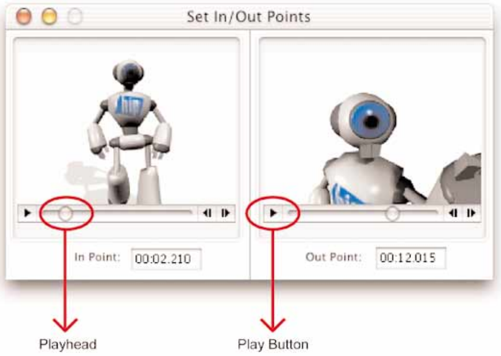
1. Select your Video Track.

2. From the Movie menu in the Main Menu Bar, select "Set In & Out Points..."

   The In & Out Point Editor opens.

3. Select the starting point for your Video Track by doing one of the following:

   - Click the Play button on the left window. Play the video until it reaches the point at which you would like the to set the In Point then hit the Pause button.
   - Drag the Playhead in the left window until it reaches the point at which you would like to set the In Point.
   - Enter the precise time code in the In Point field in [Minutes:Seconds.Ticks]. Ticks are 1/600th of a second. In other words, .600 is equal to one second; .300 would be half a second.



In Point: 00:02.210

Out Point: 00:12.015

4. Select the ending point for your Video by following the same steps in the right window.

5. Close the In & Out Point Editor.

   The change in the Video Track's duration is reflected in the Timeline.

6. Click the 🎬 to view the effects.

Fade Tools

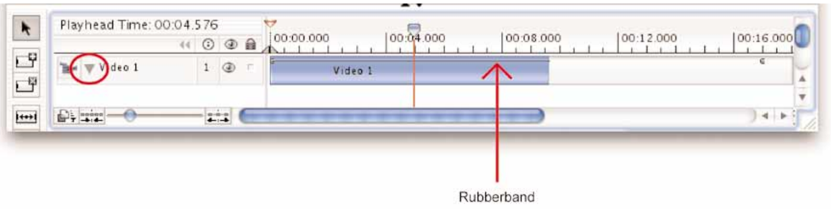
It is now easy to fade your Audio and Video Tracks in and out using the controls in the Timeline.

You may need to use the Zoom Slider at the bottom left of the Timeline to view your tracks in their entirety.

1. Select your Video Track in the Timeline.

2. Click the expansion arrow in the Track Header directly to the left of the Track Name.

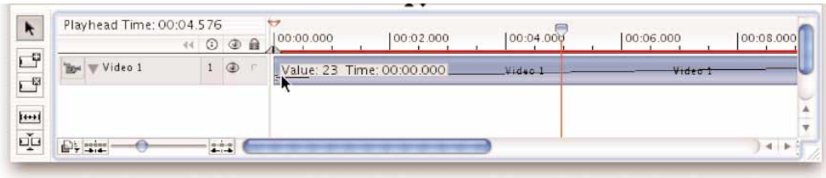
   The Video Track expands slightly to reveal the opacity ‘rubber band’; a thin line along the top of the Sample Bar, with small handles at either end of the track.



Playhead Time: 00:04.576

3. Click and drag the handle at the beginning of the Video sample to the bottom of the Sample Bar.

   A Tool Tip displays an opacity value ranging from 0 - 100 that changes as you drag vertically in the sample. An opacity of zero is invisible, while an opacity of 100 is fully visible. The current point in time is also shown.



Playhead Time: 00:04.576

Value: 23 Time: 00:00.000

   The opacity will gradually increase throughout the entire duration of the Video Track from the value you set.

Once you apply a fade to a Video or Audio Track, you will not be able to modify the Video Tracks drawing mode or the Audio Tacks volume using other methods, including via QScript.

4. To speed the fade in, click somewhere along the rubber band.

   A new handle appears.

5. Drag the new handle to the top of the Video Sample Bar at the point in the Timeline that you want the Video to be fully opaque.

   You can drag both vertically and horizontally within the Sample Bar. Handles will snap to the position of the Playhead. The Tool Tip will provide time and opacity values to help you precisely place your points. You can also view the effects of your adjustments on the Stage by dragging the Playhead along the Timeline.

6. Place as many handles within the rubber band as you like.

   To delete a handle, click on it and hit the Delete key, or simply drag the handle down or up and ‘out’ of the track.

   You can also adjust the volume of an Audio Track in precisely the same way. The volume value ranges from 0 (mute) to 100 (maximum).

Markers

Markers provide a visual cue that you can use to align tracks and samples in the Timeline, or to provide a label or flag to highlight important points in a track. This is also a great tool for creating quick Chapter Tracks (see Chapter 6—The Text Track, page 150). You can easily add a marker to an External Track:

CREATING A MARKER

1. Click on the External Track.
2. Set the desired marker position by moving the Playhead in the Timeline.
3. From the Movie Menu, select "Add Marker" (Option-Command-M).

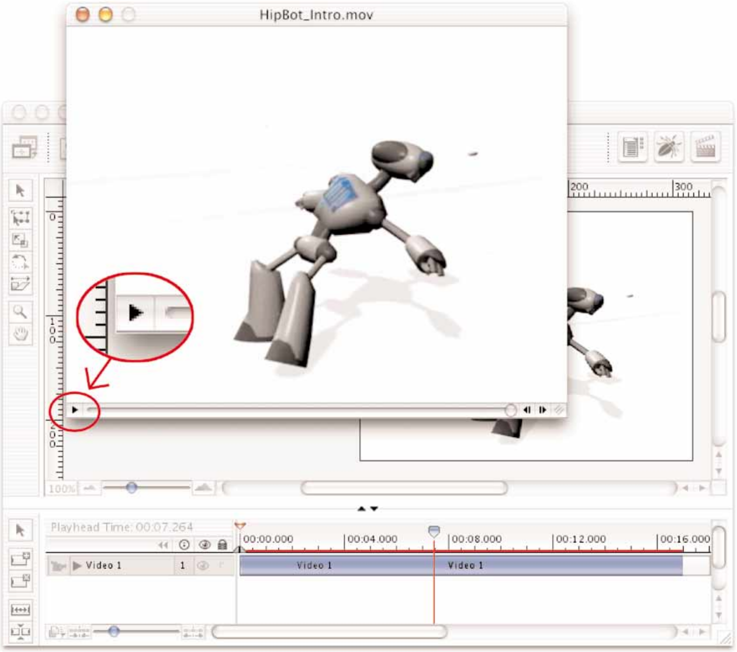
   A marker appears in the track.
4. To label a marker, click within it on the Timeline and edit its name.

You can also create a marker through the Contextual Menu control-click in the Sample Bar and select "Create Marker."

Creating a Marker on the Fly

1. Double-click on a Video or Audio sample (make sure the fade tools are hidden first).

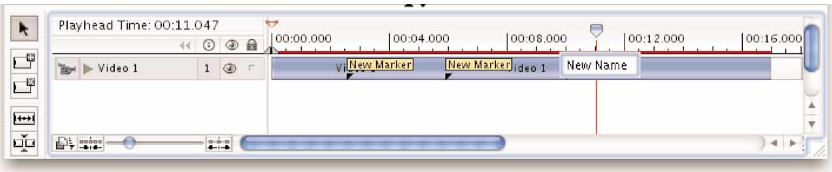
   This opens a preview of the media within the track.
2. Hit the play button on the controller.



3. As the Video or Audio track progresses, hit the letter M at the points where you would like a marker to be placed.

   Close the preview window when done.

4. The markers appear on the Video or Audio Track in the Timeline.

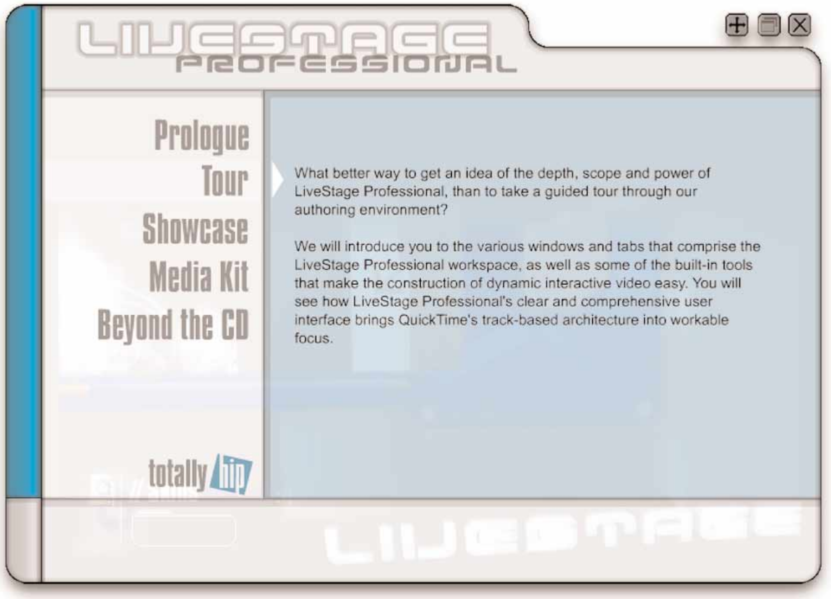


Chapter 9

The Flash Track

FLASH TRACK OVERVIEW

Macromedia Flash is a powerful media type for QuickTime development. Flash can be introduced to QuickTime as a scalable vector media format that incorporates its own interactivity at a minimal file size. Moreover, in LiveStage Professional 4.0, Flash developers can access Flash buttons and script them to control QuickTime functions, and playback. Through QuickTime, Flash developers can expand their expertise into QTVR, MPEG, real-time streaming, and many other areas not previously available through Flash alone.



What better way to get an idea of the depth, scope and power of LiveStage Professional, than to take a guided tour through our authoring environment?

We will introduce you to the various windows and tabs that comprise the LiveStage Professional workspace, as well as some of the built-in tools that make the construction of dynamic interactive video easy. You will see how LiveStage Professional's clear and comprehensive user interface brings QuickTime's track-based architecture into workable focus.

QuickTime 5 supports Flash 4 and QuickTime 6 supports Flash 5.

WHEN TO USE THE FLASH TRACK

Flash Tracks are used to add sophisticated vector art or animations that can interact with other media in your QuickTime movie, often as controllers, or other interface elements.

Use a Flash Track when you:

- Want scalable vector animated buttons and controllers and smooth tweening animations (i.e. menus)
- Have a Flash interface into which you want to incorporate high quality video.
- Have a Flash presentation into which you want to incorporate multiple media formats, while still maintaining their original format integrity (GIFs staying as GIFs, PNGs staying as PNGs, etc.).
- Are using text as a graphic element and you want to ensure the font looks identical over different platforms.

WORKING WITH THE FLASH TRACK

You can create a Flash Track in one of three ways:

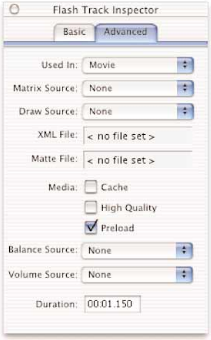
- Drag a .swf file to either the Stage or Timeline.
- Select Movie>CreateTrack>Flash... from the Menu Bar and navigate to the location of the .swf file on your computer.
- Control-click in the Track List area to access the contextual menu, select Create Flash Track, and navigate to the location of the .swf file on your computer.

The Flash Track will appear on the Stage and in the Timeline. In the case of a Fixed Stage, if the Stage is smaller than the dimensions of the Flash media, the Flash Track will resize to fit, and will not maintain the aspect ratio of the original file.

The Flash Track will have the same duration as the original Flash file. You can alter the Flash Track's duration with the Track Manipulation Tool (see Chapter 3—Working in the LiveStage Professional Environment, page 88). Warning: this will adjust the Flash file's frame rate accordingly.

tip

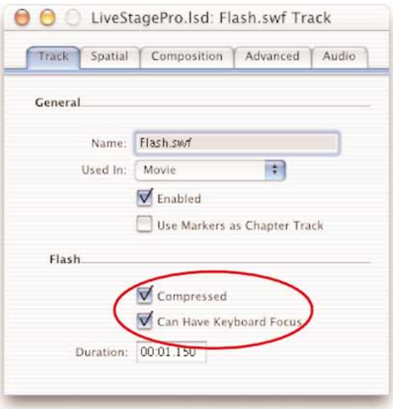
ALWAYS preload your Flash Track. This will ensure that the Flash Track has completely loaded before the QuickTime movie begins to play. Check the Preload option in the Flash Track Inspector or Track Properties window.


Flash Track Inspector

Used In: Movie
Matrix Source: None
Draw Source: None
XML File: < no file set >
Matte File: < no file set >
Media: □ Cache
□ High Quality
☑ Preload
Balance Source: None
Volume Source: None
Duration: 00:01.150

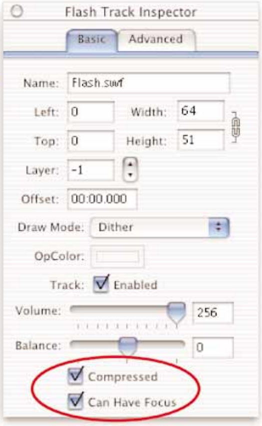
Flash Track Inspector/Properties Window

The Flash Track Properties window and the Flash Track Inspector have the same basic fields outlined in Chapter 3—Working in the LiveStage Professional Environment, on page 106, including Volume and Balance controls. You will also find three additional options:


LiveStagePro.lsd: Flash.swf Track

General
Name: Flash.swf
Used In: Movie
☑ Enabled
□ Use Markers as Chapter Track

Flash
☑ Compressed
☑ Can Have Keyboard Focus
Duration: 00:01.150


Flash Track Inspector

Name: Flash.swf
Left: 0
Width: 64
Top: 0
Height: 51
Layer: -1
Offset: 00:00.000
Draw Mode: Dither
OpColor:
Track: ☑ Enabled
Volume: 256
Balance: 0
☑ Compressed
☑ Can Have Focus

COMPRESSED

Checking this will compress the Flash Track. This option is found in the Track tab of the Flash Track Properties window and in the Basic tab in the Flash Track Inspector.

CAN HAVE (KEYBOARD) FOCUS

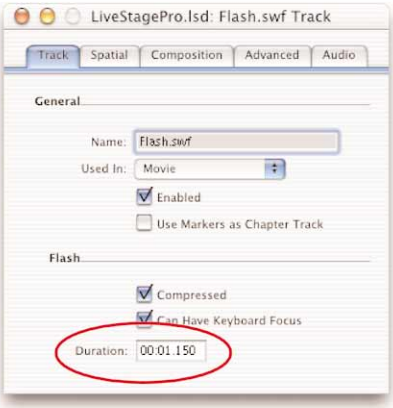
Selecting this option allows the Flash Track to receive keyboard input from the user. You will want this option selected if your Flash Track has a text-input field, for instance. Can Have (Keyboard) Focus is found in the Track tab of the Flash Track Properties window, and in the Basic tab of the Flash Track Inspector.

tip

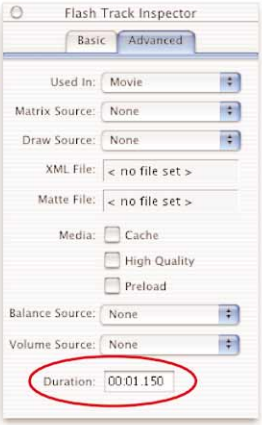
The background of a Flash file automatically exports from the Flash application with an Alpha Channel. This can be accessed by changing the Drawing Mode to “Alpha Channel” in the Flash Track Inspector or Track Properties window.

DURATION

You can enter a precise duration for the Flash Track in this field. The frame rate of your Flash Tracks will increase the more you shorten its duration. Conversely, the Flash Track’s frame rate will decrease as you extend its duration. This field is found in the Track tab of the Flash Track Properties window, and in the Advanced tab of the Flash Track Inspector.



Name: Flash.swf
Used In: Movie
☑ Enabled
□ Use Markers as Chapter Track
☑ Compressed
☑ Can Have Keyboard Focus
Duration: 00:01.150



Used In: Movie
Matrix Source: None
Draw Source: None
XML File: < no file set >
Matte File: < no file set >
Media: □ Cache
□ High Quality
□ Preload
Balance Source: None
Volume Source: None
Duration: 00:01.150

Flash Sample Properties Window

The Flash Track Sample Properties window is accessed by double-clicking on the Flash sample. The Flash Sample Properties window is where you can access the Buttons, Movie Clips, and individual frames within your .swf file and add QScripts to control QuickTime elements in your movie.

You will notice that a few of the standard tools available in most Sample Properties windows are missing. As Flash samples cannot be edited, the Create, Duplicate, and Delete Sample buttons are not present. The forward and back buttons, instead of moving you from sample to sample, move you from frame to frame within your Flash sample.

As with other Script Edit Windows, you can click and drag on the Resize Bar between the Event Handler List and the Script Edit Window to resize either pane. You can also click on the Resize Bar arrow to maximize the script area and then click again to restore the Event Handler list.

FRAMES PER SECOND

This field displays the current frame rate of your Flash sample. If you alter the duration of your Flash Track, the modified frame rate will be displayed here.

FRAME TIME

The Frame Time field displays the point where the selected frame is located in the Movie's timeline.

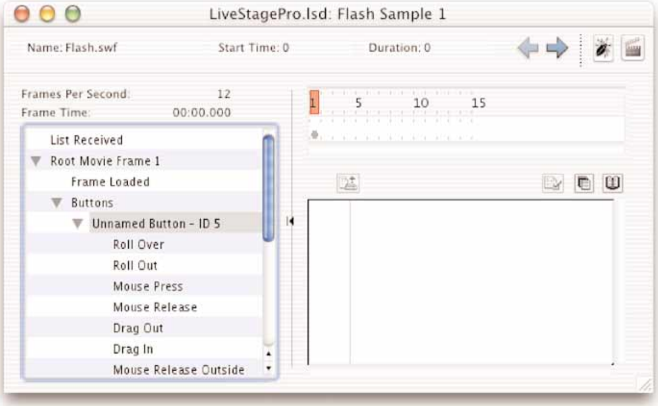
TIMELINE

The timeline along the top of the Flash Sample Properties window displays the Flash sample's timeline. The timeline is frame-based, similar to the Flash authoring environment. The frame that is currently selected is indicated by a red rectangular highlight. If a frame contains Buttons or Movie Clips, a gray dot will appear in the timeline. If Flash scripts are present, a black dot will appear on that particular frame in the timeline.

Scriptable buttons and Movie Clips can be anywhere along Flash's timeline. You will need to select the frame (with the gray or black dot) where these buttons are located to access their particular Event Handlers to add any desired QScript.

EVENT HANDLERS

Much like other scriptable tracks, the Object Event Handlers available for a particular Flash element will be listed in this pane. However, Event Handlers are presented differently in Flash samples than in other scriptable tracks. The first Event Handler in the list is the List Received handler, followed by the currently selected Frame in the Flash sample. Click on the triangle to the left of the Flash Frame name to view the elements contained within the frame such as Buttons, Movie Clips, or Event Handlers.



LiveStagePro.lsd: Flash Sample 1

Name: Flash.swf
Start Time: 0
Duration: 0

Frames Per Second: 12
Frame Time: 00:00.000

Flash buttons have different handlers than other scriptable objects in LiveStage Professional. Refer to Macromedia Flash documentation for details on what the different Flash button events are and how they work.

Use Flash Movie Clips for animated controllers, as you can change their duration without affecting the Flash Movie Clip's frame rate.

SCRIPT EDITOR

The Flash Sample Script Editor is identical to those in other scriptable tracks. You can Save and Load scripts, check syntax, and access the Defines and QScript Reference windows with the icons along the top.

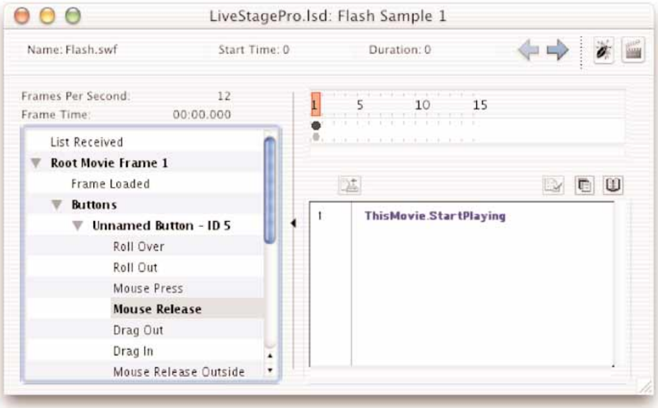
Scripting your Flash Track

You can add scripting to any frame, Movie Clip, or Button event in the Flash sample that has an Event Handler. While there are certain actions you can script to control the Flash element, more importantly, you can script the Flash element to control different objects in your movie. Flash Track QScripts are identical to scripts for other tracks with the exception of variables. The only variables that can be used in a Flash Track are Movie Variables. GlobalVars, LocalVars, and SpriteVars can not be used. See Chapter 16—Introduction to QScript to learn more about variables and QScript in general.

The procedure for scripting an Event Handler in the Flash Track is the same as other tracks.

- Select the Handler that you want to assign a script.
- Enter the QScript in the Script Edit Window to the right.

The Handler name will be displayed in boldface, indicating that a script is present.



LiveStagePro.lsd: Flash Sample 1

Name: Flash.swf
Start Time: 0
Duration: 0

Frames Per Second: 12
Frame Time: 00:00.000

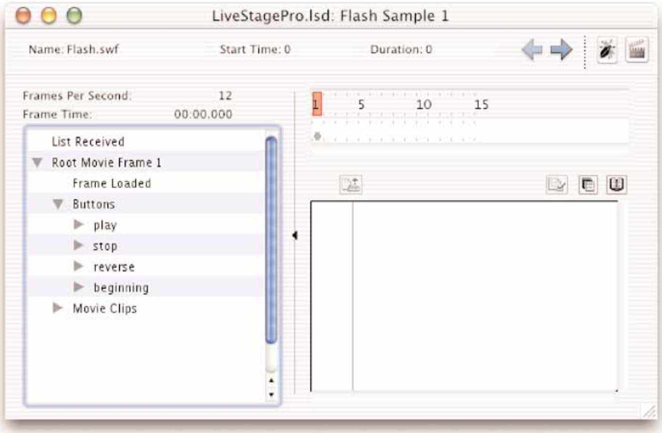
```
ThisMovie.StartPlaying
```

You can also script an individual frame by selecting it in the Timeline, selecting the frame loaded event, and then entering a script in the Script Editor.

ADVANCED USE OF THE FLASH TRACK

Labeling Flash Buttons

While you can double-click on Flash Buttons on the Stage to directly access their Event Handlers and add scripts, going back and forth between the two windows can get tedious if you have a lot of buttons to script. It's more elegant and efficient to have the names of the buttons display in the Event Handler list. This is actually accomplished through one of a number of FSCommands you can use in the Flash authoring tool from Macromedia.



LiveStagePro.lsd: Flash Sample 1

Name: Flash.swf
Start Time: 0
Duration: 0

Frames Per Second: 12
Frame Time: 00:00.000

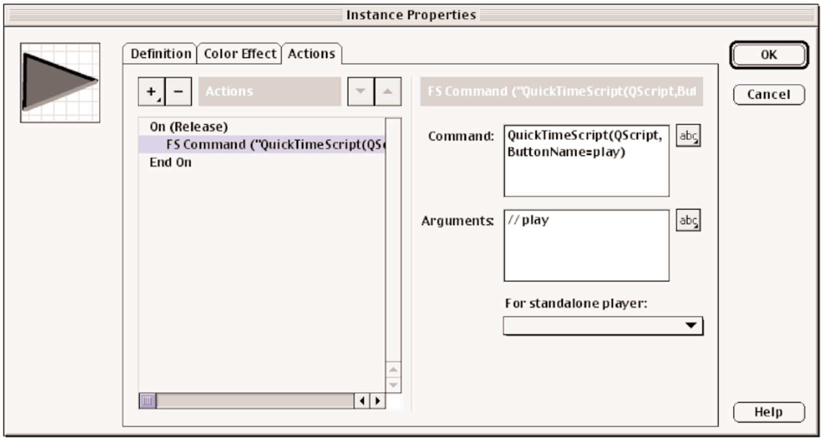
THE FSCOMMAND

The FSCommand in Flash is designed to communicate with other applications, keeping the string entered in the FSCommand intact within the .swf file. LiveStage Professional can access and display strings in your Flash file that use the FSCommand "QuickTimeScript". This allows you to add strings that LiveStage Professional understands. The most common use of FSCommands with LiveStage Professional is for adding comments to your buttons and Movie Clips. These can be viewed in the Script Edit Window once the file is in LiveStage Professional. It also allows you to specifically label the individual buttons in your Flash file that will then show up in the Event Handler list. While you can technically add QScript in Flash via the FSCommand, it is easier and more efficient to do it in LiveStage.

1. Create a Button in Flash.

2. Attach this FSCommand:

   QuickTimeScript(ButtonName=Play)



FS Command ("QuickTimeScript(QScript,But

```
On (Release)
  FS Command ("QuickTimeScript(QS
End On
```

Command: QuickTimeScript(QScript, ButtonName=play)

Arguments: //play

For standalone player:

tip

You can override an FSCommand already present in the Flash file simply by entering a QScript on that particular Event Handler. While you cannot delete a pre-existing FSCommand, you can override it by entering a blank space in the Event Handler's Script Editor.

3. Export your Flash .swf.

   When you bring the .swf into LiveStage Professional, the button will appear with the name "Play." Once your buttons have been labeled in this manner, if you make modifications to your original Flash file, any LiveStage Professional scripting you have added to the buttons will still apply.

Flash Track QScript Actions and Properties

Not only can your Flash Track control elements within your movie, but other scriptable elements in your movie can control your Flash Track as well. Here are some of the QScript Actions and Properties that can be used on the Flash Track (see Chapter 16—Introduction to QScript for scripting instructions):

GOTOFRAMENAMED

This moves the Playhead to the time of the specific frame that you have labeled on your Flash project's main timeline.

GoToFrameNamed("FrameName")

Example: `TrackNamed("Flash 1").GoToFrameNamed("Intro")`

GOTOFRAMENUMBER

If you prefer not to label your frame, this command will move the Playhead to the specific frame number.

GoToFrameNumber(FrameNumber)

Example: `TrackNamed(“Flash 1”).GoToFrameNumber(10)`

Flash frames are numbered starting with zero. Therefore, this example sends the Playhead to the 11th frame in the Flash Track.

SETZOOM

This zooms the Flash Track in and out at specific percentages. 100 is normal size.

SetZoom(Percentage)

Example: `TrackNamed(“Flash 1”).SetZoom(60)`

This sets the viewable area of the Flash Track to 60% of its normal size.

SETPAN

This pans the Flash Track by the specified X and Y percentages. Note that the Flash Track needs to be zoomed out in order to be able to pan.

SetPan(Xpercentage,Ypercentage)

Example: `TrackNamed(“Flash 1”).SetPan(25, 45)`

SETFLASHVARIABLE

One of the more powerful capabilities in the Flash/QuickTime relationship is the ability for each to pass variables to the other. This Action sets a value in the Flash Track variable that is specified. Flash can test for and then perform action scripts depending on the variable’s value. In this way, a QuickTime track could control most or all of your Flash Tracks actions.

SetFlashVariable(“Path”,”VariableName”,”Value”,”Focus”)

Example:

```
TrackNamed(“Flash 1”)
SetFlashVariable(“”,”myVariable”,”25”,”FALSE”)
```

This sets the Flash variable “myVariable” to “25” and does not alter focus.

GETFLASHVARIABLE

This retrieves the value from the specified variable in the Flash Track.

GetFlashVariable(“Path”,”Name”)

Example:

```
TrackNamed(“Flash 1”).
GetFlashVariable(“”,”myFlashVariable”)
```

FLASH TRACK WRAP UP

QuickTime and LiveStage Professional help expand the experiences that Flash developers create. You will find a number of pre-built Flash elements in the Flash folder of the Global Library that you can incorporate into your projects. Additionally, there is an entire section devoted to Flash on StageDoor (http://stagedoor.totallyhip.com), where you can take in Flash-specific tips, tricks, and tutorials.

Chapter 10

VR Track

VR TRACK OVERVIEW

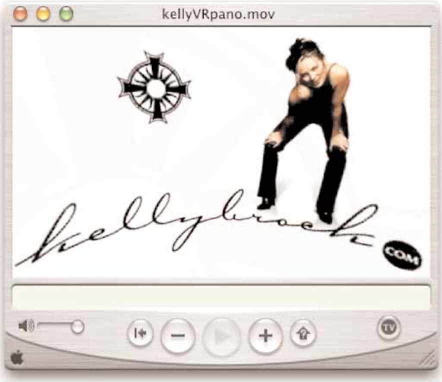
QuickTime VR takes interactivity into a new dimension—quite literally. With QuickTime VRs, a user can explore three-dimensional objects and panoramic scenes, real, or imagined. Whether the subject is a distant locale or a computer-generated 3D rendering, QTVR's provide a unique and captivating user experience. Because the VR Track is a scriptable track, you can enhance and extend the already impressive VR technology with added interactivity.

QTVR Nodes

A QTVR is a series of images that are 'stitched' together in an application such as QuickTime VR Authoring Studio or VR ToolWorks to create individual 'nodes'. A node can be one of four different types of VR:

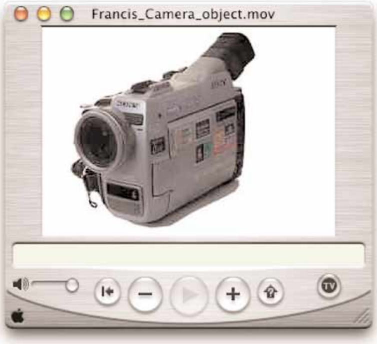
PANORAMA

A panorama is a 360° image that can be an actual location or an imaginary environment created in any number of graphics or 3D rendering programs. The user's point of view is from the center, looking outward. By using the mouse or the keyboard, the user can view the surroundings as if they were standing in the center of the environment and turning around.



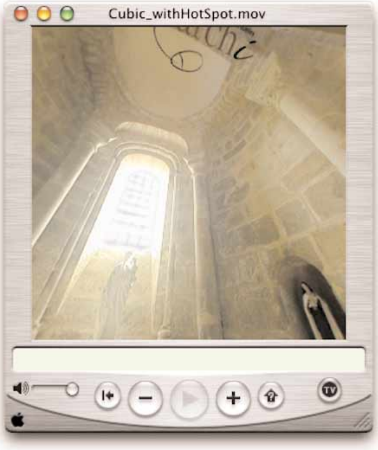
OBJECT VR

Object VRs allow the user to view and manipulate a three-dimensional object, either real or the product of a 3D graphics program. With the mouse or keyboard, the user can view the object from all sides as if they were picking it up and turning it around.



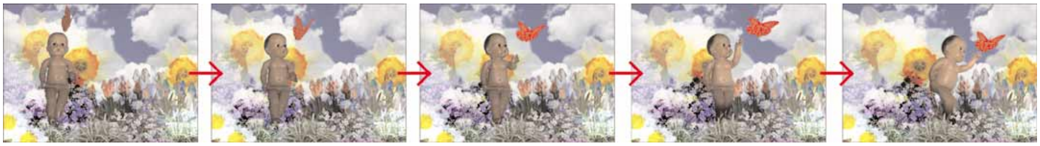
CUBIC VR

Think of a Cubic VR as a Panorama with a floor and ceiling. Not only can the user view the landscape, but they can also view the sky and ground as well.



ABSOLUTE VR

The Absolute VR is less common than its immersive counterparts. The Absolute VR is much like a grid with different images in each square. Click on any square of that grid and a new image will appear. It can be constructed to appear as though an object is jumping or reacting to whatever location is clicked.



Scenes

A QTVR scene is a construction of one or more VR nodes. These nodes can be made up of any combination of the four VR types. They can also link to both external media and Internet URLs. The user can progress through a scene by clicking on HotSpots that link one node to another.

HotSpots

QTVR Nodes usually contain HotSpots—clickable areas on the VR itself that the user can interact with to perform actions such as going to another VR or a URL. These HotSpots are also created in the QTVR authoring application, but can be scripted with QScript once the VR is brought into LiveStage Professional.

WHEN TO USE THE VR TRACK

QTVR's excel in bringing the inaccessible to the user. From their computers, your audience can visit distant or imagined places and view objects and products they might not otherwise be able to examine.

Bring a VR Track into LiveStage Professional 4.0 when you want to:

- Add a custom controller, compass, or other interactive navigational aides to a QTVR panorama.
- Combine video, sound, and graphic elements with your cubic, panorama, or object VR.
- Display additional information when the user accesses the HotSpots on your VR.
- Create a rich virtual world.

WORKING WITH THE VR TRACK

VR Tracks have a couple of qualities that differentiate them from other LiveStage Professional tracks. The most obvious difference is that VR Tracks are not time-based. While they appear in the Timeline with a duration that has been specified in the original VR movie, they are not affected by playback time. There are two other anomalies for you to be aware of as you work with the VR Track:

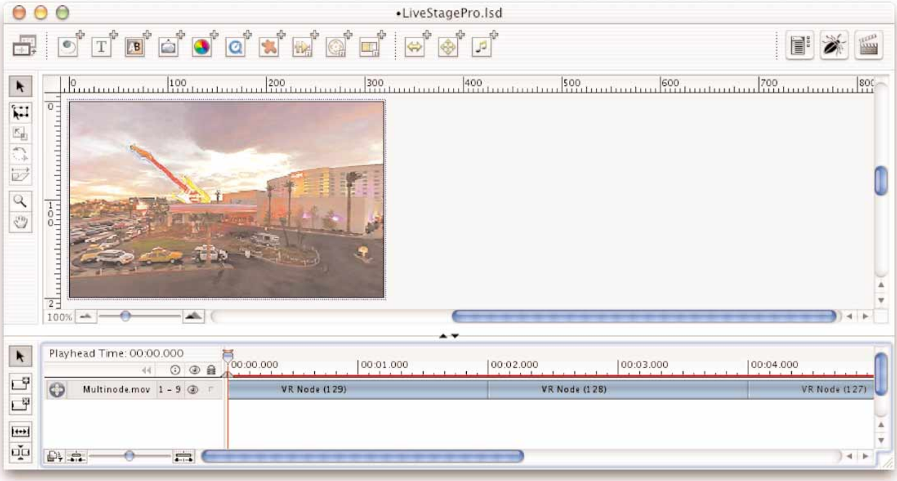
INDEX NUMBERS

The Index number for a VR Track is actually a range of numbers (i.e. 1-5). This is because a VR Track is actually made up of several tracks. LiveStage Professional displays the VR as a single track to make authoring easier and to ensure the VR will function properly, so take special care in referencing index numbers when incorporating a QTVR into your LiveStage Professional project.

QTVR NODES

VR Tracks are made up of Nodes, not samples. A multi-node QTVR scene will appear in the Timeline as separate nodes, looking much like samples in other tracks. You can name the individual Nodes. While other tracks default to a sample name of “Untitled”, VR samples are blank except for the prefix. When you name your VR sample, it will look something like this:

```
VR Node (127) MyVRSample
```



•LiveStagePro.lsd

Playhead Time: 00:00.000

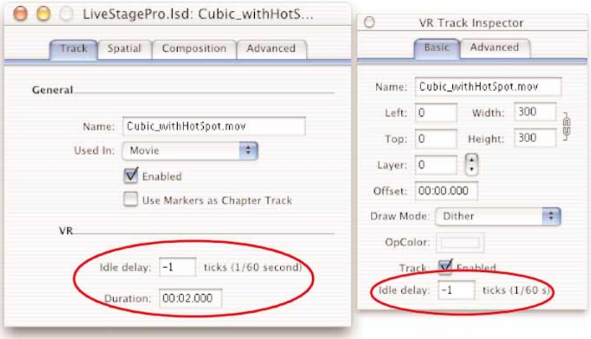
Multinode.mov 1 - 9

VR Track Inspector/Properties Window

The VR Track Properties window and the VR Track Inspector have the same basic fields outlined in Chapter 3—Working in the LiveStage Professional Environment, on page 106. You will also find these additional options:

IDLE DELAY

The VR Track, being a scriptable track, can receive Idle Events. The Idle Delay rate determines how often the VR Track will receive these events. The default value is 1, which is 1/60 of a second—or one 'tick'. You can set a value of zero, which is the fastest, or you can set a value of -1 to turn it off. The Idle Delay is found in the Track Properties window in the Track tab and in the VR Track Inspector in the Basic tab. For more information on Idle Delay, see Chapter 16—Introduction to QScript, page 312.



LiveStagePro.lsd: Cubic_withHotS...

Track | Spatial | Composition | Advanced

General

Name: Cubic_withHotSpot.mov

Used In: Movie

☑ Enabled

□ Use Markers as Chapter Track

VR

Idle delay: -1 ticks (1/60 second)

Duration: 00:02.000

VR Track Inspector

Basic | Advanced

Name: Cubic_withHotSpot.mov

Left: 0 Width: 300

Top: 0 Height: 300

Layer: 0

Offset: 00:00.000

Draw Mode: Dither

OpColor:

Track: ☑ Enabled

Idle delay: -1 ticks (1/60 s)

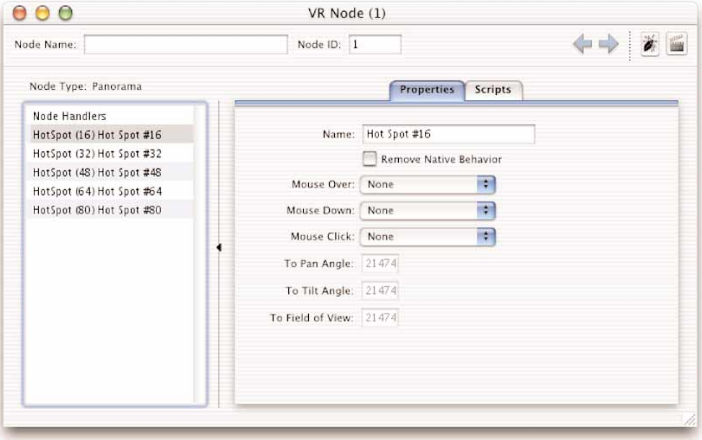
DURATION

The duration that appears in this field is set by the original VR movie. When combining your VR with a linear movie, you can alter the duration of a Panorama VR, but not that of an Object VR. Object movies will not work if their duration is changed, our panorama functionality will be unaffected by a change in duration. The Duration is found in the Track tab of the Track Properties window and in the Advanced tab of the VR Track Inspector. You can also use the

Track Manipulation tool to manually adjust the duration in the Timeline (see Chapter 3—Working in the LiveStage Professional Work Environment, page 88).

VR Sample Properties Window

The VR Track Sample Properties window can be accessed by double clicking on the VR node. Because you cannot edit samples (Nodes) in the VR Track, the standard sample tools found in most Sample Properties windows have been omitted. The forward and back buttons, instead of moving you from sample to sample, move you from node to node, provided your VR sample is a multi-node Scene. As with many other LiveStage Professional windows, you can click and drag on the Resize Bar between the HotSpot List and the tab section to resize either area.



Node Name:
Node ID: 1

Node Type: Panorama

Node Handlers
HotSpot (16) Hot Spot #16
HotSpot (32) Hot Spot #32
HotSpot (48) Hot Spot #48
HotSpot (64) Hot Spot #64
HotSpot (80) Hot Spot #80

Name: Hot Spot #16
□ Remove Native Behavior
Mouse Over: None
Mouse Down: None
Mouse Click: None
To Pan Angle: 21474
To Tilt Angle: 21474
To Field of View: 21474

NODE NAME/NODE ID

Each Node in your VR Track will have a Node ID assigned, and may or may not have a Node Name depending on the authoring tool used to create the VR. You can enter a name in this field that will then appear after the VR Node (ID#) title in the Timeline. The ID # is what you refer to when using the GoToNodeID QScript command. This node name will appear in the QuickTime player window when the user moves the mouse over the node.

NODE TYPE

Node Type indicates the type of the current VR node, such as a Panorama or Object.

HOTSPOT LIST

The HotSpot list displays all of the HotSpots contained in the VR node. At the top of the list, you will see an entry called Node Handlers. This contains information relevant to the VR Node as a whole. Select an individual HotSpot in this list to set its properties and add scripts through the Properties and Scripts tabs to the right.

PROPERTIES TAB

The Properties tab offers different settings for Nodes and their HotSpots. When ‘Node Handlers’ is selected in the list, you can attach a Tween Track (see Chapter13—Additional

LiveStage Professional Tracks, page 259) to control the Pan, Tilt, or Zoom of the whole Node. When you select a HotSpot, you can access a number of options in the Properties tab:

Name

You can name or rename the individual HotSpot in this field. This name will be displayed in the QuickTime Player when the user's mouse is over the HotSpot, and here in the Sample Properties window in the HotSpot List.

Remove Native Behavior

Checking this option will disable any pre-defined actions for the HotSpot added in the original construction of the VR. This allows you to have the HotSpot perform actions other than those specified in the VR authoring program. Leave this unchecked if you do not want to change the HotSpot's existing functionality, or if you want to augment its programming with your own QScripts.

Custom Cursors

Custom cursors can help your user detect the different HotSpots in your VR. You can choose from a variety of pre-made cursors like colored arrows or even a turtle, to be displayed when the user's mouse enters or clicks on a HotSpot. You can also make your own cursors in programs like "ResEdit."

Mouse Over The mouse is over the HotSpot but the mouse button is not pressed.

Mouse Down The mouse is over the HotSpot and the mouse button is being pressed.

Mouse Click The mouse is currently over the HotSpot and has been clicked.

SET NODE PROPERTIES

In LiveStage Professional 4.0, you can now set the Field of View, the Pan Angle, and the Tilt Angle for a destination Node. In other words, you can determine what your users will be looking at when they jump to the next node in your QTVR scene. Note that you can only set these properties for linked nodes, not for the first node. You can, however, alter the VR's properties in the original VR authoring application or in LiveStage Professional through QScript. See the QScript Reference Appendix for more information.

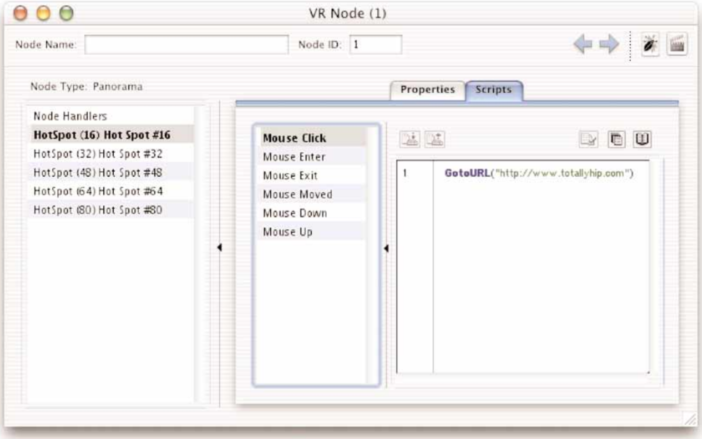
To Pan Angle: This determines the horizontal angle that will be the center point of what the users will see when they link to the node.

To Tilt Angle: This determines the vertical angle that will be the center point of what the users will see when they link to the node.

To Field of View: This will determine how far the node will be zoomed in when the user links to it.

SCRIPTS TAB

The VR sample Script Editor is identical to those in other scriptable tracks. You can Save and Load scripts, check syntax, or access the Defines and QScript Reference windows through the icons along the top. The Resize Bar allows you to hide the Event Handlers List, maximizing your scripting area.


VR Node (1)
Properties
Scripts
Node Handlers
HotSpot (16) Hot Spot #16
HotSpot (32) Hot Spot #32
HotSpot (48) Hot Spot #48
HotSpot (64) Hot Spot #64
HotSpot (80) Hot Spot #80
Mouse Click
Mouse Enter
Mouse Exit
Mouse Moved
Mouse Down
Mouse Up

Node Name:

Node ID: 1

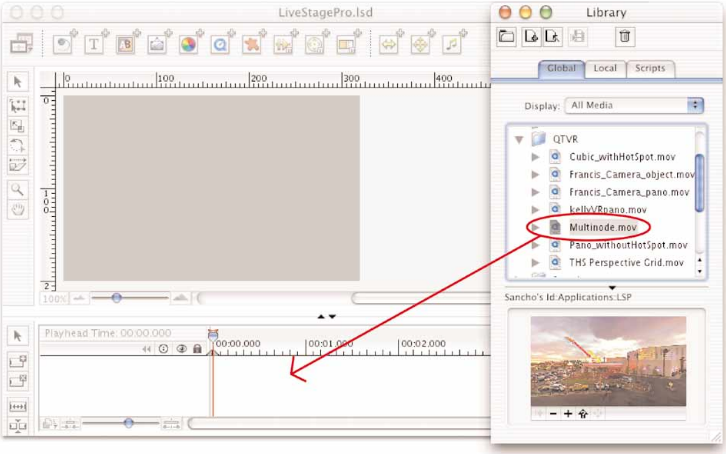
Node Type: Panorama

```
GotoURL("http://www.totallyhip.com")
```

Building a VR Track

The VR Track is created in one of three ways. You can either drag the QTVR .mov file directly to the Stage or Timeline, or you can select Movie>CreateTrack>VR from the Menu Bar and navigate to the location of the QTVR .mov file on your computer. Alternately, you can control-click in the Track List area to open a Contextual Menu to create a VR Track.

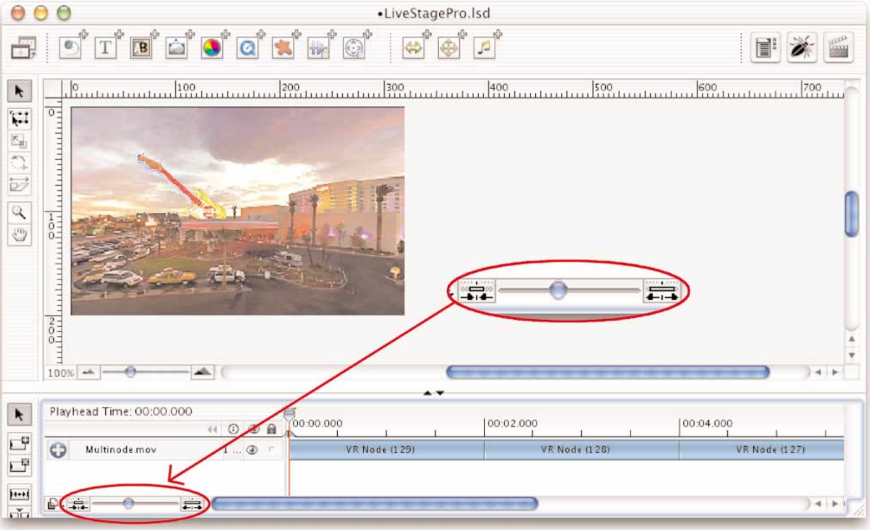
1. In the Global tab of the Library window, open the QTVR folder. Select Multinode.mov and drag this file into your project's Timeline.



Display: All Media

Playhead Time: 00:00.000

2. Using the Zoom controls at the bottom of the Timeline, zoom out until you can view the individual Node Names.



•LiveStagePro.lsd

Playhead Time: 00:00.000

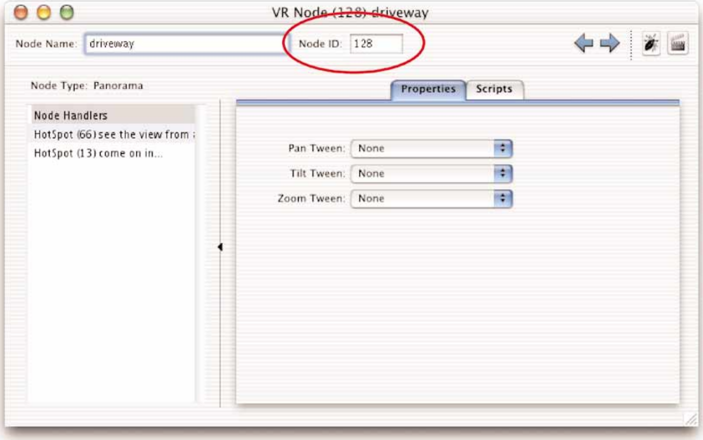
Multinode.mov

3. Double-click the second sample (Node) in the Timeline.

The VR sample Properties window opens.

4. Change the Node’s Name to ‘driveway’.

Take note of the ID# for this Node.



VR Node (128) driveway

Node Name: driveway

Node ID: 128

Node Type: Panorama

Node Handlers
HotSpot (66) see the view from :
HotSpot (13) come on in...

Pan Tween: None

Tilt Tween: None

Zoom Tween: None

Controlling your VR Track

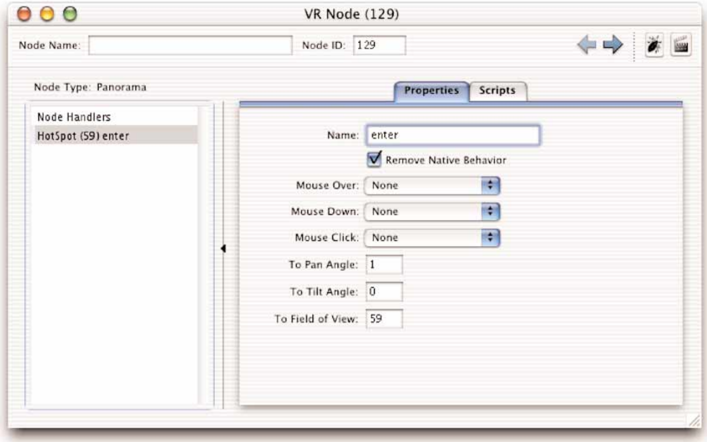
QScripts are added to individual VR HotSpots in exactly the same way as other scriptable tracks. Select the Node or HotSpot from the HotSpot List, choose the applicable Event Handler, and add a QScript in the Script Editor. If you select a Node Handler, the Event Handlers List will offer Frame Loaded and Idle Events to script. If you select a HotSpot, you can add QScripts to typical Mouse Events. For more information on Event Handlers, see

Chapter 16—Introduction to QScript, page 312. As with all scriptable tracks, when a script is attached to an Event Handler, the Handler displays in boldface.

6. Click the 'Previous' button in the VR sample Properties window.

7. Select HotSpot 59.

8. In the Properties tab, change the HotSpot's Name to 'enter'.

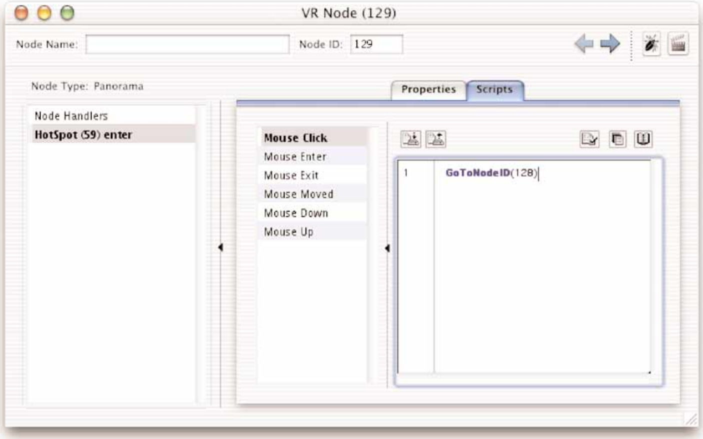


Node Name:
Node ID: 129
Node Type: Panorama

Name: enter
☑ Remove Native Behavior
Mouse Over: None
Mouse Down: None
Mouse Click: None
To Pan Angle: 1
To Tilt Angle: 0
To Field of View: 59

9. Click the Scripts tab and select Mouse Click from the Event Handlers List.

10. In the Script Editor field, enter the following QScript, specifying the ID# of the Node we visited in Step 3 in the parentheses:



Node Name:
Node ID: 129
Node Type: Panorama

```
GoToNodeID(128)
```

11. This Script moves the Playhead to the 'driveway' Node when the user clicks on this particular HotSpot.

12. Preview your movie and click on the slate image.

ADVANCED USE OF THE VR TRACK

- You can use scripting to integrate various media objects with one or more VR Tracks. However, there are two restrictions to keep in mind:
  - The only variable declaration you can use in VR Tracks is the Movie Variable (MovieVars). LocalVars, SpriteVars, and GlobalVars are not supported. See Chapter 16—Introduction to QScript for more information on Variables.
  - VR Tracks do not support Custom Event Handlers (Chapter16—Introduction to QScript, page 313).
- There are a number of Behaviors available that you can add to a Sprite Track to affect your VR Track. You can have your Panorama automatically Pan, Tilt, or Zoom.
- You can create a custom controller, either in a Sprite Track or through the VR Controls FastTrack™, to give the user an alternative to dragging the mouse over the VR.
- You can use a VR as a controller for other media, for instance using the values of Tilt, Pan, and Zoom to trigger actions on other tracks or movies.

tip

QTVR's used as Child Movies in Movie Track (See Chapter 11—The Movie Track, page 215), must be loaded twice in order to be viewed if the project is being deployed for QuickTime 5. This has been fixed in QuickTime 6.

VR TRACK WRAP UP

LiveStage Professional enhances already engaging QuickTime VR's, rounding out the interactive experience they offer. The VR Track has been used to great effect with projects like Greenpeace's Great Bear Rainforest site (http://www.greenpeace.org/greatbear/). On the LiveStage Professional 4.0 CD-ROM you will find a folder filled with different examples of panoramic, cubic, multi-node, and object VRs. You can also find several featured examples and tutorials on the VR Track on the StageDoor site at http://stagedoor.totallyhip.com.

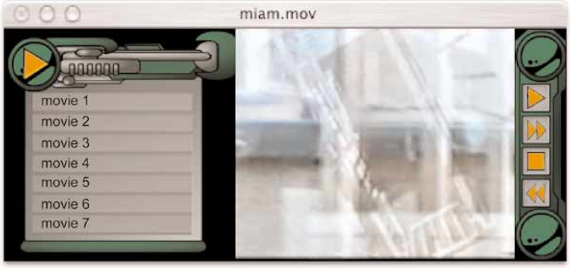
Chapter 11

The Movie Track

MOVIE TRACK OVERVIEW

The Movie Track, commonly known as Movie-in-a-Movie (MIAM), is a movie container. Movie Tracks contain separate movies that can be loaded and controlled independently of the main movie's timeline. For instance, your main movie can be paused while your Movie Track plays, or you can use the main movie to control the playback of media in your Movie Track.

Though it is called a Movie Track, the media MIAM contains are not limited solely to QuickTime movies. A Movie Track can contain any type of media that QuickTime recognizes. A Movie Track can also contain fully interactive movies with multiple tracks. Media can be embedded directly into the Movie Track or they can be linked via URLs. Linked movies are only loaded into memory when they are called for via QScript.



WHEN TO USE THE MOVIE TRACK

Movie-in-a-Movie is often used for creating media 'players' that offer up a collection of image, audio, and video files. It is also a very effective means of improving your workflow with similar projects. For example, you can create a template that uses MIAM to access media applicable to a variety of projects you might author.

Use a Movie Track when:

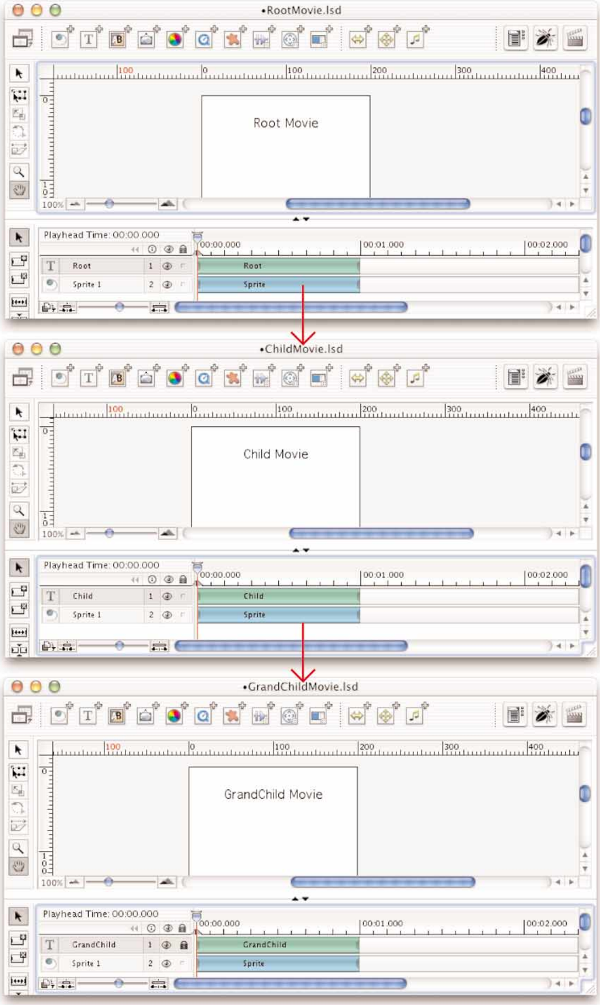
- You want to have continuous background music playing regardless of whether your main movie is paused, playing, rewinding etc.
- You want to present elements that have different timelines in a single movie and you want the user to be able to access these elements from any point in the movie.
- You want the user to be able to switch from one movie or media element to another in a single interface, instead of having a different window for each.
- You have a repository of varied media that you would like to keep in their native formats while allowing your user to access and control them through a single interface.
- You want your movie to present media elements that are downloaded to the user's machine only when requested.
- You have several different movies that access an external media element, and you want to be able to automatically update all the movies when the element is modified.

- Your movie involves media elements that are going to change periodically, and you want the movie to update automatically without having to recompile the original project.

WORKING WITH THE MOVIE TRACK

The Movie Track is created through the Menu Bar via Movie>Create Track>Movie (Command-Option-N), or by clicking on the in the Track Creation Toolbar. You can also control-click in the Track List area of the Timeline to open a contextual menu and then select Create Movie Track.

When you create a Movie Track, a hierarchy is set up within your project. Your main timeline becomes the “Root”, or “Parent” Movie and the Movie Track you have just created is considered its “Child”. It is possible for a movie that is loaded into the “Child” to have a Movie Track of its own. In this case, the movie in the Child movie would be considered a ‘grandchild’, or for targeting purposes, a Child of the Child. Conceivably, you could continue to nest movies *ad nauseam*. However, even for the most complex projects you would most likely need only two or three generations.



Example: Movie A = Parent (Root) Movie
Movie B = Child Movie of Movie A
Movie C = Child Movie of Movie B, 'grandchild' Movie of Movie A

Movie A

- Root Movie
- Parent of Movie B
- Can target ALL movies

Movie B

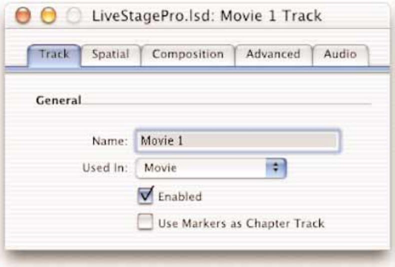
- Child of Movie A
- Parent of Movie C
- Targets Movie A, using "Parent" or "Root"
- Targets Movie C, using "Child"

Movie C

- Child of Movie B
- Targets Movie B, using "Parent"
- Targets Movie A, using "Root"

You will need to keep this construction clearly in mind when you start to script your MIAM, as you will see in the "MIAM Targeting" section later in this chapter.

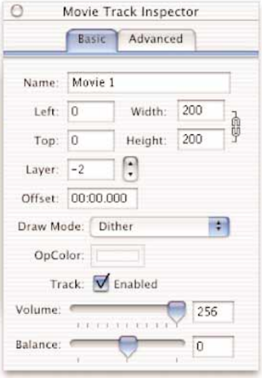
Movie Track Inspector/Properties Window



LiveStagePro.lsd: Movie 1 Track

General

Name: Movie 1
Used In: Movie
☑ Enabled
□ Use Markers as Chapter Track



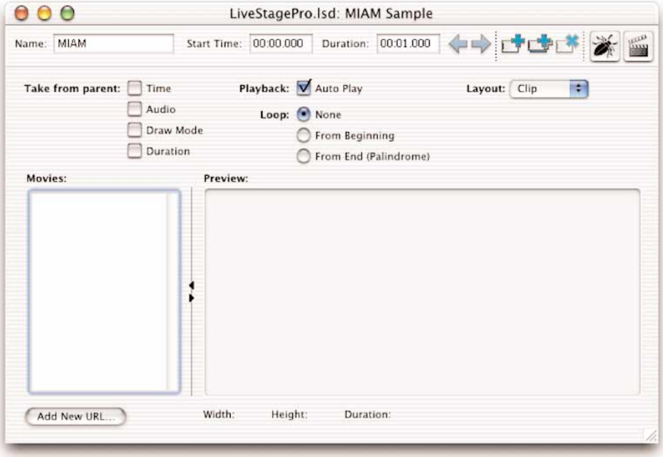
Movie Track Inspector

Name: Movie 1
Left: 0 Width: 200
Top: 0 Height: 200
Layer: -2
Offset: 00:00.000
Draw Mode: Dither
OpColor:
Track: ☑ Enabled
Volume: 256
Balance: 0

The Movie Track Properties window and the Movie Track Inspector have the same basic fields outlined in Chapter 3—Working in the LiveStage Professional Environment, on page 106. The one difference from most other tracks is the addition of Volume and Balance controls. You can set the volume and balance for your Movie Track in either the Track Properties window or the Inspector.

Movie Sample Properties Window

In the Movie Track's Sample Properties window you can add your Child Movies and determine how they will play.



LiveStagePro.lsd: MIAM Sample

Name: MIAM
Start Time: 00:00.000
Duration: 00:01.000

Take from parent: □ Time □ Audio □ Draw Mode □ Duration
Playback: ☑ Auto Play
Loop: ● None ○ From Beginning ○ From End (Palindrome)
Layout: Clip

Aside from the standard fields and buttons along the top of the window (explained in Chapter 3—Working in the LiveStage Professional Environment, page 88), the Movie Sample Properties window provides options that will be applied to all of the movies within your Movie Track. You will also notice a resize bar between the Movies and Preview sections that you can drag to resize either pane. Clicking on the arrows on this bar will maximize one pane or the other.

TAKE FROM PARENT

The options in this section determine how dependent the Child Movie's properties will be on its Parent. Selections from this list will impose the settings of the Parent Movie Track on all the Child Movies loaded into the movie track, though not to 'grandchild' movies.

Time

Select this option if you want the frame rate and playback of the Child Movie to be determined by the Parent. If the Parent Movie is stopped, the Child Movie will stop. If the Parent Movie is rewinding, the Child Movie will rewind as well.

Audio

If you select this option, the volume and balance settings of the Parent, set in the Inspector or the Movie Track Properties window will apply to the Child Movie as well.

Draw Mode

When you check this setting, the Child Movie will have the same Drawing Mode as that of the Parent Movie.

PLAYBACK: AUTO PLAY

When this is set, the Child Movies will automatically start playing when loaded. It is generally best to avoid this option as the Child Movie can begin playing before sufficient data has been loaded (see Troubleshooting at the end of this Chapter).

LOOP

Select from these three options to control how your Child Movies will loop during playback.

None

The Child Movie will play once and stop when it reaches the end. This is the default selection.

From Beginning

When this option is selected, the Child Movie will start again at the beginning once it reaches the end.

From End (Palindrome)

When the Child Movie reaches the end of its duration, it will play in reverse back to the beginning and the cycle will repeat. Streaming Movies will not palindrome.

LAYOUT

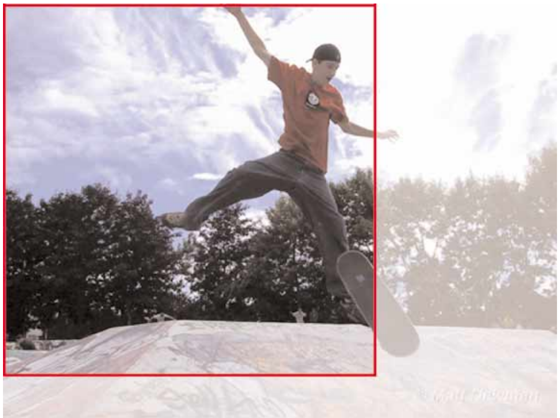
This pop-up menu determines how the Child Movies will visually fit into the Movie Track if the dimensions of the Movie Track do not match those of the Child.

None

The Child Movie will be scaled and its aspect ratio adjusted to match the dimensions of the Movie Track.

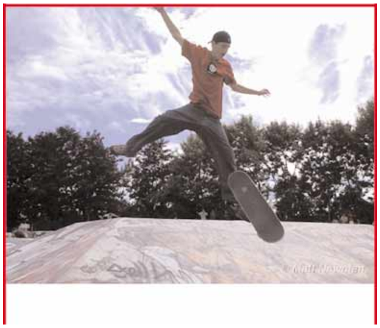
Clip

The Child Movie will not be resized and it will appear at the top left corner of the Movie Track. If it is larger than the Movie Track, then the areas that extend below and to the right of the Movie Track will not be visible.



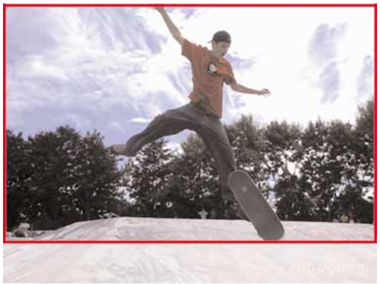
Meet

The Child Movie will be resized to fit within the Movie Track, while maintaining its aspect ratio. If the Movie Track's proportions differ from those of the Child Movie, there will be blank space to the bottom or the right of the Child Movie.



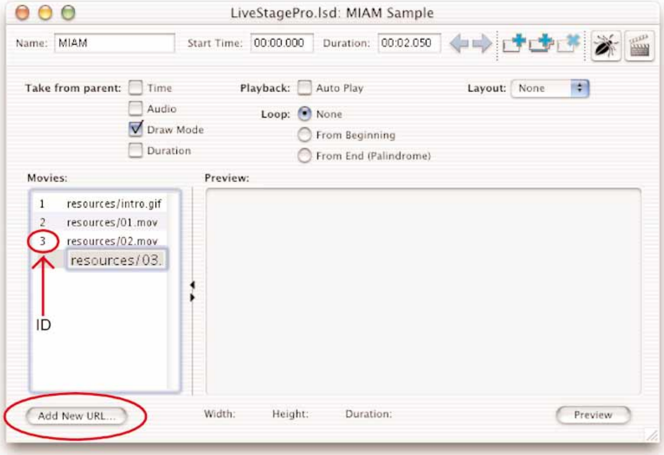
Slice

The Child Movie will be scaled to completely fill the Movie Track while maintaining its aspect ratio. Any portions outside of the boundaries of the Movie Track will not be visible.



MOVIES

The left pane of the Movie Sample Properties window is the Movie List. This is where you add and sort your Child Movies. Child Movies smaller than 100K can be embedded into the movie by either dragging them to this pane. To link a larger file, click the “Add New URL...” button below.



LiveStagePro.lsd: MIAM Sample

Name: MIAM Start Time: 00:00.000 Duration: 00:02.050

Take from parent: □ Time □ Audio ☑ Draw Mode □ Duration

Playback: □ Auto Play

Loop: ● None ○ From Beginning ○ From End (Palindrome)

Layout: None

Movies:

1 resources/intro.gif
2 resources/01.mov
3 resources/02.mov
resources/03.

Preview:

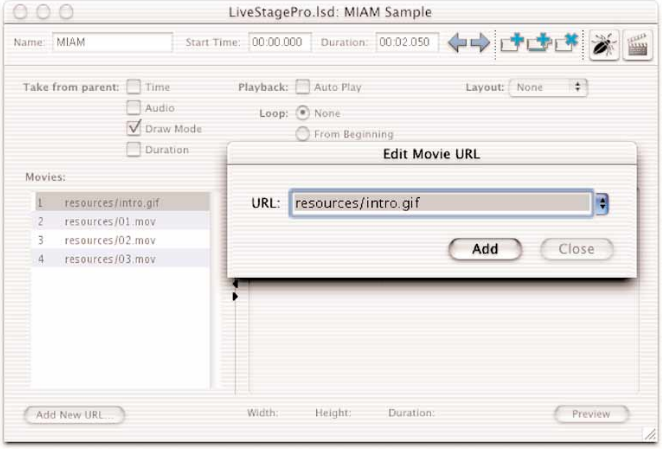
Width: Height: Duration:

ID Numbers

When a Child Movie is added, it is assigned a sequential ID number. The numbers at the left of the Child Movie names indicate the ID numbers of each movie within this list. You can change the sequence of the Child Movies, and thus their ID numbers, by selecting and dragging them to new locations in the list. These ID numbers are used when accessing the Movie Track via QScript.

Modifying Movies in the Movies List

You can change the URL of a linked movie by double-clicking on the Child Movie in the Movies list. The “Edit URL” dialog opens, where you can enter a new path. You can also click on the arrows to the right of the URL field to select from previously entered URLs. URLs can also be added to this list through QScript.



LiveStagePro.lsd: MIAM Sample

Name: MIAM Start Time: 00:00.000 Duration: 00:02.050

Take from parent: □ Time □ Audio ☑ Draw Mode □ Duration

Playback: □ Auto Play

Loop: ● None ○ From Beginning

Layout: None

Movies:

1	resources/intro.gif
2	resources/01.mov
3	resources/02.mov
4	resources/03.mov

Edit Movie URL

URL: resources/intro.gif

PREVIEW

You can preview your embedded and linked Child Movies, including those that have HTTP and RTSP URLs, in the Preview pane. The preview will display the Child Movie’s dimensions and duration. When a streaming Child Movie is selected, a Preview button will appear, giving you the option to preview the stream.

tip

Embed your own ‘loading’ graphic as your initial Child Movie to avoid the default QuickTime Loading box that appears when the MIAM has nothing else to load.

Building a Movie Track

1. Create a Movie Track using one of the options described previously in the “Working with the Movie Track” section.

   Once the Movie Track has been created, you need to consider how you want to bring in the Child Movies.

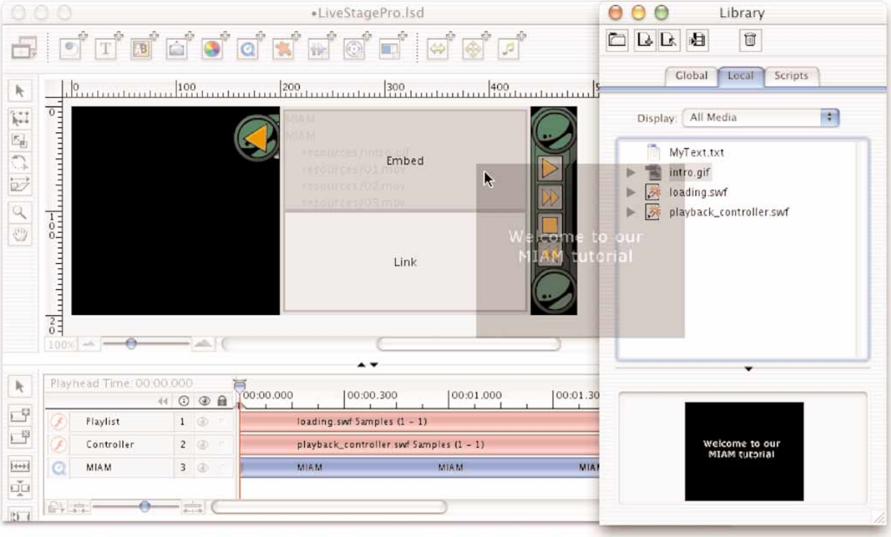
EMBEDDING CHILD MOVIES

The file size of a Child Movie you embed into a Movie Track has to be very small, no greater that 100K. Embedded Child Movies are saved as part of the Parent Movie’s file, increasing the overall file size. QuickTime will also load embedded Child Movies completely into RAM. You can embed media in the Movie Track by dragging and dropping the element into the Movie Sample Properties window, the Stage, or the Timeline.

Embedding through the Stage or Timeline

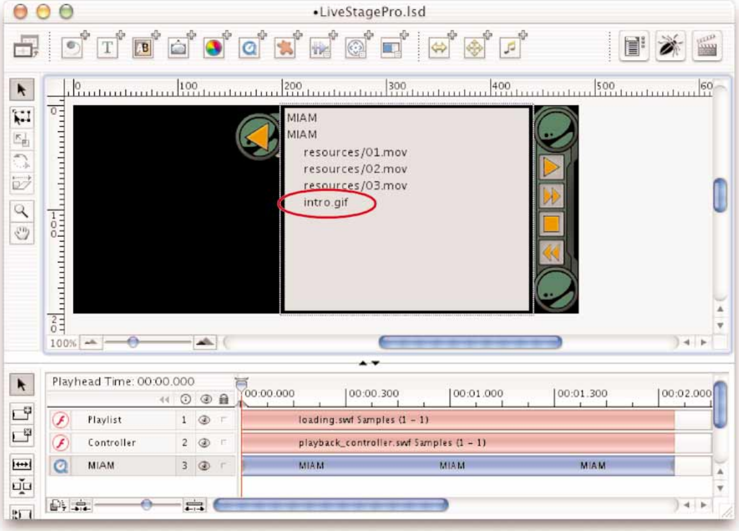
2. Drag the media object you want embedded from the Library onto the Movie Track in either the Stage or the Timeline.

   As you drag the media over the track, two options appear “Embed” and “Link”.



3. Drop the media onto the “Embed” section.

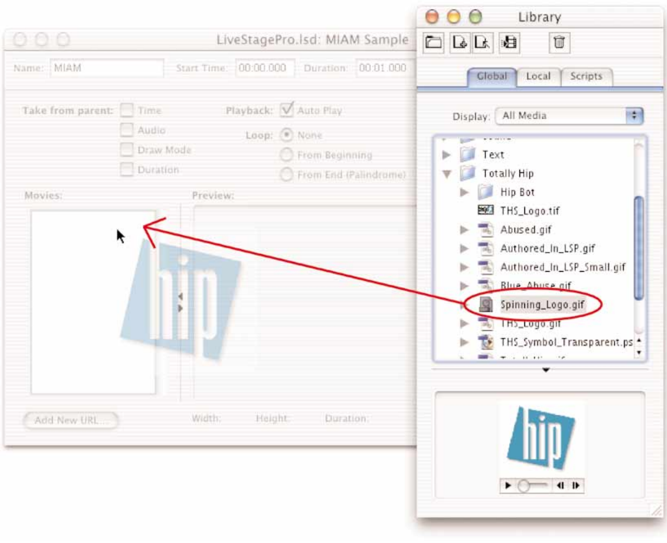
   The media element appears in the track on the Stage, along with its name.



Embedding through the Movie Sample Properties Window

1. Double-click the Movie Track sample on either the Stage or the Timeline to open the Movie Sample Properties window.

2. Drag the media element you want embedded to the Movie List pane.



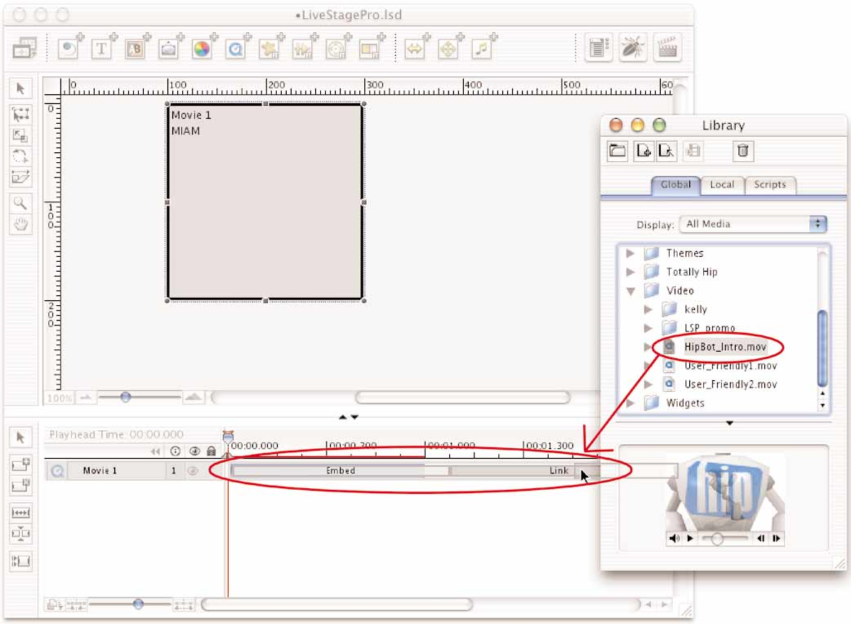
LINKING CHILD MOVIES

Linked Child Movies are only added to the movie and downloaded to the user's machine when the user or the Root movie requests them. As media contained in linked movies are not saved within the overall movie, they do not add to your movie's file size, and they can be edited independently of the container movie. Any Child Movie over 100K will need to be linked.

Linking through the Stage or Timeline

1. Drag the media element you want to link from the Library to the Stage or the Timeline.

   Two options appear: 'Embed' and 'Link'



2. Drop the object onto the ‘Link’ option.

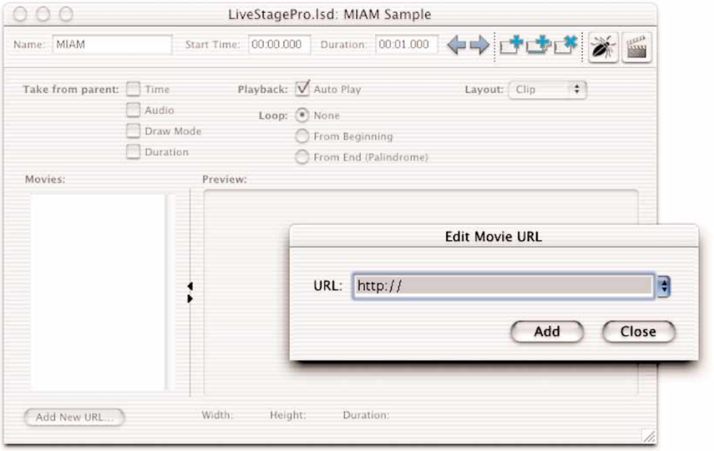
   The object’s name, with an absolute URL, showing its location on your computer appears on the Stage.

Linking through the Movie Sample Properties Window

1. Double-click the Movie Track sample on the Stage or in the Timeline to open the Sample Properties window.

2. Click on the “Add New URL...” button.

   A dialog window opens.



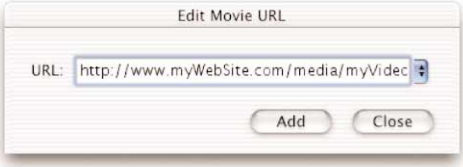
LiveStagePro.lsd: MIAM Sample

Name: MIAM
Start Time: 00:00.000
Duration: 00:01.000
Take from parent: □ Time □ Audio □ Draw Mode □ Duration
Playback: ☑ Auto Play
Loop: ● None ○ From Beginning ○ From End (Palindrome)
Layout: Clip

Edit Movie URL
URL: http://

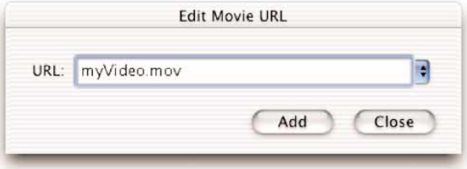
3. Type in the location of the media element you want to incorporate. This location can be absolute, such as an HTTP or RTSP address, or it can be relative to your project file’s location on your computer.

Example 1: The Child Movie to be linked is located at a Web address.



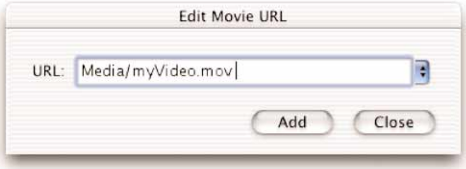
Edit Movie URL
URL: http://www.myWebSite.com/media/myVidec

Example 2: The Child Movie to be linked will be on the end user’s computer in the same folder as the Root movie file.



Edit Movie URL
URL: myVideo.mov

Example 3: The Child Movie to be linked will be located in a folder called "Media" in the same folder on the end user's computer as the Root movie file.



Edit Movie URL

URL: Media/myVideo.mov

4. Continue to add as many Child Movie URLs as you like, clicking the Add button after each entry.

   You can view a history of the URLs you have entered by clicking on the arrow button to the right of the URL field.

Linking Dynamically through QScript

You can also dynamically link a Child Movie by creating the ID and path for a Child Movie through a QScript that can be executed by any scriptable track. Linking dynamically requires two steps. First, the Child Movie must be added to the list and then it can be loaded.

1. In the Script Edit Window of a scriptable track (Sprite, Text, Flash or QTVR), select an Event Handler (i.e. Mouse Click).

2. Using the syntax described below, add a Child Movie to the Movie Track list. This 'introduces' the Child Movie to the Movie Track and assigns it an ID number.

   Identify the name of the Movie Track, the ID you want the Child Movie to have, and the path to the Child Movie's location. Any embedded or linked Child Movies that have the same ID number will be deleted from the list.

   ```
   TrackNamed("MovieTrackName").AddChildMovie(ID#, "URL")
   ```

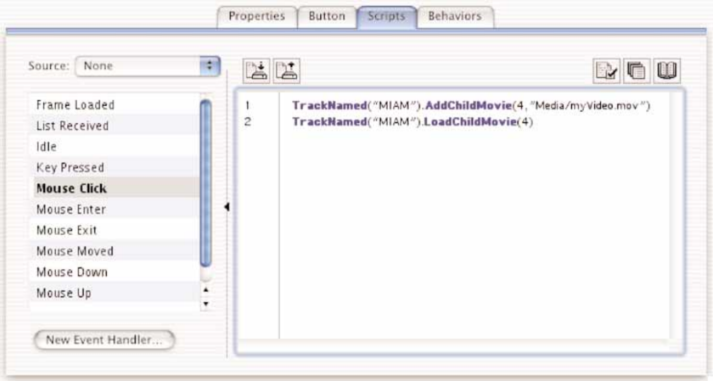
3. Load the Child Movie into the Movie Track by identifying the Movie Track and the ID of the Child Movie you want to load.

   ```
   TrackNamed("MovieTrackName").LoadChildMovie(ID)
   ```

Example:

```
TrackNamed("MIAM").AddChildMovie(4,"Media/myVideo.mov")
TrackNamed("MIAM").LoadChildMovie(4)
```

This script dynamically adds myVideo.mov, (located in the Media folder in the same directory as the Root movie file) into ID number four in a Movie Track named MIAM. It then loads myVideo.mov into the Movie Track via its newly created ID.



```
TrackNamed("MIAM").AddChildMovie(4, "Media/myVideo.mov")
TrackNamed("MIAM").LoadChildMovie(4)
```

Controlling your Movie Track

Movie-in-a-Movie may be controlled through QScript. For instance, you can have a button on a Sprite Track controller load and play your Child Movies on demand. Through QScript you can add, load, and replace Child Movies from your Movie Track's Movies list. You can also control playback.

LOADING CHILD MOVIES

Before a Child Movie can be played, it must be loaded. The first Child Movie in the Movies list (ID number 1) is loaded automatically when the root movie loads. Therefore, it is a good idea to use a placeholder graphic as the first Child Movie and to embed it. Doing so will provide a clean user interface and will remove the default QuickTime movie loading graphic on start up. The remaining Child Movies may be loaded via QScript from a scriptable track using the following form:

```
TrackNamed("MovieTrackName").LoadChildMovie(ID)
```

tip

Think of the syntax "ChildMovieTrack Named" as meaning "Child Movie in the Track Named."

CONTROLLING CHILD MOVIES

Once your Child Movie is loaded you can control it by targeting it through the Name or Index Number of the Movie Track that contains it:

```
ChildMovieTrackNamed("MovieTrack_Name").StartPlaying
ChildMovieTrackofIndex(MovieTrack_Index#).StartPlaying
```

Other than adding and loading a Child Movie into a track, any scripts that affect a Child Movie must be targeted in this way.

ADVANCED USE OF THE MOVIE TRACK

MIAM Targets

Targeting is a good habit to adopt throughout your QScripts (see Chapter 16—Introduction to QScript, page 320, to learn about Targeting), and is critical when dealing with Movie-in-a-Movie. When writing QScript for Movie-in-a-Movie, specify both the track and a movie target within that track.

PARENT AND CHILD TARGETS

Root and Parent Movies communicate with their Child Movies easily:

Targeting the Movie Track:

TrackNamed(“MovieTrack_Name”)
TrackOfID(MovieTrack_ID)
TrackOfIndex(MovieTrack_Index)

Targeting a Child Movie within a Movie Track:

ChildMovieTrackNamed(“MovieTrack_Name”)
ChildMovieTrackOfID(MovieTrack_ID)
ChildMovieTrackOfIndex(MovieTrack_Index)

Child Movie Targeting a Parent

Child Movies can communicate with their Parent Movies. As the Child can only have a single Parent, there is no need to specify its name, Index Number, or ID:

ParentMovie

Targeting the Root Movie

Movies on any level, including both Child and “grandchild” Movies, can communicate with the Root movie.

RootMovie

HIERARCHICAL TARGETING

The more complex the communications between MIAM components, the more care you will need to exercise in their targeting. Target your components following the MIAM hierarchy, starting from the outermost component and moving inwards through the hierarchy:

Example: Movie>Track>Sprite>Sample

Child Movie Targeting a Child Movie

Child Movies can communicate with each other, but must do so via their Parent Movie:

ParentMovie.ChildMovieNamed(“ChildMovieName”)

Child Movie Targeting Parent Components

Child Movies must communicate through their Parent to target the Parent’s tracks and sprites.

ParentMovie.TrackNamed(“Track_Name”)
ParentMovie.TrackOfIndex(Track_Index)
ParentMovie.TrackofID(Track_ID)
ParentMovie.TrackNamed(“SpriteTrack_Name”).SpriteNamed(“Sprite_Name”)

If your Child Movie is a QuickTime movie and you have entered an Identification Name (see Chapter 3—Working in the LiveStage Professional Environment), there is no need to target through the Parent. Simply using ChildMovieNamed(“Identification_Name”) will automatically target the Child Movie, regardless of what track it is housed in.

TROUBLESHOOTING

Child Movie Plays before Fully Loaded

Problems can arise when a Child Movie begins to play before it has been sufficiently loaded. This can occur when Auto Play is selected for the Child Movies, or if the StartPlaying command immediately follows the LoadChildMovie(ID) command.

To fix this, deselect the Auto Play option, and instead test for how much of the file has been loaded, initializing playback once a sufficient amount has been reached.

```
Example:    IF(ChildMovieTrackOfIndex(2).MaxLoadedTimeInMovie > 100)
            ChildMovieTrackOfIndex(2).StartPlaying
            END IF
```

tip

Due to a known issue with QuickTime, if a Custom Event in a sprite contained in a Parent Movie is being triggered by a Child Movie, that Custom Event will act as if it is within the Child Movie itself.

VR Child Movies and QuickTime 5

If you are deploying for QuickTime 5 and are using a VR as a Child Movie, the VR will need to be loaded twice before it will work, due to a bug that has since been corrected in QuickTime 6. You can do this using the following QScript:

```
GlobalVars gLoading
gLoading=0
TrackNamed("MovieTrack").LoadChildMovie(ID of VR Child Movie)
gLoading=1
```

The above script should be placed on the Event Handler that you want to have load the VR (e.g., a Mouse Click)

Then in the Idle Event Handler of a Sprite Track, enter:

```
GlobalVars gLoading
IF(gLoading=1)
        IF(ChildMovieTrackNamed("MovieTrack").MaxLoadedTimeInMovie >
        (ChildMovieTrackNamed("MovieTrack").GetDuration/2))
        TrackNamed("MovieTrack").LoadChildMovie(ID of VR Child Movie)
        gLoading=0
        ENDIF
ENDIF
```

The VR Child Movie is loaded once, and then loaded a second time once half of the VR has loaded in.

MOVIE TRACK WRAP UP

The many advantages the Movie Track provides will no doubt shape many of your future projects as you grow more familiar and proficient with Movie-in-a-Movie. The Project Samples folder on the CD-ROM has a folder of MIAM examples to help you get a feel for the MIAM experience. You will also find a number of detailed tutorials and articles on MIAM on StageDoor. Additionally, visit http://www.toddishere.com to see how MIAM, used in conjunction with QTLists (see Chapter 18—QTLists and XML), can provide database-driven on-demand content delivery.

Chapter 12

The Streaming Track

STREAMING TRACK OVERVIEW

The Streaming Track references RTP/RTSP streaming media. Streaming media consists of live or pre-recorded events that are housed on a streaming server. As opposed to HTTP or progressive download delivery, RTSP streams do not download to the user's computer. The data packets are received, assembled, played and then discarded, making RTSP streams a great alternative for presenting large movies that would be impractical for the user to download.

A Streaming Track does not consist of an actual streaming file. Instead, it contains a pointer movie that references the actual streaming file on the streaming server. You can use sprites, QTVR, Flash, and other tracks in conjunction with the Streaming Track to add interactivity and other enhancements to your streamed presentation.

Streaming movies need to be placed on a Streaming server (RTP/RTSP) in order to stream. A common error in authoring projects with the Streaming Track is to bring the actual streaming QuickTime movie into the Streaming Track. Instead, you need to create a 'pointer' movie that references the streaming movie's URL on the server. This pointer movie is what should be then incorporated into the LiveStage Professional Streaming Track. To learn more about streaming QuickTime movies, visit http://www.apple.com/quicktime/tools_tips/tutorials/streaming.html.

WHEN TO USE A STREAMING TRACK

Streaming is not always the best answer for deployment. At any given bit rate streaming media is likely to provide poorer quality than HTTP delivery. HTTP delivery has zero packet loss while streaming media may, and most likely will, exhibit some packet loss over the Internet. This is less of an issue if you are streaming over an Internal LAN.

Use a Streaming Track when:

- You are presenting a live event that you want to deliver to your audience in real time.
- Your presentation's file size is too large for your users to download.
- Your pre-recorded presentation is long and you want to provide a means for your user to jump to different sections or chapters on demand.
- You want to prevent the average user from being able to save your presentation to their computer, providing an extra (though not bulletproof) layer of protection.
- You have a live or on demand event with multiple streams that you would like to incorporate into a single player, so that the user can switch to the camera angle of their choice.

Note that you will need to have your media housed on a streaming server to make use of the Streaming Track. Darwin Streaming Server is a free open-source streaming server available from Apple.

To determine the duration of a pre-recorded streaming movie, you can use the Movie Monitor tool movie located in the Tools menu. The Movie Monitor tool provides information on any QuickTime movie selected.

WORKING WITH THE STREAMING TRACK

Some Streaming Tracks, as in the case of live events, do not have a predetermined duration. When you are constructing your QuickTime movie around a live stream that has an unknown duration, place an asterisk in the duration field of each track to give the track an unlimited duration.

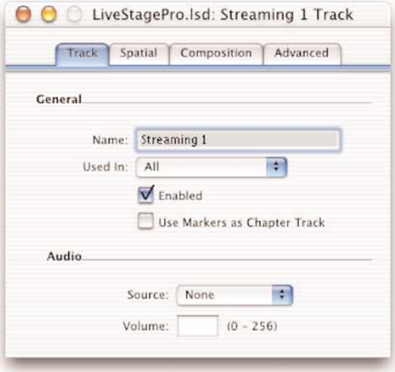
The Streaming Track is not scriptable, but can be controlled through any scriptable track.

There are a few ways to add a Streaming Track to your project:

- Select Movie>Create Track>Streaming... from the Menu Bar. In the dialog window that opens, navigate to your pointer movie's location on your computer to select it.
- Control-click in the Track List area of the Timeline to access the contextual menu and select "Create Streaming Track...".
- Simply drag the pointer movie file to the Stage or Timeline.

Streaming Track Inspector/Properties Window

The Streaming Track Properties window and Streaming Track Inspector have all the standard settings described in Chapter 3—Working in the LiveStage Professional Environment, page 106.The one difference from most other tracks is the presence of the Volume and Balance controls. You can set the volume and balance for your Streaming Track in either the Audio tab of the Track Properties window or in the Basic tab in the Inspector. You can also attach a Tween or Modifier Track as a Source for the Balance or Volume to create different effects. For more information on Tween and Modifier Tracks, see Chapter 13—Additional LiveStage Professional Tracks. These sources can be specified in the Audio tab of the Streaming Track Properties window, or in the Advanced tab of Streaming Track Inspector.



LiveStagePro.lsd: Streaming 1 Track

General

Name: Streaming 1

Used In: All

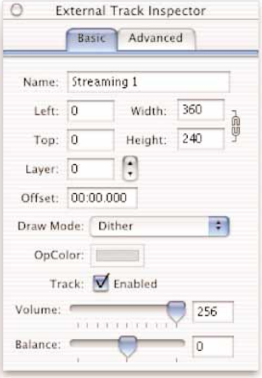
☑ Enabled

□ Use Markers as Chapter Track

Audio

Source: None

Volume: (0 - 256)



External Track Inspector

Name: Streaming 1

Left: 0 Width: 360

Top: 0 Height: 240

Layer: 0

Offset: 00:00.000

Draw Mode: Dither

OpColor:

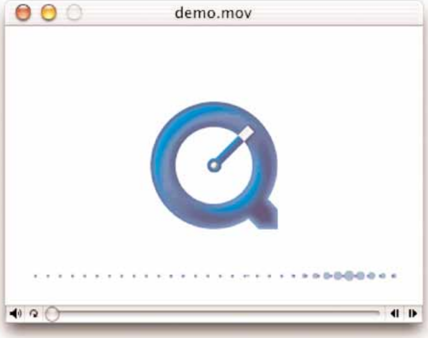
Track: ☑ Enabled

Volume: 256

Balance: 0

Streaming Sample Properties Window

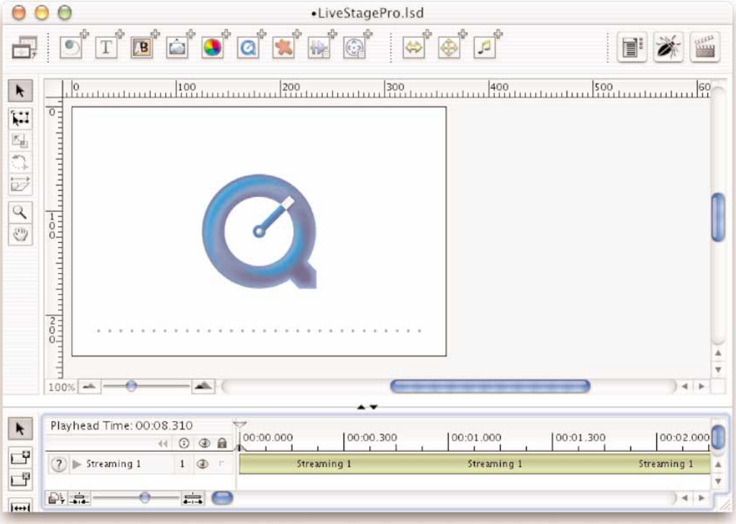
Since a Streaming Track sample cannot be edited or augmented, double clicking on the Streaming sample simply displays a preview of the Streaming Track.



Building a Streaming Track

1. Create a Streaming Track using one of the above methods.

   The Streaming Track will appear in the Timeline and a representation of the Streaming Track will appear on the Stage.



2. Double click the Streaming sample if you wish to view a preview of the stream.

ADVANCED USE OF THE STREAMING TRACK

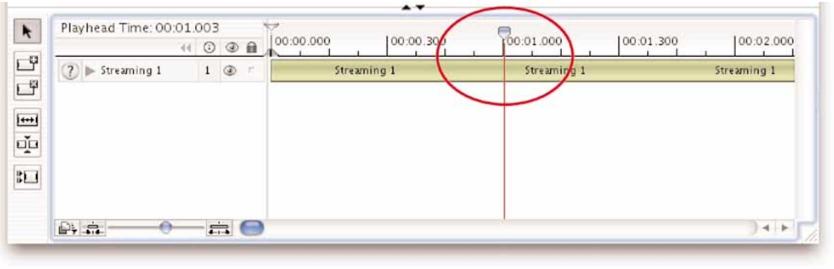
Using Markers to Create a Chapter Track

If your streaming movie is lengthy and you want to afford users the opportunity to access discrete points in its timeline on demand, you can create “Chapters”. Users can click on individual Chapters displayed in the QuickTime Player to jump to a particular point in the presentation that interests them. For instance, if your movie is a lecture covering a range of subjects, you could assign a different Chapter to the beginning of each new topic.

1. Review your Streaming movie and choose the points where you think a Chapter would be appropriate. Note the point in time when each of these sections begins.

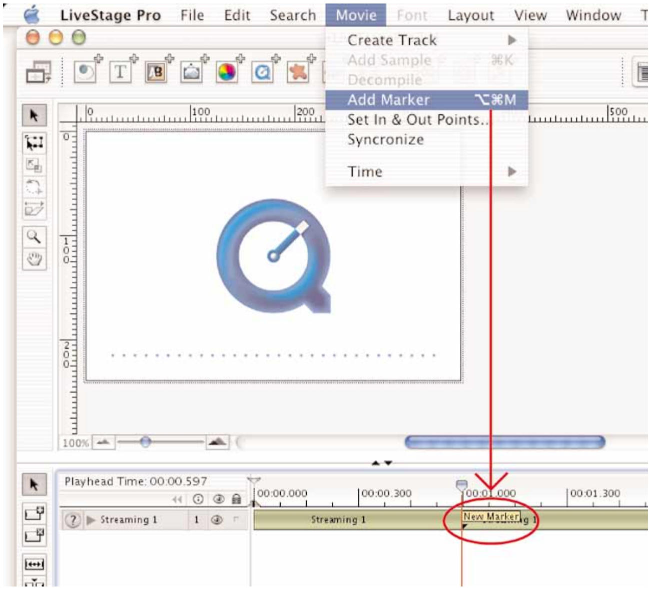
2. In LiveStage Professional’s Timeline, select the Streaming Track by clicking on it once.

3. Move the Playhead to the point in time where you want to place your first Chapter.



4. From the Menu Bar, select Movie>Add Marker (or hit the letter M on your keyboard).

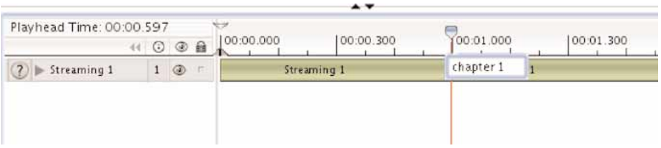
   A new marker appears in the Streaming Track sample.



5. Click once on the Marker in the Timeline.

   The in-place editor becomes active.

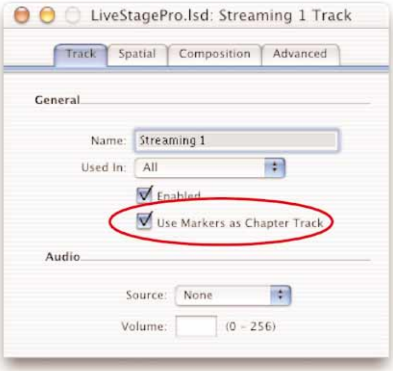
6. Enter the name of your first Chapter and then press the Return key.



7. Repeat Steps 3 through 6 for each of the Chapters you wish to add.

8. Double click the Streaming Track Header to open the Track Properties window.

9. Select the “Use Markers as Chapter Track” option.



LiveStagePro.lsd: Streaming 1 Track

General

Name: Streaming 1

Used In: All

☑ Enabled

☑ Use Markers as Chapter Track

Audio

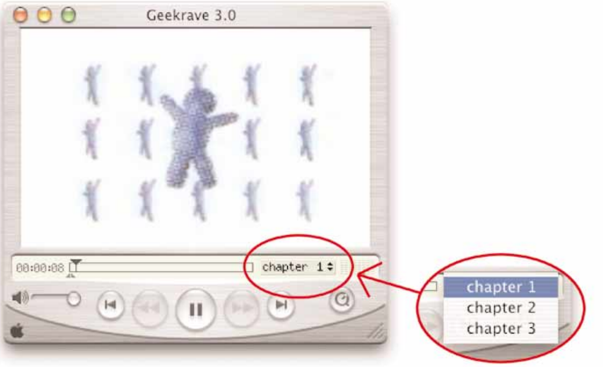
Source: None

Volume: (0 - 256)

Close the Track Properties window.

10. Export your QuickTime movie (Command-E) and open it.

Select from the Chapters listed at the bottom right corner of the QuickTime Player to jump to the appropriate sections.



Geekrave 3.0

tip

QT 6 has a new a feature called Instant-On that minimizes the buffering time of a stream if the user has sufficient bandwidth.

Making Good Use of the Buffering Screen

There is a ‘buffering’ delay of usually eight to twelve seconds when accessing a stream, during which time a generic QuickTime loading screen generally plays. To enhance the user’s viewing experience while your stream is buffering, you have a few options:

- Set the streaming track’s drawing layer to a value that hides it behind another track, Chapter 14—Working with Multiple Tracks, page 281). Then use QScript to test when the stream begins to play and then change the Drawing Layer, moving the Streaming Track to the foreground. StageDoor tutorial #26 explains the process for testing a stream’s status.

- Provide one or more images to coincide with the buffering, paused, or playing status of your stream. On StageDoor (http://stagedoor.totallyhip.com), Tutorial #26 describes this process in detailed steps.

STREAMING TRACK WRAP UP

Adding interactivity to a Streaming Track can greatly enhance the user's experience. It has been used to great effect with many major events. Everything from online learning to entertainment can benefit from streams enhanced by interactivity. Check out the Featured Sites section of StageDoor to experience some impressive interactive streams. To learn about streaming and about how to prepare a streaming movie, visit the following tutorials on the Apple site:

An Introduction to QuickTime Streaming
http://www.apple.com/quicktime/qttutorial/streaming.html

Preparing Stored Media for Streaming
http://www.apple.com/quicktime/authoring/qtss/pgs/qt03.htm

Chapter 13

Additional LiveStage Professional Tracks

OVERVIEW

Though not as commonly used as the other tracks in LiveStage Professional, the Color, Effects, Instrument, Tween, and Modifier Tracks each have their own unique functionality that can significantly enhance your movie.

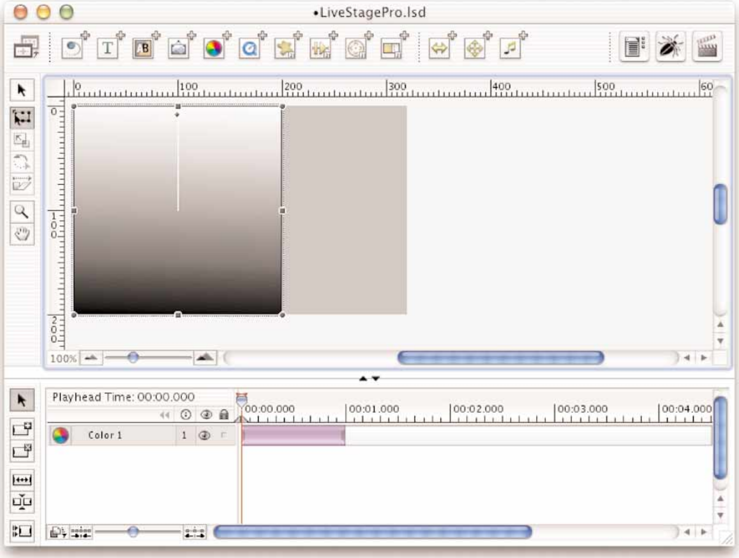
THE COLOR TRACK

The Color Track is a simple track that can be used to easily provide a solid color or gradient background to your movie through the QuickTime vector codec. You can also use the Color Track to provide backdrops for various elements in the movie, or put several instances of this track together to provide a unique and changing background. Regardless of the dimensions of your Color Track, the addition to your movie's file size will be very small.

Working with the Color Track

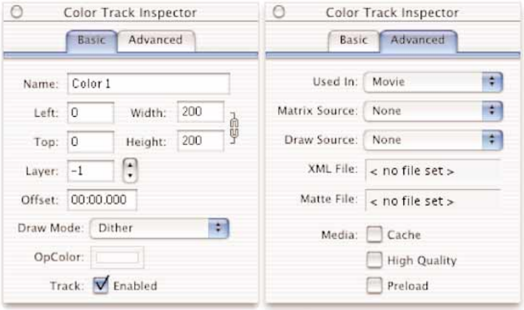
Color Tracks can house multiple samples, each containing different color or gradient information. By using multiple samples, you can animate the color or gradient over time. Colors and gradient properties are set through the Color Sample Properties window.

To create a Color Track, click on the in the Track Creation Toolbar or select Create Track>Color (Option-Command-C) from the Movie menu on the Menu Bar. You can also control-click in the Track List area of the Timeline to open a contextual menu that will allow you to create a Color Track from there. A Color Track with a single sample appears in the Timeline and on the Stage as a simple black and white vertical linear gradient.



COLOR TRACK INSPECTOR/TRACK PROPERTIES WINDOW

The Color Track Inspector and Properties Window both offer the basic parameters explained in detail in Chapter 3—Working in the LiveStage Professional Environment, page 106.

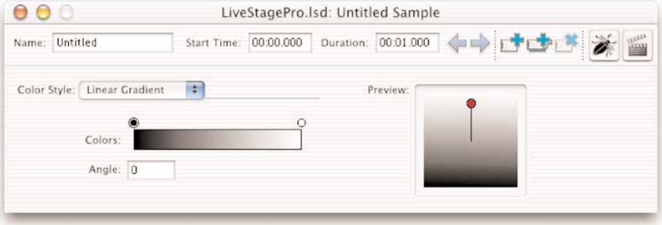


Name: Color 1
Left: 0 Width: 200
Top: 0 Height: 200
Layer: -1
Offset: 00:00.000
Draw Mode: Dither
OpColor:
Track: ☑ Enabled

Used In: Movie
Matrix Source: None
Draw Source: None
XML File: < no file set >
Matte File: < no file set >
Media: □ Cache
□ High Quality
□ Preload

COLOR TRACK SAMPLE PROPERTIES WINDOW

Double clicking on the Color sample opens the Color Sample Properties window. Like other Sample Properties windows, you can modify the Color sample's name, start time, and duration, as well as navigate through and edit additional samples. The different options and buttons along the top of the Color Sample Properties window are explained in Chapter 3—Working in the LiveStage Professional Environment, page 88.



Name: Untitled Start Time: 00:00.000 Duration: 00:01.000
Color Style: Linear Gradient
Colors:
Angle: 0
Preview:

In addition to the common features, the Color Sample Properties window provides three tools for creating and modifying the color scheme within the sample.

Color Style

This popup menu offers three options for the rendering of the Color sample. You can choose from a Solid Color, Linear Gradient, or Circular Gradient.

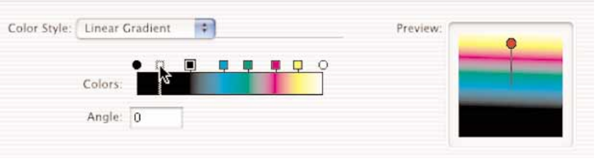
Colors

You can select the different colors in your gradient through the Colors Editor. Above the Editor at either end are two circular color chips. Double-clicking on either of these chips opens a Color Picker that enables you to replace the color at that end of the current gradient. Any alterations you make in the Colors Editor will be reflected in the Preview window to the right, as well as on the Stage.

- Add new colors by clicking on either circular chip and dragging to a new position on the Colors Editor. When you release the mouse, a square chip appears and the Color Picker opens. Select a new color to add to your gradient or click 'Cancel' to maintain the original color.

- Add, delete, and reposition as many square chips as you like to create highly colorful effects.
- Reposition chips by clicking and dragging. Only the square chips can be repositioned; the circular chips cannot. You can increase the spread of a color by making the second chip the same color as the first and repositioning it along the gradient editor.
- Modify chip colors by double clicking on them to open the Color Picker.
- Delete them by clicking on them and pressing the delete key. Only the square chips can be deleted; the circular chips cannot.

You can hold down the shift key while dragging the linear gradient angle dot to snap the angle to multiples of 30 and 45 degrees.



Color Style: Linear Gradient

Colors:

Angle: 0

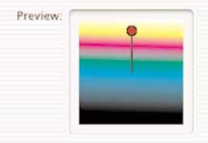
Preview:

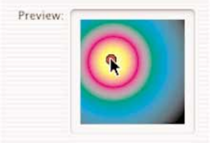
Angle

When you select a Linear Gradient, the Angle control is activated. You can enter the angle value in degrees in this field to alter the axis on which the gradient is drawn. The results will appear in the Preview window as well as on the Stage.

Preview

The Preview window acts a visual representation of your Color sample, as well as a gradient angle editor. When the Gradient option is selected in the Color Style popup, a red dot appears in the Preview window. Repositioning this dot will alter the angle of a Linear Gradient or the center of a Circular Gradient. Again, the effects are visible in real time in both the Preview window and the Stage.





THE EFFECT TRACK

The Effect Track can be used to achieve a variety of visual effects, including movie style transitions and special effects. QuickTime Effects are processed in real time on the user's computer, so they add very little to the file size of the movie. However, they increase the demands on the user's CPU.

The Effect Track can:

- Create effects on a single visual track.
- Transition between two visual sources.
- Create standalone effects that do not require any source.
- Create matte effects for video.

The Effect Track's effects require your user to have the components that a Full Installation of QuickTime installs on their computer. Through the QScript action ComponentVersion, you can detect for needed components that are not installed on the user's system. See Appendix 1—QScript Reference for more information.

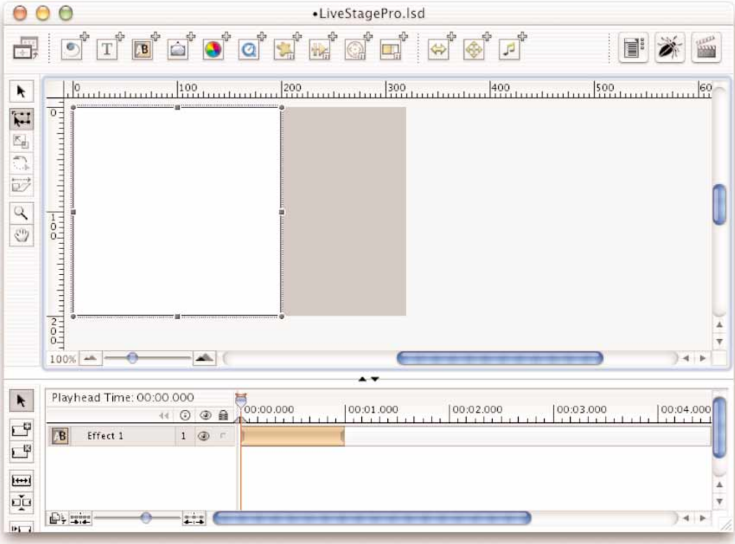
Working with the Effect Track

The Effect Track generally operates on other tracks to create a variety of visual results. Effect Tracks can house multiple samples, all of which can conceivably contain different visual effects operating on different tracks. An effect can use zero, one, two, or three tracks as sources. Depending on the number of sources you select, different types of effects will be made available. When a single source track is chosen, the effect produced is called a Filter Effect. When two source tracks are selected, you will get a Transition or Compositing Effect. Three source tracks combine to create what is known as a Travelling Matte Effect. You can also opt to select no source tracks, which allows you to choose from QuickTime's Special Effects.

While it operates on other tracks, the Effect Track does not directly affect the source track(s). Instead, the visual information of the source track(s) is recreated and the effect added within the Effect Track itself. Therefore, if you want the effect to be seamless and to display properly in your movie, an Effect Track that involves one or more source tracks will need to be positioned correctly in your movie:

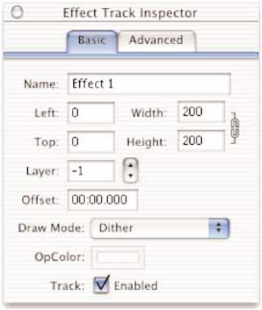
- The position and dimensions of the Effect Track must match the position and dimensions of its source track(s).
- The Effect Track must be on a Drawing Layer above its source track(s). See Chapter 4—Introduction to Tracks, page 121 for more information on Drawing Layers.
- The Effect sample must be located at the same point in the Timeline as the source track(s).

To create an Effect Track, click on the AB in the Track Creation Toolbar or select Create Track>Effect (Option-Command-E) from the Movie menu on the Menu Bar. You can also create an Effect Track through the track contextual menu by control-clicking in the Track List area of the Timeline and selecting "Create Effect Track". An Effect Track with a single sample appears in the Timeline and an empty sample appears on the Stage.

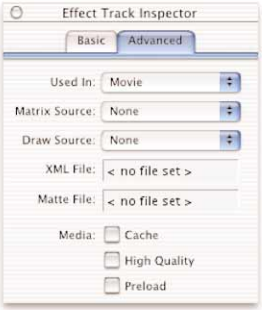


EFFECT TRACK INSPECTOR/PROPERTIES WINDOW

The Effect Track Inspector and Properties Window both offer the basic parameters explained in detail in Chapter 3—Working in the LiveStage Professional Environment, page 106.



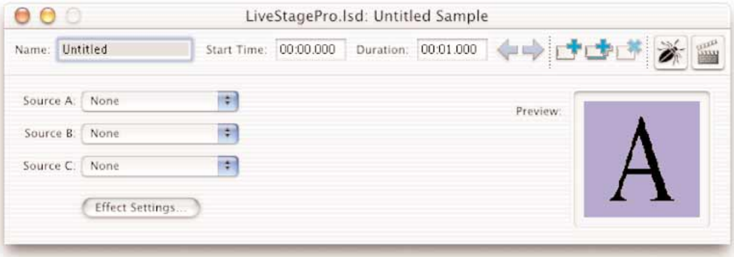
Name: Effect 1
Left: 0 Width: 200
Top: 0 Height: 200
Layer: -1
Offset: 00:00.000
Draw Mode: Dither
OpColor:
Track: ☑ Enabled



Used In: Movie
Matrix Source: None
Draw Source: None
XML File: < no file set >
Matte File: < no file set >
Media: □ Cache
□ High Quality
□ Preload

EFFECT TRACK SAMPLE PROPERTIES WINDOW

Double clicking on the Effect sample opens the Effect Sample Properties window. You can modify the Effect sample's name, start time, and duration, as well as navigate through and edit additional samples through the familiar options and buttons along the top of the Effect Sample Properties window. These are explained in detail in Chapter 3—Working in the LiveStage Professional Environment, page 88.



Name: Untitled Start Time: 00:00.000 Duration: 00:01.000
Source A: None
Source B: None
Source C: None

Source A

Specify the first source track for the Effect in the Source A popup menu. All visual tracks that can be operated on by the Effect Track will appear here. Single-source effects can apply effects like film grain, emboss, contrast, etc. If you want the Effect Track to create a Special Effect like Fire, Cloud, or Ripple, select 'None' in this and the other two Source menus.

Source B (+ A)

If you want your Effect Track to operate on two or more tracks, specify the second source track here. Two-source effects are transitions. For instance, if you want to create a cross fade transition, Source A will be the first element that fades out, while Source B will be the second element that appears in its place. In this case, the Effect sample has to "overlap" the two track's samples.

Source C (+A+B)

Selecting a source here creates a three-source effect: a traveling matte. The track you select here will serve as the matte for your Traveling matte effect. Any black areas in the source will reveal the Source A track, while the white areas will display the Source B track.

Preview

The Preview pane provides a basic example of the effect's functionality and appearance.

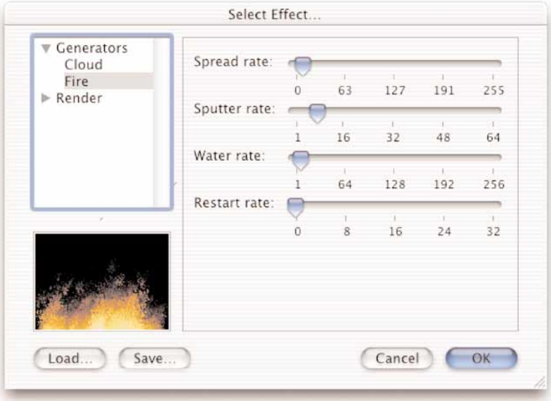
Effect Settings

This button opens the Select Effect... window so that you can set the type of Effect and its parameters.

The Select Effect... window will change depending on the version of QuickTime you have installed. To view the most complete set, you will want to have QuickTime 6 installed.

SELECT EFFECT... WINDOW

The selections you make in the Source popup menus will determine the Effects that the Select Effect... window provides. The window is made up of three main areas:



Effect List

This pane lists all the available effects according to the number of Sources you have selected.

Preview

You can see a basic demonstration of the effect in action in this section.

Properties

This area presents the different properties that can be adjusted for the selected Effect. These properties will affect how the effect looks when it is applied to your visual track.

Creating Effects

As already mentioned, there are several types of effects that you can create depending on the number of source tracks you are using.

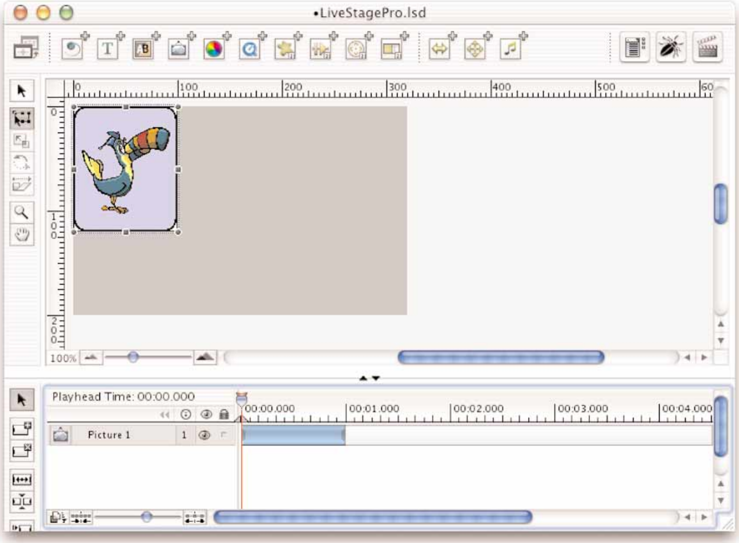
FILTER EFFECTS

Filter Effects visually modify single source tracks. These are basic operations you would perform on a digital picture when editing it. You can add a lens flare, blur, 'film noise', or even make the image or video sepia toned.

1. Open the Library window by selecting Window>Show Library Window (Command-Y).

2. Click on the Global tab and open the Images folder.

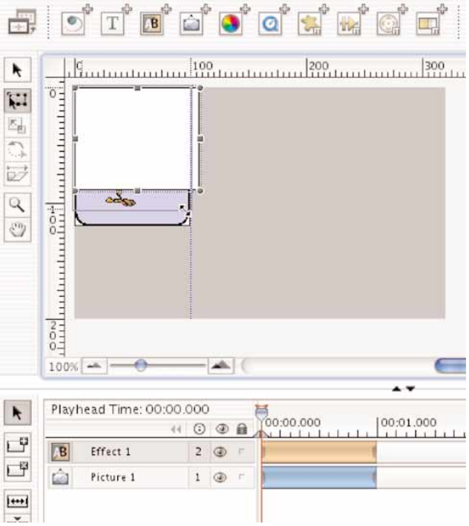
   From the Animals folder, drag an image directly to the Stage or Timeline. A Picture Track is created.



3. Create an Effect Track using one of the methods described earlier.

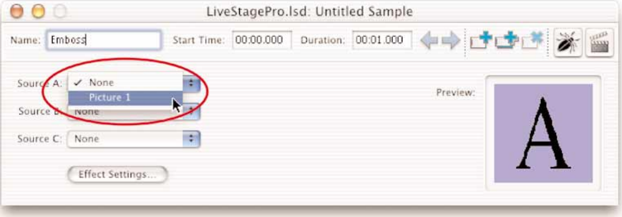
   The Effect Track appears with an empty sample in the Timeline and on the Stage.

4. On the Stage, resize the Effect Track to cover the Picture Track.



5. Double-click the Effect sample to open the Effect Sample Properties window.

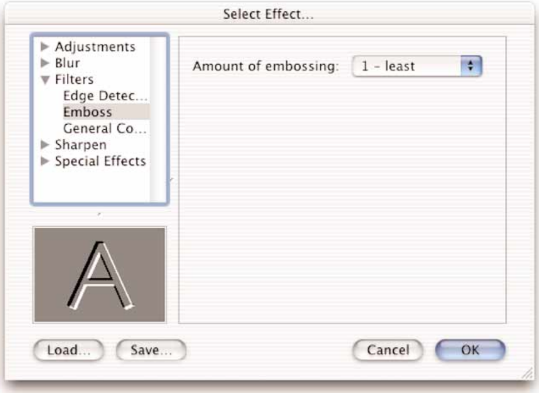
6. Name the sample "Emboss" in the Name field.



7. In the Source A popup, select the "Picture 1" track and click the Effect Settings button at the bottom.

The "Select Effect..." window opens.

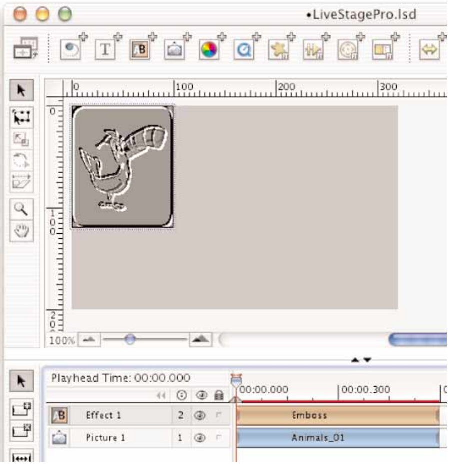
8. Click on the triangle next to 'Filters' in the Effect List and select 'Emboss'.



The Preview displays an example of the effect 'Emboss' will produce.

9. Click OK.

The "Select Effect..." window closes and the Stage displays the image in the Picture 1 track with the Emboss effect applied.



TRANSITION & COMPOSITOR EFFECTS

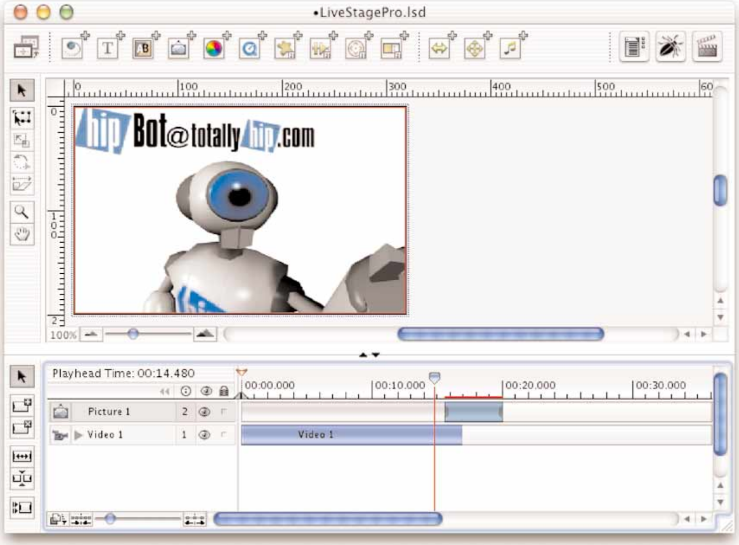
Transition effects can be used to change smoothly from one visual element to another. A very common transition is a film style Cross Fade, which fades one visual element out while simultaneously fading a second one in. In addition to Cross Fade, the Effect Track offers a number of other wipes and transitions. To create a transition the two sources must overlap each other for the duration of the transition.

You can also use two-source effects to blend two visual elements together. Two-source effects like these work best when the two tracks have the same dimensions and position on the Stage.

1. Create the two visual tracks that you want to use for the transition.

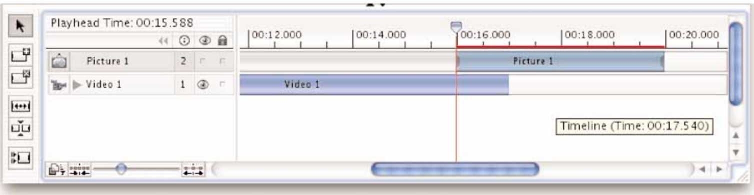
   These can be a combination of Picture, Video, or Streaming Tracks.

2. Position them over one another on the Stage. For best results, make sure they have identical dimensions. For information on how to reposition and resize elements on the Stage, see Chapter 3—LiveStage Professional Environment, page 70.



3. Move the second track to the right along the Timeline, so that the beginning of the second track overlaps the end of the first track.

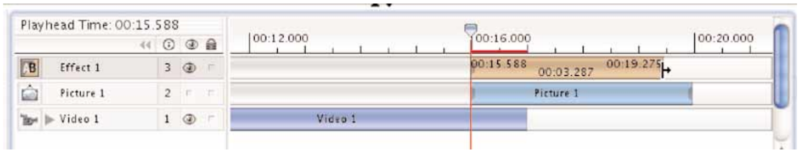
   The duration of the overlap will determine the length of the transition. For a smooth transition, you will probably want a minimum of 1 second of overlap.



4. Create an Effect Track by clicking on the B.

5. Move the Effect sample to the start of your second (in time) track's sample.

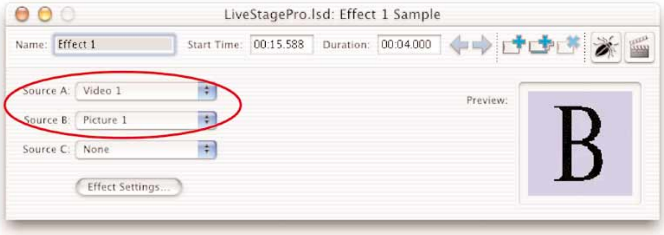
6. Drag the right edge of the Effect sample to the point where the first (in time) track ends.



7. On the Stage, resize and reposition the Effect Track to cover the two source tracks exactly.

8. Double-click on the Effect sample either on the Stage or in the Timeline to open the Sample Properties window.

9. Select your first track in the Source A popup menu, and your second track in the Source B popup menu.


LiveStagePro.lsd: Effect 1 Sample

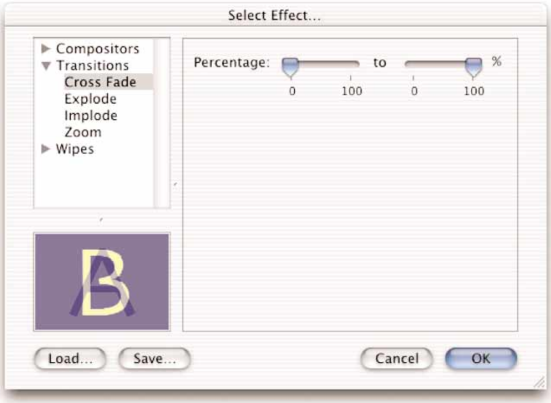
Name: Effect 1 Start Time: 00:15.588 Duration: 00:04.000

Source A: Video 1

Source B: Picture 1

Source C: None

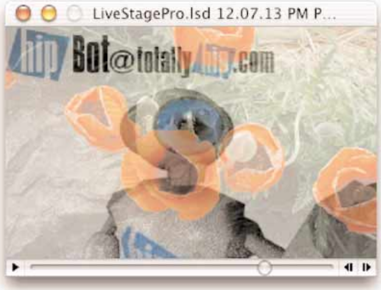
10. Click the Effect Settings button and select a transition from the Effect List.


Select Effect...

Percentage: 0 to 100 %

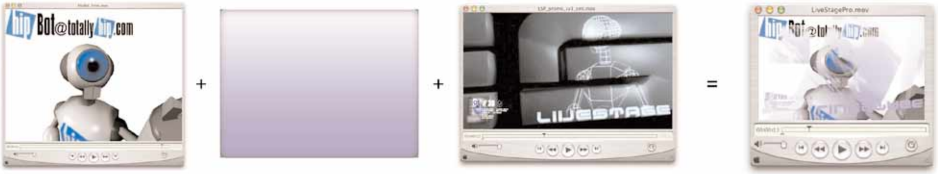
Click OK.

11. Preview your movie to view the transition.

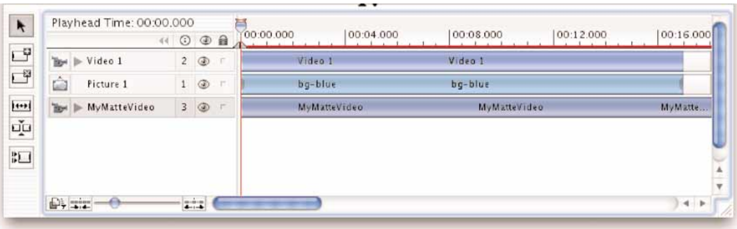


TRAVELING MATTE [NEW IN LIVESTAGE PROFESSIONAL 4.0!]

The Traveling Matte is a new effect that allows you to composite two different visual sources with a moving matte. As with most simple 8-bit mattes, the Traveling Matte has 256 levels of transparency; black and darker areas will display one visual source, while the white areas will display the second visual source. However, the Traveling Matte can be a video. Therefore, instead of a static matte, you get an animated composite of the two source tracks.



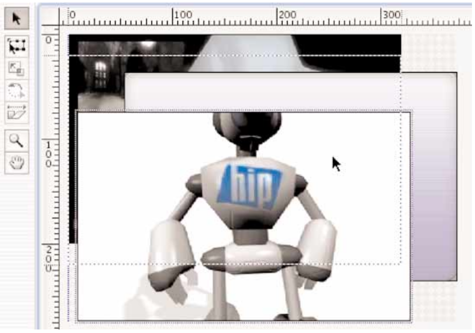
1. Create three visual tracks, one of which should be a video track—the higher the contrast in the video, the greater the effect.



2. Position the three tracks one on top of the other on the Stage. For best results, they should all be the same size.

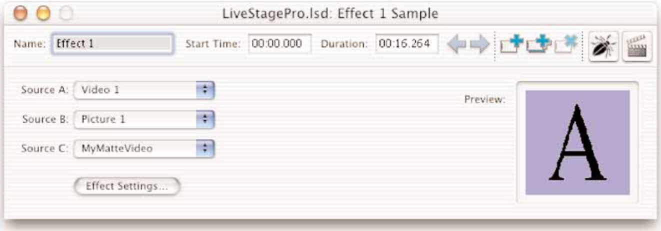
3. Create an Effect Track (Option-Command-E).

4. Resize and position the Effect Track on the Stage directly over the three source tracks.



5. Double-click the Effect sample to open the Effect Sample Properties window.

6. For Source A, select the first visual track.
   For Source B, select the second visual track.
   For Source C, select the matte video track.



LiveStagePro.lsd: Effect 1 Sample

Name: Effect 1 Start Time: 00:00.000 Duration: 00:16.264

Source A: Video 1

Source B: Picture 1

Source C: MyMatteVideo

7. Click the Effect Settings button.

8. Click the triangle next to Compositors and select ‘Traveling...’ and then click OK.

9. Preview your movie to view the effects.

   The first visual track (Source A) will be visible through the dark areas of the video matte. The second track (Source B) will be visible through the light areas.

SPECIAL EFFECTS

QuickTime also offers three special effects that do not require a source track: Fire, Cloud, and Ripple. Just add an Effect Track to your project, open the Effect Tracks’ sample, click on the Effect Settings button, and select the Effect in the Effect List. To view your movie ‘through’ the Fire or Cloud effects, you will need to activate the Effect Track’s Alpha Channel. In the Basic tab of the Effect Track Inspector (Command-I), select ‘Alpha Channel’ for the Draw Mode. Note the effect is not applied to any particular track. It is simply a special effect sitting on its own layer within your project.

Ripple

The Ripple effect gives you the impression that you are viewing your movie underwater. You can use this effect in conjunction with a mask so that only certain areas of your movie will be “under water”.

Fire

Send your movie up in flames with the Fire Effect. Activate the Effect Track's Alpha Channel by selecting it in the Effect Track Inspector's Draw Mode popup menu, and then set various fire properties to achieve the desired effect.

Cloud

Add a spinning nebulous cloud to your movie with or without the Alpha Channel.

tip

What is MIDI? MIDI (Musical Instrument Digital Interface) is an industry-standard protocol that enables computers and musical instruments to communicate. MIDI does not transmit audio itself, but instead provides information about audio events, such as which note has been played, how long it has been played for, how loudly it has been played, and so on. As MIDI is not an actual audio file like .mp3 or .wav, a tremendous amount of MIDI data can be used to generate music and sounds for an extremely small file size.

THE INSTRUMENT TRACK

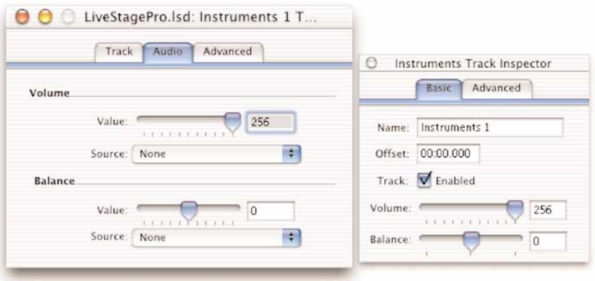
The Instrument Track provides MIDI instruments that can be played through QScript. Using the instruments within the Instrument Track, QScript can create chords, sounds, and alter tones to enhance the user's interactive experience. An Instrument Track can contain any number of instrument specifications. These instrument specifications can be made up of different MIDI instruments or even small audio samples.

Working with the Instrument Track

Instrument Tracks are created by clicking on the ♬ or by selecting Movie>Create Track>Instrument (Option-Command-R). Additionally, you can create an Instrument Track through the track contextual menu that you can access by control-clicking in the Track List area of the Timeline.

INSTRUMENT TRACK INSPECTOR/PROPERTIES WINDOW

The Instrument Track Inspector and Track Properties window offer standard track properties that are available for all audible tracks. You will find the standard settings found in these windows described in Chapter 3—LiveStage Professional Environment, page 106.


LiveStagePro.lsd: Instruments 1 T...
Track
Audio
Advanced
Instruments Track Inspector
Basic
Advanced

Volume
Value: 256
Source: None
Balance
Value: 0
Source: None

Name: Instruments 1
Offset: 00:00.000
Track: ☑ Enabled
Volume: 256
Balance: 0

The one difference from most other Track Inspectors is the presence of the Volume and Balance controls. You can set the volume and balance for your Instrument Track in the Basic tab of the Inspector or in the Audio tab of the Track Properties window. You can also attach a Tween or Modifier Track as a Source for the Balance or Volume to create different effects. These sources can be specified in the Advanced tab of Instrument Track Inspector, or the Audio tab of the Instrument Track Properties window.

INSTRUMENT TRACK SAMPLE PROPERTIES WINDOW

The Instrument Track can contain one or more Instrument samples, each of which may contain multiple instruments. Double-clicking on an Instrument sample in the Timeline opens the Sample Properties window. Common tools across the top of the Sample Properties window allow you to modify and navigate through the samples in the Instrument Track. For more information on the tools available here, see Chapter 3—LiveStage Professional Environment, page 88.

tip

You can also incorporate existing MIDI files into your project, but this is different from an Instrument Track. MIDI files are treated as External Tracks (see Chapter 8—External Tracks) much like audio files, while Instrument Tracks hold only the instruments and must have QScript supply the notes to be played.

Instrument List

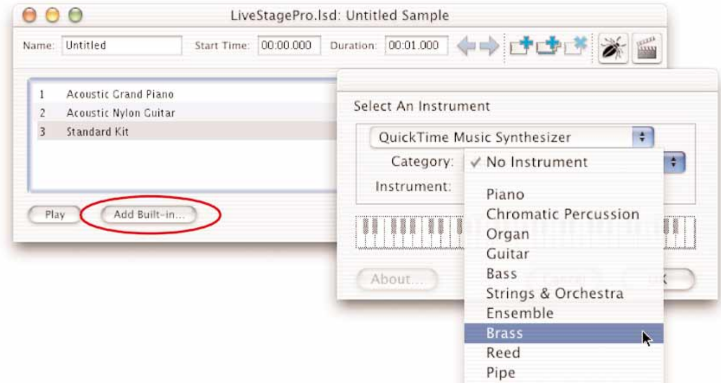
The Sample Properties window consists of a single field which houses the various instruments you want to include in the sample. Each instrument is assigned an Index Number when it is created. This Index Number is used to refer to the particular instrument in the QScripts written to play it. You can select and drag instruments to different locations in the list and thus alter this Index Number.

Add Built-In Button

Clicking the ‘Add Built-in’ button opens the Instrument Selection panel so you can preview and add instruments to the Instrument List.

Play Button

Select a MIDI instrument in the list and hit the Play button to hear a short melody, previewing its sound. You can also play a sampled audio file in the list in this way.



LiveStagePro.lsd: Untitled Sample

Name: Untitled Start Time: 00:00.000 Duration: 00:01.000

1 Acoustic Grand Piano
2 Acoustic Nylon Guitar
3 Standard Kit

Select An Instrument

QuickTime Music Synthesizer

Category: ✓ No Instrument

Instrument:

ADDING INSTRUMENTS

You can add QuickTime synthesizer instruments or sampled audio to provide the sounds of the instruments your QScript will play.

Default Synthesizer

When you select the Default Synthesizer, you are telling QuickTime to utilize the default MIDI device on the computer in which the movie is playing. This may be the standard QuickTime MIDI synthesizer or it may be an alternate MIDI sound card or external MIDI device such as a keyboard. Although using this option may provide a much nicer sounding playback of the MIDI instruments on the users computer, you have no guarantees as to what playback mechanism will be used. The result could be of poorer quality than what the QuickTime Music Synthesizer can produce.

Because the instruments in the QuickTime Music Synthesizer are built directly into QuickTime, they exist on the user's computer and therefore do not need to be physically within your project—potentially a huge savings in file size.

QuickTime Music Synthesizer

The QuickTime Music Synthesizer provides a wide range of MIDI instruments available for use in your projects. This set of instruments conforms to the standard set available for playback on any system that supports MIDI. These instruments are built into QuickTime itself and are made up of sampled instrument sounds and sound effects. Numerous different instruments in well over a dozen different categories provide everything from an 'Aah Choir' to a 'Harpsichord' to a 'Laser Gun'.

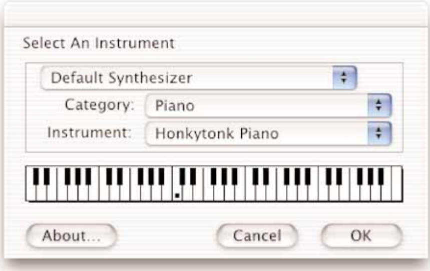
To add an Instrument from the Default or QuickTime Music Synthesizer:

1. Create an Instrument Track using one of the methods described.

2. Double-click on the Instrument sample in the Timeline to open the Instrument Sample Properties window.

3. Click on the 'Add Built-in' button.

   The Instrument Selection panel opens.



Select An Instrument

Default Synthesizer

Category: Piano

Instrument: Honkytonk Piano

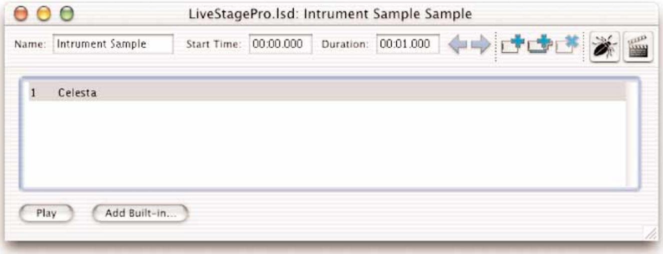
4. Change the Default Synthesizer to QuickTime Music Synthesizer.

5. From the Category menu, select the instrument category you would like to access.

6. From the Instrument menu, select the specific instrument you would like to add.

   To preview what the instrument will sound like when different notes are played, you can click on the keys on the piano keyboard below the Instrument menu. Note that some Instruments, such as "Drum Kits" will play a different effect depending on the note played.

7. Click OK.



LiveStagePro.lsd: Intrument Sample Sample

Name: Intrument Sample   Start Time: 00:00.000   Duration: 00:01.000

1 Celesta

Since sampled sounds are not built into QuickTime, they are stored within your movie. This will increase your file size accordingly.

The Instrument is added to the Instrument List.

8. Repeat steps 3-7 to add as many instruments to your list as you need.

Sampled Sounds

In addition to the provided library of MIDI instruments, you can also use your own sampled sounds to be used just like the other instruments. This can come in handy when you have a particular sound you want to incorporate or you cannot find a sound you like in the MIDI Instrument list. These sampled sounds can be in the form of .aiff, .wav, or any other digital sound format that QuickTime recognizes.

tip

Although sampled sounds work well as instruments, QuickTime imposes a 256KB size limit on sampled sounds. Thus, sound files larger than 256KB cannot be added to your Instrument List.

To add a sampled sound to the Instrument sample:

1. Double-click on the Instrument sample.

2. Drag the sampled sound's file from the Library or desktop directly to the Instrument List.

   The new sample appears in the list. It can now be used just like any built-in instrument.

Scripting the Instrument Track

Once you have assembled your Instrument List in the Sample Properties window, you can start to use them within your movie. You can play notes with specified instruments by calling the "PlayNote" QScript command using the syntax:

```
PlayNote(Instrument, Delay, Pitch, Velocity, Duration)
```

Instrument

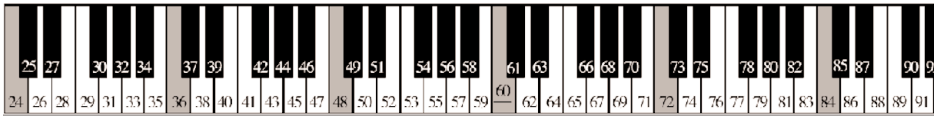
This is the Index Number assigned to the instrument in your Instrument sample.

Delay

The Delay parameter specifies the length of delay before the note starts playing, as based on the movie's time scale. For example, specifying 600 is equivalent to a one-second delay.

Pitch

The Pitch denotes which note—or frequency—the specified instrument should play. The frequency range is from zero to 127. The following graphic depicts the pitch values as they correspond to actual notes on a piano keyboard:



Velocity

In MIDI, velocity defines the note's volume, ranging from zero to 100.

Duration

Duration designates how long the note will be played, also according to the Movie's time scale.

TARGETING

To use the PlayNote command in a script for an Instrument Track, you need to target the specific track and, if necessary, a specific movie.

Example:

```
TrackNamed("Instrument 1").PlayNote(3, 300, 60, 100, 1200)
```

Using the third instrument (3) in the track named "Instrument 1" this plays a middle C (60) for two seconds (1200) at full volume (100) after a delay of half a second (300).

For more information on targeting in QScript see Chapter 16—Introduction to QScript, page 320.

THE TWEEN TRACK

The Tween Track is what is known as a 'data generating' track. Data generating tracks have no visual or audio representation, but instead generate numeric values that act on other tracks in your project. They can be used to create such effects as changing volume for an audio track, altering a VR's pan, or changing a sprite's location, shape, or image over time. Data generating tracks are attached to different tracks through the various 'Source' popup menus found in the Advanced tab of most Track Inspectors as well as throughout the Sprite Sample Properties window.

Tween Tracks produce numeric values derived from a set of start and end values while your movie is playing. Tween Tracks can take many different forms. For example, Matrix tweens can change a visual object's position, size, angle of rotation, or skew. Graphics Mode Tweens can be used to fade in or out a visual element, while Index/Layer/Flag Tweens can be used to create animations by changing a sprite's image over time. Tween Tracks operate on algorithms, adding minimal file size to your project.

Working with the Tween Track

The Tween Track can produce significant effects within your movies. However, there are a few caveats to its use.

- The visual media that the Tween Track creates is rendered on the fly on the user's machine and thus is CPU dependent. While you can create effects like a Video fade out with a relatively small file size, the results may be CPU-intensive for slower machines.

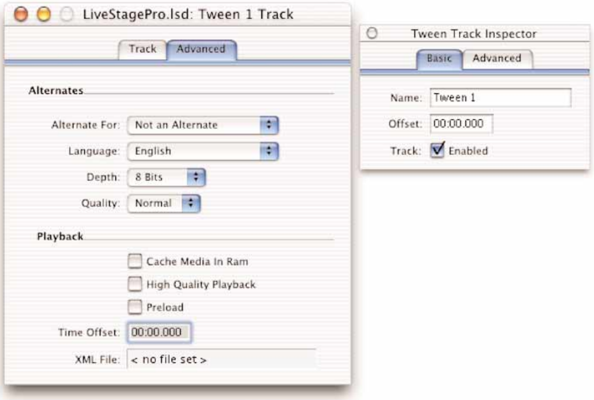
- The Tween Tracks generate data based on the passage of time, so it is tied to the movement of the Playhead.

- The Tween Track, when used as a source for an object's property, takes precedence over any other values attached to the property. That property can no longer be altered by any other means. For instance, if you add a Tween Track to a sprite's Draw Mode, you will not be able to alter the values that affect that sprite's Draw Mode, whether through the Properties tab or through QScript.

To create a Tween Track, click on the ⇔ , or select Movie>Create Track>Tween (Option-Command-W). You can also utilize the track contextual menu to create the Tween Track by control-clicking in the Track List area of the Timeline. The Tween Track appears in the Timeline with a single sample. You can have multiple samples in a Tween Track. However, the tweens contained in a given sample will only function while the Playhead is within that sample. You can repeat a tween by duplicating its sample.

THE TWEEN TRACK INSPECTOR/PROPERTIES WINDOW

The Tween Track Inspector and Track Properties window consist of the standard properties that are discussed in detail in Chapter 3—Working in the LifeStage Professional Environment, page 106. The Track Properties window provides a more comprehensive range of basic properties to set.



Alternates

Alternate For: Not an Alternate

Language: English

Depth: 8 Bits

Quality: Normal

Playback

☐ Cache Media In Ram

☐ High Quality Playback

☐ Preload

Time Offset: 00:00.000

XML File: < no file set >

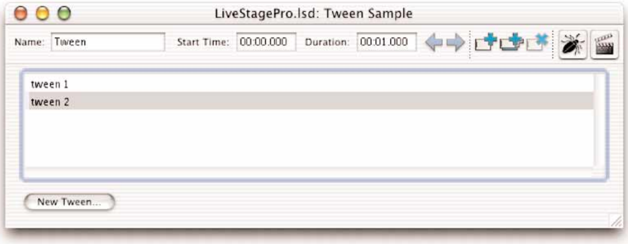
Name: Tween 1

Offset: 00:00.000

Track: ☑ Enabled

TWEEN SAMPLE PROPERTIES WINDOW

Double-click the Tween sample in the Timeline to open the Tween Sample Properties window. The Tween Sample Properties window is very similar to the Instrument Sample Properties window. Below the usual sample fields and tools is a single field that lists the tweens available in the sample. Tweens are created and then added to this Tween List. Once listed here, they can be attached to different tracks using that track's Source popup menu(s). You can alter the name of a tween within the list by clicking on it once to open the in-place editor. To delete a tween, select it and hit the delete key. Below the Tween List is the 'New Tween' button. Click this button to create a new tween in the current sample.



Name: Tween Start Time: 00:00.000 Duration: 00:01.000

tween 1

tween 2

TWEEN EDITOR

Clicking on the New Tween button in the Tween Sample Properties window opens the Tween Editor. This is where you select the type of tween and alter its parameters. The fields you see across the top of the Tween Editor window are common to all Tween types:

Name

The name field defaults to “Untitled” but you can change it to anything you like. The name you enter here will be displayed in the Tween List and Source popup menus. It is generally a good idea to provide a meaningful name so that you can differentiate one tween from another in the Source popup menus.

ID

The ID field is the unique ID for the tween. The only time you would ever use this is when you want to have the tween change from sample to sample. You would make the new tween in the next sample have the same tween ID. This new tween would be attached wherever the original tween is attached.

Start/End Time

Entering Start and End Times affects the results of the tween. The Start Time does not reflect the movie’s time, but is instead the Start Time for the tween within the Tween sample. For instance, if the Tween sample has a starting time of 00:01.000, and a tween within that sample has a Start Time of 00:01.000, the actual movie time at which the tween will start is 00:02.000.

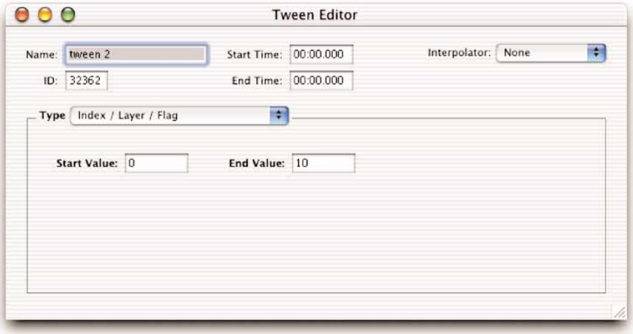
Interpolator

Tweens produce change smoothly and evenly, but what if you want to accelerate or slow a portion of your tween, as you would when easing in and easing out movement? Interpolators allow you to alter the tween, in effect ‘tweening the tween’, to change the way the tween’s time values are produced. Interpolators can produce results from a tween that would otherwise be difficult or impossible to produce. The Interpolator popup menu provides a list of tweens you have already created that can be used as interpolators for the selected tween.

Type

The Type popup menu provides a list of the different tweens you can create. When you select a tween type from this list, the field below the Type popup menu will change to reflect the properties you can alter for that tween. Some tween types provide pre-set fields where you can enter numeric values, while others provide a graphic approach to assigning tween values.

Index/Layer/Flag Tweens


Tween Editor

Name: tween 2
Start Time: 00:00.000
Interpolator: None
ID: 32362
End Time: 00:00.000
Type: Index / Layer / Flag
Start Value: 0
End Value: 10

The Index/Layer/Flag tween generates sequential values over time. This tween is used most often to alter the source of a sprite's image or to modify its layer or visible properties. The values produced by the Tween when it is attached to an Image Index Source property, for example, will change a sprite's image index, and therefore its image, to the index numbers provided by the tween.

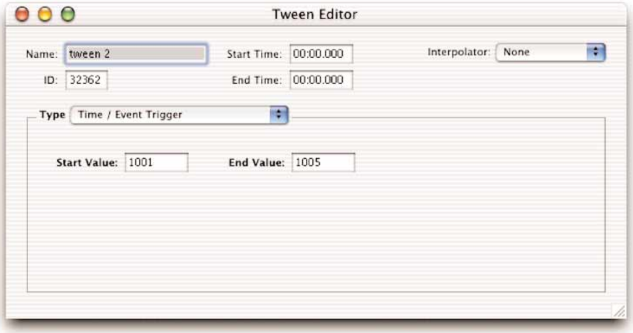
The Index/Layer/Flag tween produces values calculated from the Start and End values that you supply over the Duration that you designate. If no Start and End values are specified in the Tween, the Tween sample's duration will be the tween's duration.

Example: Start Time = 00:00.000
End Time = 00:05.000

Start Value = 0
End Value = 10

When the Tween sample begins to play, this Tween executes, producing values smoothly and incrementally starting from zero and ending at ten by the time the Tween sample reaches five seconds into playback. At 00:02.000, the tween's value would be four, at 00:02.300 (two and one half seconds) the tween's value would be five, at 00:05.000, the tween's value would be ten. If the sample continues to play, the tween's value will remain at ten.

Time/Event Trigger Tweens


Tween Editor

Name: tween 2
Start Time: 00:00.000
Interpolator: None
ID: 32362
End Time: 00:00.000
Type: Time / Event Trigger
Start Value: 1001
End Value: 1005

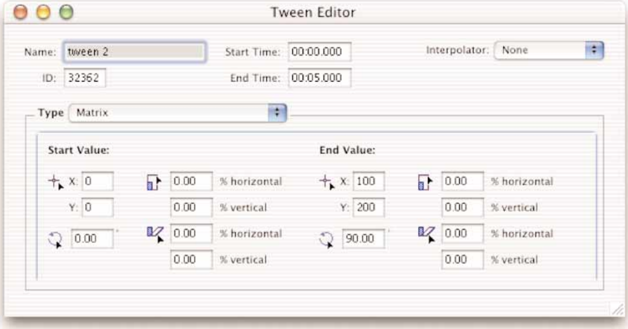
Use the Time/Event Trigger tween when you want to execute one or more script events over time. Like the Index/Layer/Flag tween, the Time/Event Trigger tween produces its values based on a single Start Value and End Value. These values can specify time values or event IDs that can trigger events in a sprite.

Example:

```
Start Time = 00:12.000
End Time = 00:14.000

Start Value = 1001
End Value = 1005
```

The Tween will produce a value of 1001 for 12 seconds, then over the following two seconds, it will produce 1002, 1003, 1004, and 1005. It will continue to produce a value of 1005 for as long as the sample continues to play. These values can be, for example, Custom Event IDs (see Chapter 16—Introduction to QScript, page 313) that will be triggered and will execute different script actions when the Tween produces the number.

Matrix Tween



Tween Editor

Name: tween 2
ID: 32362
Start Time: 00:00.000
End Time: 00:05.000
Interpolator: None

Type: Matrix

Start Value:
X: 0 | 0.00 % horizontal
Y: 0 | 0.00 % vertical
0.00 | 0.00 % horizontal
0.00 % vertical

End Value:
X: 100 | 0.00 % horizontal
Y: 200 | 0.00 % vertical
90.00 | 0.00 % horizontal
0.00 % vertical

The Matrix tween is used to alter the spatial properties of a sprite. Through a Matrix Tween, you can animate a sprite’s position, skew, rotation, and scale over a designated time.

Example:

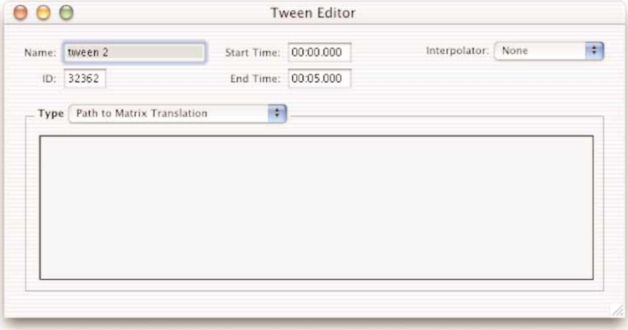
```
Start Time = 00:00.000
End Time = 00:05.000

    Start Value:
    X = 0
    Y = 0
    Rotation = 0.00

    End Value:
    X = 100
    Y = 200
    Rotation = 90.00
```

tip

Though a sprite is often positioned at X=0, Y=0 by default, if you want the sprite's start position to be X=0, Y=0, you still need to physically set the zeros in the X and Y fields in the tween. Otherwise, the tween may produce unexpected results.

When the Tween sample begins to play, the sprite using this tween as its Matrix Source will begin to move from the top left corner of the Stage to the right and down as it rotates. When the Tween sample reaches five seconds, the sprite will be rotated on one side, positioned at 100 pixels from the left edge and 200 pixels from the top edge.

Path to Matrix Translation



Tween Editor

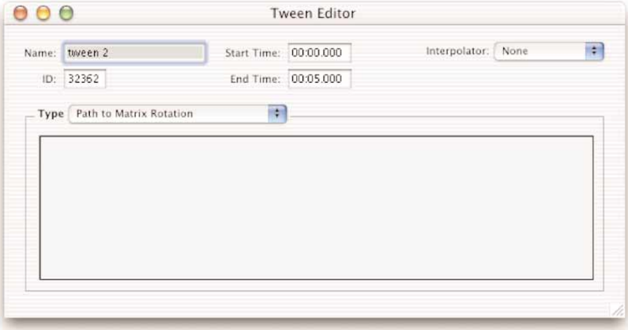
Name: tween 2
Start Time: 00:00.000
Interpolator: None
ID: 32362
End Time: 00:05.000
Type: Path to Matrix Translation

The Path to Matrix Translation tween allows you to create a path for an object to move along over time. The Start and End Times will determine the duration of the movement while the path you create will determine where the object will move. Click on the Stage in the Tween Editor to create the path. A red square appears at any newly-created point. Once two or more points have been created, the first point will become a black square. Intermediate points along the path will be black circles.

- To change the path, click on any of the points and drag them to new locations.
- Selected points are shown in red.
- Option-click a point to smooth a curve. Option-click the point again to remove the smoothing.
- Hold down the shift or Command key to select more than one point, which can then be moved simultaneously.
- To plot a smooth path, hold down the option key (Mac) or the CTRL-ALT keys (Windows) and start clicking to create the line. Bezier points will appear between click points that you can then drag to alter the curve.
- Select a group of points by clicking on the starting point, holding down the Shift key, and then clicking on the last point.

- You can modify the Tween Editor Stage's grid settings and the view through the Layout and View menus (see Chapter 3—Working in the LiveStage Professional Environment, page 70).

Path to Matrix Rotation


Tween Editor

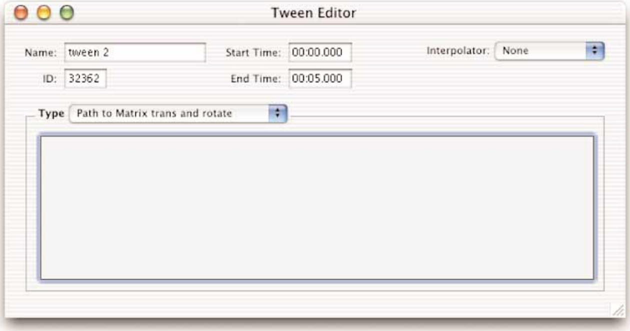
Name: tween 2
Start Time: 00:00.000
Interpolator: None
ID: 32362
End Time: 00:05.000
Type: Path to Matrix Rotation

tip

Tween points can be tricky. If a point you are trying to remove does not delete at first when you select it and hit the delete key, try moving it slightly and hitting delete again.

The Path to Matrix Rotation tween can be used to rotate a tween around its registration point. The path is created in the same manner as the Path to Matrix Translation tween. The object will appear tangent to its current position on the path.

Path to Matrix Translation and Rotation


Tween Editor

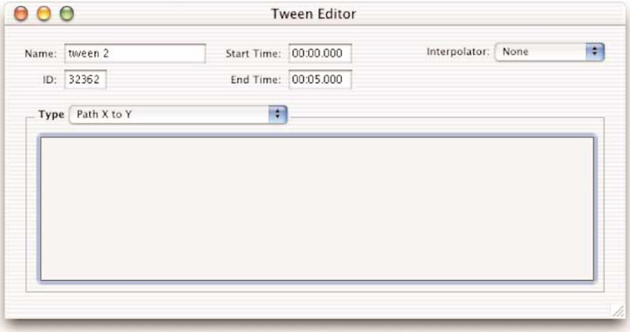
Name: tween 2
Start Time: 00:00.000
Interpolator: None
ID: 32362
End Time: 00:05.000
Type: Path to Matrix trans and rotate

tip

A diagonal line at 45 degrees from the top left to the bottom right will create an interpolator that does nothing. The current time is plotted along the X axis, and the tweened time is read from the Y axis. For a 45 degree line, these values would always be the same.

The Path to Matrix Translation and Rotation tween is a combination of the Path to Matrix Translation and the Path to Matrix Rotation. Click in the Tween Editor Stage to create a path for the object to follow. The object will rotate while moving along the path. It will follow the contours of the path much like a car driving on a winding road. The procedure to create this tween is the same as the Translation and Rotation tweens.

Path X to Y


Tween Editor

Name: tween 2
Start Time: 00:00.000
Interpolator: None
ID: 32362
End Time: 00:05.000
Type: Path X to Y

tip

The Stage is a rough means of plotting your tween and is not designed for precise placement. The shape of your path provides the interpolator's influence, so you may need to experiment and test to achieve your desired results.

This tween is used as an interpolator, allowing you to adjust the timing parameters of another tween. One practical use of the Path X to Y interpolator is to "Ease In" or "Ease Out" a movement. Instead of a quick and jerky start, easing in a tween starts the movement slowly and then increases the speed, easing out is just the opposite. This generates a more natural movement.

The values in a Path X to Y tween are set graphically as opposed to numerically in pre-set fields. This allows a lot more flexibility in constructing the tween. The Stage in this case is just a background where you plot the values of your interpolator tween. The tween's duration in time is plotted along the Stage's X-axis and the values are plotted along the Y-axis. The value range of the Y-axis starts at the top and increases towards the bottom.

You can add as many different points to the Path X to Y tween as you like. End points are denoted as black squares, and mid points along the path are circles. To change the position of a point, click on it. The point turns red and can now be dragged to the desired location. To delete a point, click it and hit delete.

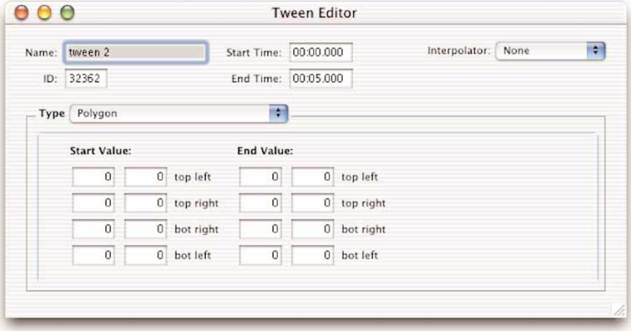
Path Y to X

This tween is similar to the Path X to Y tween except that the axis values are switched. With this tween, the current time runs along the Y-axis and the values are generated along the X-axis. As with the Path X to Y, this tween is used strictly as an interpolator.

tip

A good way to start off when designing a Polygon tween is to set each of the Start Values to the existing X, Y coordinate positions of the sprite, then increase all the End Values proportionally. This will give you a simple "zoom in" effect. You can then modify the End Values to achieve the effect you want.

Polygon

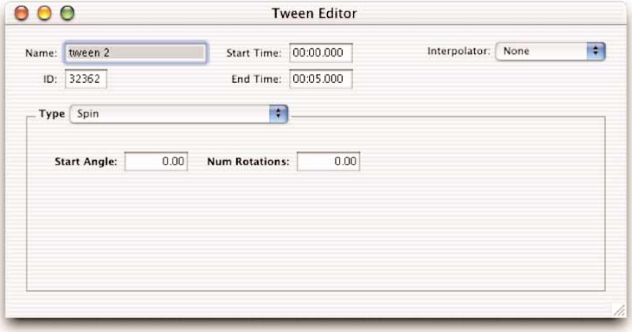


Name: tween 2
Start Time: 00:00.000
Interpolator: None
ID: 32362
End Time: 00:05.000
Type: Polygon

Start Value:			End Value:		
0	0	top left	0	0	top left
0	0	top right	0	0	top right
0	0	bot right	0	0	bot right
0	0	bot left	0	0	bot left

This tween is used to stretch and distort an object, as if you were pulling on its corners. Set the start coordinates for each of the corners of the object you wish to tween and set the End coordinates for the full distortion. The object will morph to the distorted coordinates over the time set in the Start and End Time fields.

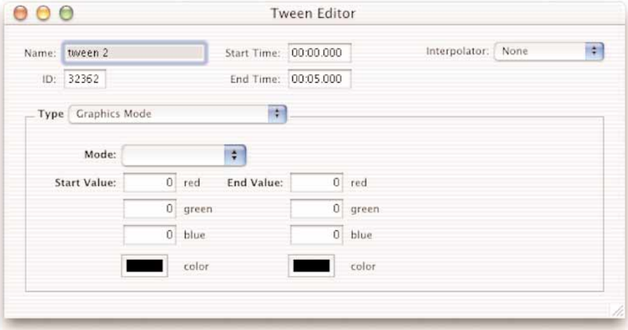
Spin



Name: tween 2
Start Time: 00:00.000
Interpolator: None
ID: 32362
End Time: 00:05.000
Type Spin
Start Angle: 0.00
Num Rotations: 0.00

The Spin tween does just that—it spins an object. Specify the starting angle and number of rotations for the object, as well as the Start and End Time. The object will rotate clockwise if the number of rotations is positive, counterclockwise if negative. The object will rotate around its top left corner. However, this is generally not a good tween to use for a sprite. A sprite using this tween will rotate around the top left corner of the Sprite Track, not the sprite itself. The Spin Tween is best used as a component of a Multi Matrix Tween.

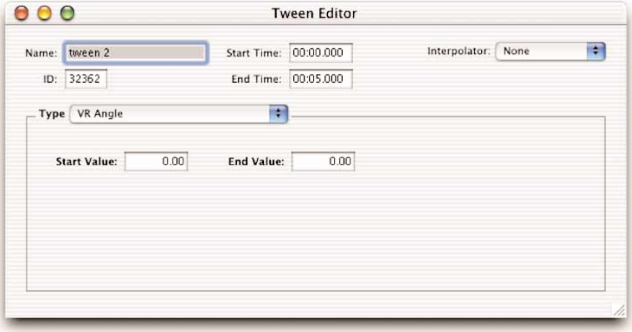
Graphics Mode Tween



Name: tween 2
Start Time: 00:00.000
Interpolator: None
ID: 32362
End Time: 00:05.000
Type Graphics Mode
Mode:
Start Value: 0 red
0 green
0 blue
color
End Value: 0 red
0 green
0 blue
color

Graphics Mode tweens can be used to alter the RGB color values and Draw Mode of an object. This tween can be useful for creating a smooth fade in or out of a sprite or visual track. You can also use this to change an object's color over time. As with other tweens, Start and End Times and Values are set to create the effect. This type of tween is applied through an element's Draw Mode Source popup menu.

VR Angle



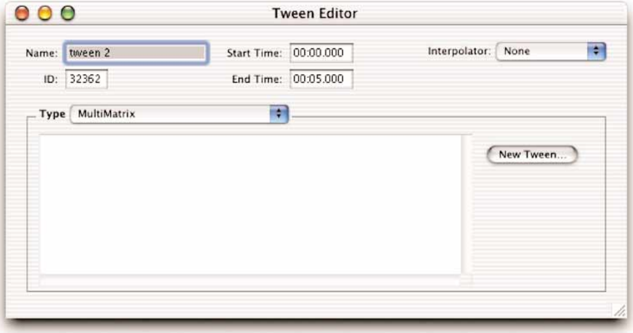
Name: tween 2
Start Time: 00:00.000
Interpolator: None
ID: 32362
End Time: 00:05.000
Type VR Angle
Start Value: 0.00
End Value: 0.00

The VR Angle tween allows you to specify a range of angles that you want a VR to move through in the designated duration. To move a VR to the right (Pan), down (Tilt) or to zoom in, enter the lower number in the End Value and the higher number in the Start Value. To move a VR to the left, up, or to zoom out, enter the higher number in the End Value and the lower number in the Start Value.

tip

The order of tweens in the MultiMatrix tween list can have a significant effect on the tweened object. Experiment with the list order to achieve the desired result.

MultiMatrix Tween



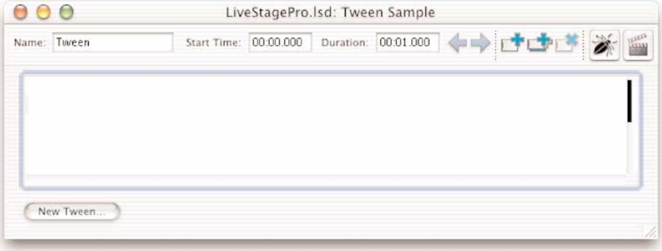
Name: tween 2
ID: 32362
Start Time: 00:00.000
End Time: 00:05.000
Interpolator: None
Type MultiMatrix

The MultiMatrix Tween allows you combine several matrix tweens into a single tween. You could incorporate a spin tween, a path tween, and a polygon tween to spin, move, and distort an object simultaneously. Clicking on the New Tween in the MultiMatrix Tween Editor allows you to add a tween to this tween in precisely the same way you would add a tween to the Tween sample. The only difference is that there are fewer tweens to select from in the Type popup menu. Double-clicking on any of the tweens in the MultiMatrix tween list will open the Tween Editor for that particular tween. To remove a tween from the MultiMatrix tween, select the tween and hit the delete key. To reorder the list, select a tween in the tween list and drag it to a new position.

Creating an Index/Layer/Flag Tween

1. Create a Tween Track by clicking on the ⇔ .

2. Double-click the Tween sample to open the Tween Sample Properties window.



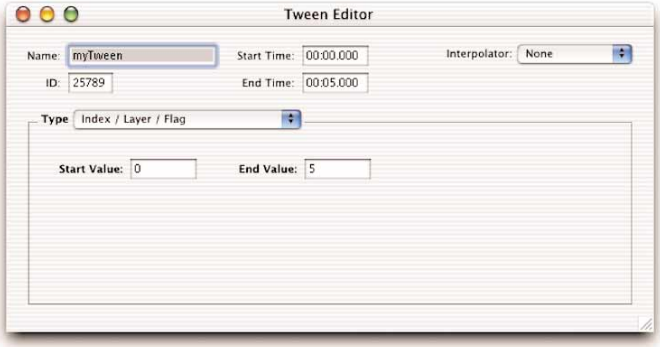
Name: Tween
Start Time: 00:00.000
Duration: 00:01.000

3. In the End Time field, enter 00:05.000 and then click on the 'New Tween' button.

   The Tween Editor opens.

4. Name the tween "myTween" and in the End Time field, enter 00:05.000.

5. From the Type popup menu, select Index/Layer/Flag.



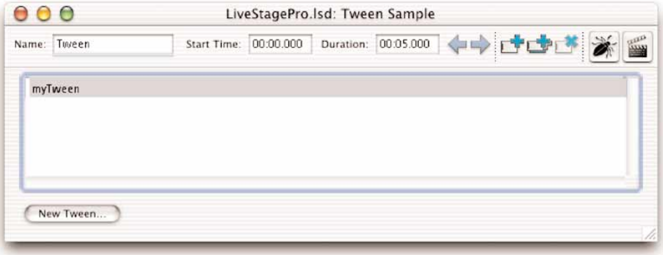
Tween Editor

Name: myTween
Start Time: 00:00.000
Interpolator: None
ID: 25789
End Time: 00:05.000
Type: Index / Layer / Flag
Start Value: 0
End Value: 5

6. Enter a Start Value of one and an End Value of five.

   Over a period of 5 seconds, “myTween” will produce the values 1, 2, 3, 4, 5. When “myTween” is attached to a track or sprite through a Source popup, the element controlled by the Source will use these values to determine its properties.

   Close the Tween Editor. “myTween” appears in the Tween List of the Sample Properties window.



LiveStagePro.lsd: Tween Sample

Name: Tween
Start Time: 00:00.000
Duration: 00:05.000

myTween

7. Close the Tween Sample Properties window.

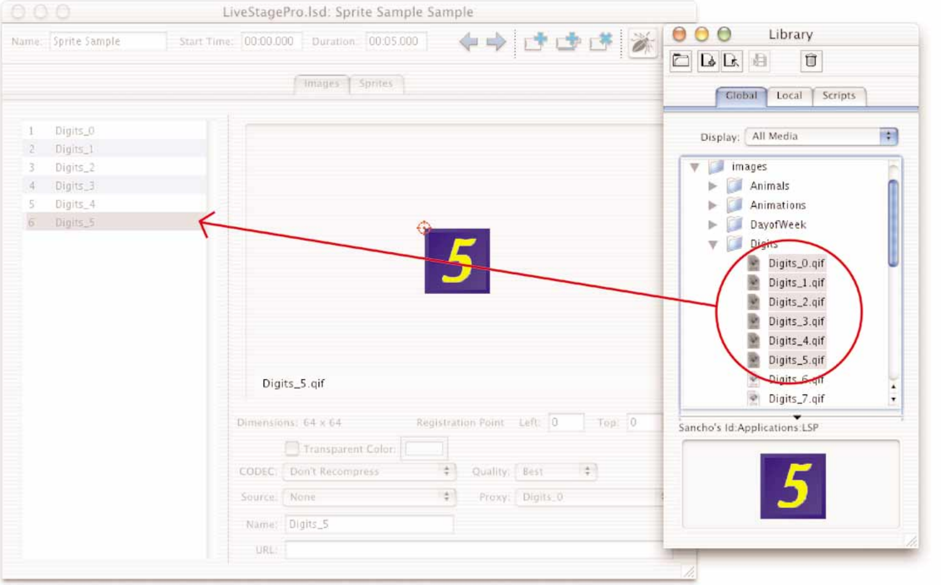
ADDING A TWEEN TO A SPRITE

1. Create a Sprite Track and change its duration to match that of the Tween Track by clicking and dragging on the right edge of the Sample Bar in the Timeline.

2. Double-click the sprite sample to open the Sprite Sample Properties window.

3. In the Library window, open the Images folder.

4. From the Digits folder, drag Digits_1 through Digits_5 to the Images tab of your Sprite Sample Properties window.



LiveStagePro.lsd: Sprite Sample Sample

Name: Sprite Sample Start Time: 00:00.000 Duration: 00:05.000

1	Digits_0
2	Digits_1
3	Digits_2
4	Digits_3
5	Digits_4
6	Digits_5

Digits_5.qif

Dimensions: 64 x 64 Registration Point Left: 0 Top: 0

□ Transparent Color:

CODEC: Don't Recompress Quality: Best

Source: None Proxy: Digits_0

Name: Digits_5

URL:

Library

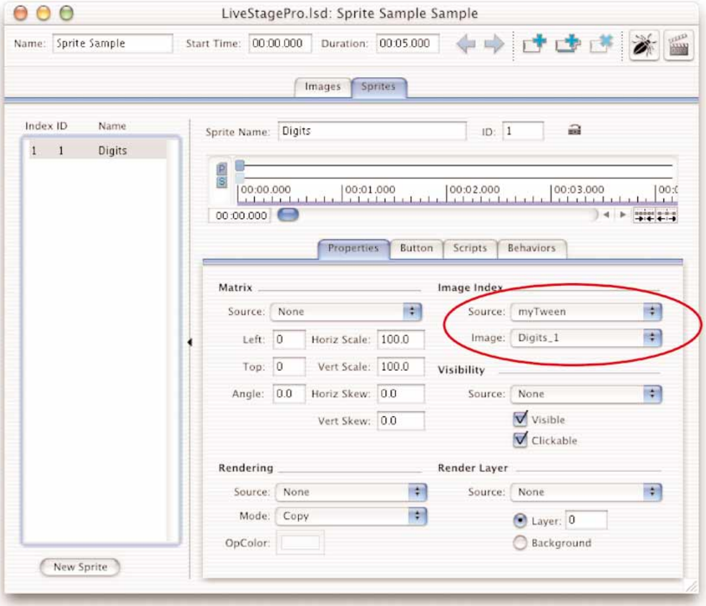
Display: All Media

Sancho's Id:Applications:LSP

The images are given index numbers from one through five.

5. Click the Sprites tab, and make a new sprite by clicking on the “New Sprite” button.

6. In the Image Index section, select “myTween” from the Source popup menu, and Digits_1 from the Image popup menu.



LiveStagePro.lsd: Sprite Sample Sample

Name: Sprite Sample Start Time: 00:00.000 Duration: 00:05.000

Index	ID	Name
1	1	Digits

Sprite Name: Digits ID: 1

00:00.000

Matrix
Source: None
Left: 0 Horiz Scale: 100.0
Top: 0 Vert Scale: 100.0
Angle: 0.0 Horiz Skew: 0.0
Vert Skew: 0.0

Image Index
Source: myTween
Image: Digits_1

Visibility
Source: None
☑ Visible
☑ Clickable

Rendering
Source: None
Mode: Copy
OpColor:

Render Layer
Source: None
● Layer: 0
○ Background

The sprite’s image index information will now be controlled by “myTween”. Whenever “myTween” produces a numeric value, the sprite’s image will change to that particular Index Number.

Example: “myTween” produces a value of four.

The sprite image with an Image Index of four, in this case Digits_4, will display.

7. Preview your movie by clicking on the 🎬.

   Press the Play button on the controller. The images progress from numbers one through five, over five seconds.

CHANGING THE TWEEN DURATION

1. Double-click the Tween sample to open the Sample Properties window and then double-click on “myTween” to open it.

2. Change the end time of “myTween” to 00:03.000.

   Close the Tween Editor and the Tween Sample Properties window.

3. Preview your movie.

   The images progress from number one through number five over three seconds, remaining at image number five until the movie ends.

ADDING AN INTERPOLATOR TWEEN

1. Using the project from the previous exercise, change the end time of the “myTween” tween back to 00:05.000 in the Tween Editor.

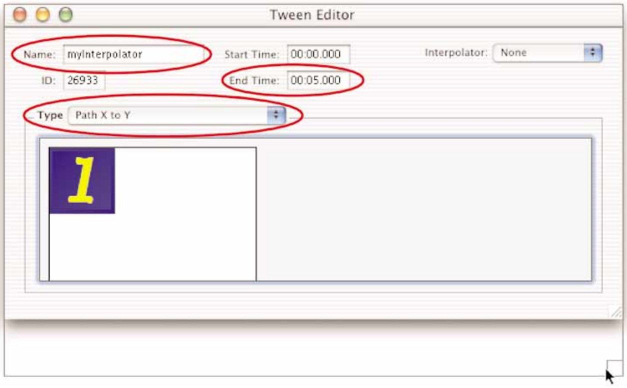
   Close the Tween Editor.

2. Click the New Tween button to create a new Tween.

3. Name the new Tween “myInterpolator” and set its End Time to 00:05.000.

4. From the Type popup menu, select Path X to Y.

   The Stage appears in the field below. Resize the Tween Editor window to view the Stage in its entirety by dragging the bottom right corner.



Tween Editor

Name: myInterpolator

ID: 26933

Start Time: 00:00.000

End Time: 00:05.000

Interpolator: None

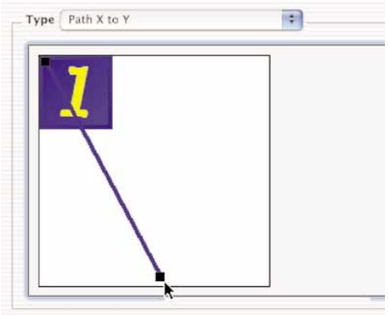
Type Path X to Y

5. Click at the top left corner of the Stage in the Tween Editor.

   A black square appears.

6. Click at the bottom of the Stage, halfway across.

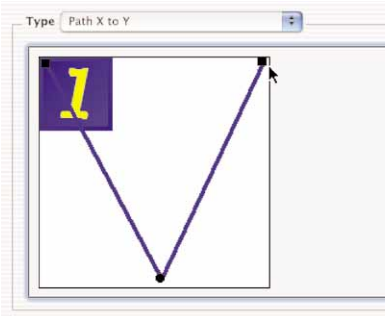
   Another black square appears, joined to the first by a bold blue line.



Type Path X to Y

7. Finally, click in the top right corner.

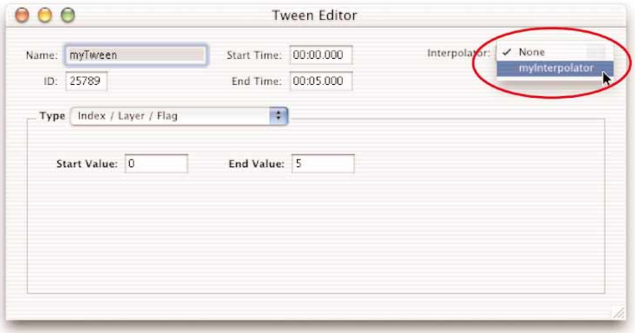
   The three points now are in the shape of a large ‘V’.



Type Path X to Y

8. Close the Tween Editor and open “myTween”.

9. From the Interpolator popup menu, select “myInterpolator”.



Tween Editor

Name: myTween

ID: 25789

Start Time: 00:00.000

End Time: 00:05.000

Interpolator: ✓ None / myInterpolator

Type: Index / Layer / Flag

Start Value: 0

End Value: 5

Close the Tween Editor.

10. Preview your movie.

    The numbers display in ascending order to the halfway point of the timeline, at which point they display in reverse order until the end of playback (1, 2, 3, 4, 5, 4, 3, 2, 1). If the ‘V’ in the “myInterpolator” Tween Editor were inverted, the numbers would display from the maximum to the minimum and then back to the maximum again over the duration of playback (5, 4, 3, 2, 1, 2, 3, 4, 5) .

    In order to understand what is happening here, notice that the ‘5’ image only appears for a brief moment; this is because the Interpolator tween only gives the maximum value on the vertex of the ‘V’ which has been drawn. To have the ‘5’ appear for a longer time, the path would have to be in the shape of a “U.”

11. Reopen the myInterpolator tween and drag the vertex of the ‘V’ well below the bottom margin of the Stage.

THE MODIFIER TRACK

The Modifier Track is also a data-generating track like the Tween Track, and like the Tween Track, Modifier Tracks produce numeric values over time. Unlike the Tween Track, Modifier Tracks provide values from a sequence of numbers stored within the track, as opposed to generating numbers through an algorithm based on Start and End values like the Tween Track. Modifier Tracks are used to create repeating sequences or to trigger timed events synchronized with the playback of the movie. For instance, if you want to animate a sprite that contains three images at precise intervals, you can create a Modifier Track that generates the values 1, 2, and 3 at a rate of five frames per second. You can then have this sequence repeat ten times. Use this Modifier Track as the Source for the sprite’s Image Index and the sprite will then display images 1, 2, and 3 in sequence at a rate of five frames per second, for ten repetitions. You could also create a more complex series of values such as: 2, 1, 3, 3, 2. You can have the Modifier generate several consecutive sequences to create animations that are more complex.

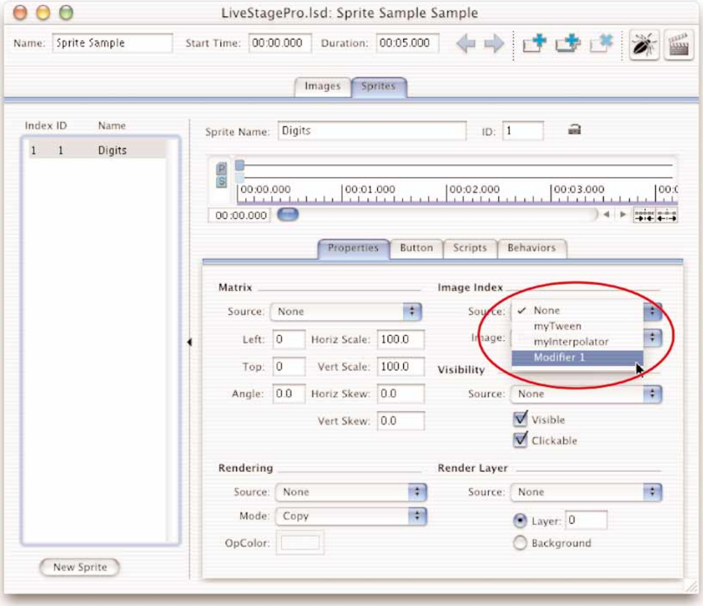
Modifier Tracks produce values that have been pre-set within the track and only have to generate these sequences according to time. The Modifier Track generates values, and executes events according to frame rates. This can make it a more accurate alternative to Idle Events that can produce varying results on different machines. Modifier Tracks can also be used as a more reliable alternative to Frame Loaded Events, which can conceivably be lost on the rare occasion that the frame containing the event is dropped during playback.

tip

When you use a Modifier or Tween Track to control a sprite's properties, that property is 'locked' to the Modifier/Tween and cannot be changed through a script. To work around this issue, you can disable and enable your Modifier or Tween Track as needed. See Chapter 4—Introduction to Tracks, page 122.

Working with the Modifier Track

As with the Tween Track, Modifier Tracks supply values to other tracks through their respective Source popup menus. For instance, if you create an Index/Layer/Flag Modifier Track to set the Image Index property for a sprite, in the Sprite Sample Properties window, you would select the Modifier Track from the Image Source popup menu.



Name: Sprite Sample
Start Time: 00:00.000
Duration: 00:05.000

Index	ID	Name
1	1	Digits

Sprite Name: Digits
ID: 1

Source: None
Left: 0 Horiz Scale: 100.0
Top: 0 Vert Scale: 100.0
Angle: 0.0 Horiz Skew: 0.0
Vert Skew: 0.0

Source: ✓ None
myTween
myInterpolator
Modifier 1
Image:

Source: None
☑ Visible
☑ Clickable

Source: None
Mode: Copy
OpColor:

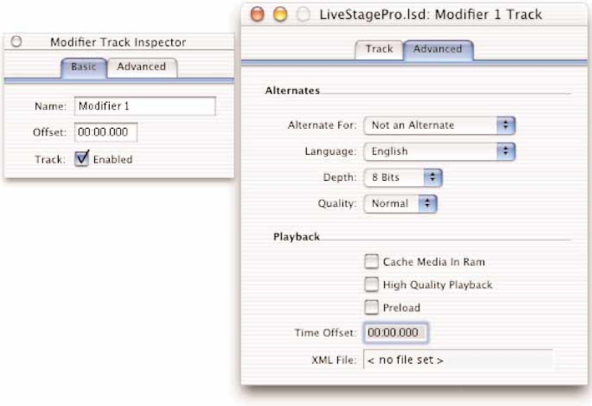
Source: None
● Layer: 0
○ Background

To create a Modifier Track, click on the or select Create Track>Modifier (Option-Command-N) from the Movie menu. Again, you can also use the track contextual menu to create the Modifier Track by control-clicking on the Track List area of the Timeline.

The Modifier Track contains a single sample that can house different sequences, provided they are all the same type of Modifier. In other words, you cannot have an Index/Layer/Flag modifier in the same Modifier Track as a VR Angle. Neither can you have multiple samples in a Modifier Track. All of the sequences within the Modifier sample will be applied one after the other to the Source that the Modifier Track controls. If you want different sequences to be applied to different Sources, or if you want to use different types of Modifiers, create separate Modifier Tracks. The sample duration cannot be altered through the Track Manipulation Tool. Instead, its duration adjusts to reflect the parameters that are set within the track.

MODIFIER TRACK INSPECTOR/PROPERTIES WINDOW

The Modifier Track Inspector and Track Properties window consist of the standard properties that are discussed in detail in Chapter 3—Working in the LiveStage Professional Environment, page 106. The Track Properties window provides a more comprehensive range of basic properties to set.



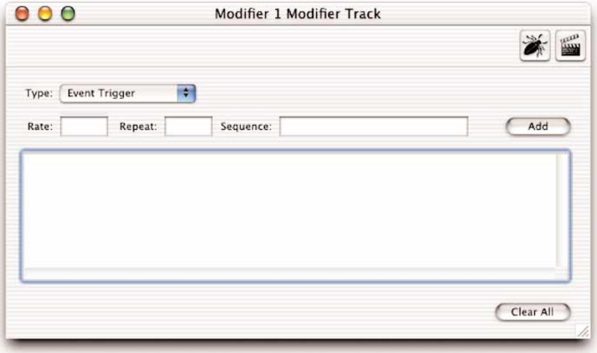
Name: Modifier 1
Offset: 00:00.000
Track: ☑ Enabled

Alternate For: Not an Alternate
Language: English
Depth: 8 Bits
Quality: Normal

□ Cache Media In Ram
□ High Quality Playback
□ Preload
Time Offset: 00:00.000
XML File: < no file set >

MODIFIER TRACK SAMPLE PROPERTIES WINDOW

As you can only have a single sample in the Modifier Track, you will not find the sample navigation options found in other Sample Properties windows. The Modifier Sample Properties window provides three fields and a Type popup menu similar to that of the Tween Track.



Type: Event Trigger
Rate:
Repeat:
Sequence:

Rate

The Rate specifies the speed in Frames per Second at which you would like the sequence to execute. Each value in the sequence correlates to an individual ‘frame’. Thus, if you have a sequence 1, 2, 3, 2, 1 and a Rate of four frames per second, in the course of one second the Modifier will have gone through 1, 2, 3, 2.

Repeat

This designates the number of times you would like the sequence to repeat.

Sequence

The Sequence is the numerical series of values you want to send to the associated object.

Type

The Modifier Track offers four different types to select from in this popup menu:

Event Trigger

Event Triggers are used to generate events in sprites. For example, you could create a Custom Event (see Chapter 16—Introduction to QScript, page 311), and then have that event execute at a set point in time:

Example:

```
First Sequence:
Type = Event Trigger
Rate = 1
Repeat = 10
Sequence = 0

Second Sequence:

Type = Event Trigger
Rate = 1
Repeat = 1
Sequence = 12345
```

After 10 seconds, the Modifier will generate the value “12345”. A Custom Event of ID “12345” will execute at this point (provided the Modifier is attached to the sprite containing the Custom Event).

Events are triggered when values change, so a Modifier simply containing the proper sequence will not be sufficient to execute the event in the sprite. The Modifier must change the values produced within the sequence in order for the event to be triggered. You can accomplish this by adding a zero before the trigger value in your sequences.

Example:

```
Rate = 1
Repeat = 1
Sequence = 0 12345
```

The Modifier Track generates a value of zero, followed by 12345. The value’s change to the Custom Event ID triggers the event.

Index/Layer/Flag

Use this Modifier when you want to change an object’s Index, Layer, or Flag value.

VR Angle

The values generated here can be used to pan, tilt, and zoom a VR Track. The Modifier Track is selected as the Source in the VR Sample Properties window.

tip

The duration of a Modifier Track depends on the Rate, Repeat, and Length of the sequence as follows: Duration = Length * Repeat / Rate. For example, if you have the sequence 0, 1, 2, 3, 4, 5 to be repeated 5 times with a rate of 3, the Modifier Track will have a duration of 10 seconds, while the sequence 0, 1, 2 repeated 6 times with a rate of 4 will have a duration of 4.5 (00:04.300) seconds.

Exercise: how long is a Modifier Track with the sequence 3 1 4 1 5 repeated 2 times with a rate of 5, followed by the sequence 2 7 1 8 2 8 repeated 3 times with a rate of 9 ?

(Answer: 2 + 2 = 4 seconds)

tip

There is a bug in QuickTime 4.1 with Windows that does not convert Custom Event Handler ID numbers to a format that Windows understands. This bug has been fixed in later versions of QuickTime. If, however, you must deploy a QuickTime 4.1 project, use any of these event numbers to avoid this bug: 65792, 13154, 197376, 263168, 328960, 394752, 460544, 526336, 592128, 657920, 723712, 789504, 855296, 921088, 986880

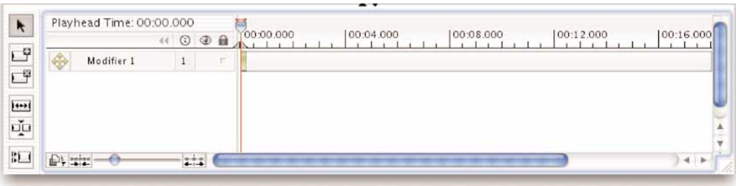
Volume/Balance

Set the volume and/or balance settings for an audio track with this type of Modifier. The range of values is from -256 to +256 for volume and -128 (full left) to +128 (full right) for balance.

CREATING A MODIFIER

1. Create a Modifier Track in one of the ways that have been previously described .

   A Modifier Track with a small sample appears in the Timeline.



2. Double-click the Modifier sample to open the Modifier Sample Properties window.

3. Specify the type of sequence you wish to create in the Type popup menu.

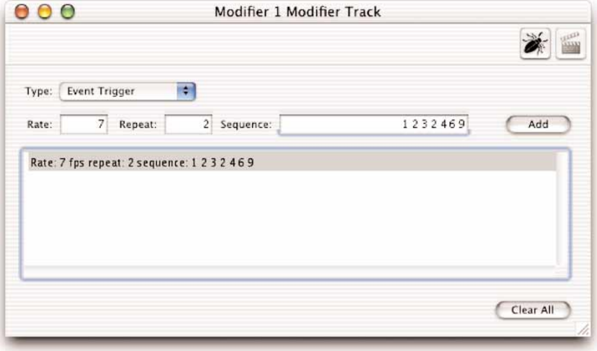
4. Enter the rate (in frames per second) for the sequence in the Rate field.

5. Enter how many times you want the sequence to repeat in the Repeat field.

6. Enter the sequence of numbers you want the Modifier Track to generate.

7. Click the Add button.

   The sequence is added to the list.

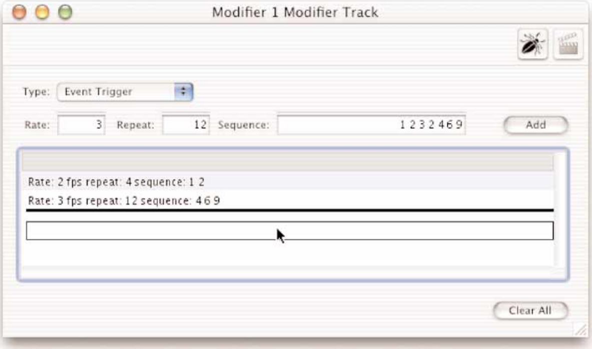

Modifier 1 Modifier Track

Type: Event Trigger

Rate: 7 Repeat: 2 Sequence: 1 2 3 2 4 6 9

Rate: 7 fps repeat: 2 sequence: 1 2 3 2 4 6 9

To edit a sequence, click on it in the list and edit the values in the fields, without clicking the Add button. To delete a sequence, click on it in the list and hit the delete key. To delete all sequences in the list simultaneously, hit the Clear All button. To reorder sequences, select the sequence and drag it to a new position in the list.



Type: Event Trigger

Rate: 3 Repeat: 12 Sequence: 1 2 3 2 4 6 9

Rate: 2 fps repeat: 4 sequence: 1 2

Rate: 3 fps repeat: 12 sequence: 4 6 9

ADDITIONAL TRACKS WRAP UP

The various tracks covered in this chapter range from simple to advanced and, while used less often than many of their other track counterparts, can still have significant effects on your projects. On the LiveStage Professional 4.0 CD-ROM, in the Project Samples folder, you can find completed QuickTime movies and associated projects that demonstrate some of these tracks in action. In the “Featured Demos” folder, you can explore some of the different effects available to you through the Effects Track. In the “Small Hands” folder, you can see the Tween and Modifier Tracks used in conjunction to create a simple animation. Additionally, you will find some project samples illustrating Tween, Modifier, and Effects tracks on Totally Hip’s LiveStage Developers Network site at http://www.totallyhip.com/lsdn).

Chapter 14

Using Multiple Tracks

OVERVIEW

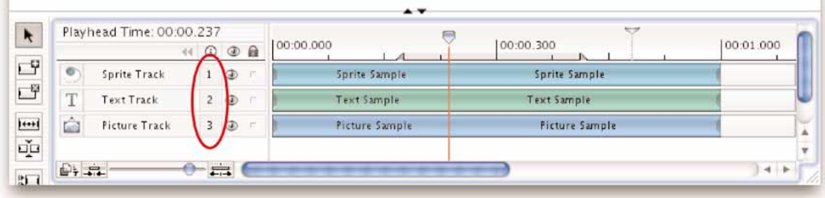
Now that you have been introduced to the various track types that comprise LiveStage Professional, you can begin to combine them into multiple-track projects. Multiple track construction is the foundation of LiveStage Professional authored movies. Although multiple-track movies are easy to create in LiveStage Professional, understanding the fundamentals of their construction will help you avoid potential pitfalls, save time, and build optimized movies.

INDEX NUMBERS AND DRAWING LAYERS

Tracks are ordered within QuickTime according to two parameters: the Index Number and the Drawing Layer. The Index Number determines the order in which tracks are loaded into the QuickTime movie, while the Drawing Layer determines how a particular visual element appears in relation to another, whether in the foreground or in the background. It is important to consider both points when you are constructing your projects. While the Drawing Layer will be visually obvious on the Stage, the Index Number should also be considered when authoring, as it can directly affect your movie's playback.

INDEX NUMBER

Each track is assigned an Index Number in ascending order according to when it was created. The Index Number determines the order in which a track will load into QuickTime. The track with an Index of '1' will be loaded into memory first, followed by the track of Index '2' and so on.



You can alter a track's Index by selecting and dragging the track to a new location within the track list while in Load Order View (see Chapter 3—Working in the LiveStage Professional Environment, page 86).You will find hints on optimizing the load order of your tracks in the "Optimizing" section, later in this chapter.

tip

Since the range of Drawing Layer values is so great, it is a good idea to spread out the values you assign, in case you decide you want to move layers around. It is easier to move a layer between one with a value of 100 and another of 110, than it is to move a layer between one with a value of 100 and another with 101.

DRAWING LAYERS

Drawing Layers are numerical values that determine whether visual tracks will appear in front of or behind one another. Drawing Layer values are assigned to tracks automatically in descending order as each track is created. The most recently created track is assigned the lowest number, appearing in the movie's foreground. The range of Drawing Layer values in LiveStage Professional runs from -9999 to 99999. It is important to note that Drawing Layers with negative numbers appear in front of those with positive numbers. For scripting purposes, QuickTime supports values ranging from -32767 to 32768. Drawing Layers greater than 32700 and less than -32700 have been reserved for FastTracks™. Moving tracks outside of this range should be done with care.

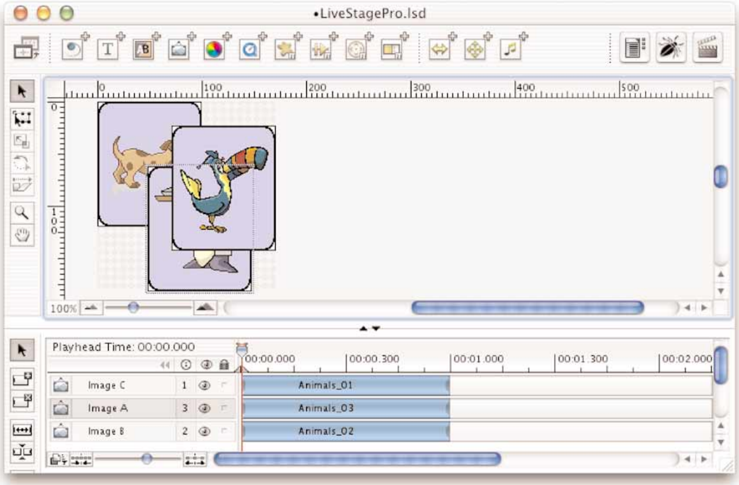
A

B

C

Example: Image A has a Drawing Layer of 0
Image B has a Drawing Layer of 1
Image C has a Drawing Layer of -1

Image C will be in the foreground. Image A will appear behind Image C, and Image B will be in the background.



You can change the value of a track's Drawing Layer through its Inspector (See Chapter 3 – Working in the LiveStage Professional Environment, page 106 to learn how to use the Inspector) or through scripting (See Chapter 16 – Introduction to QScript, page 309).

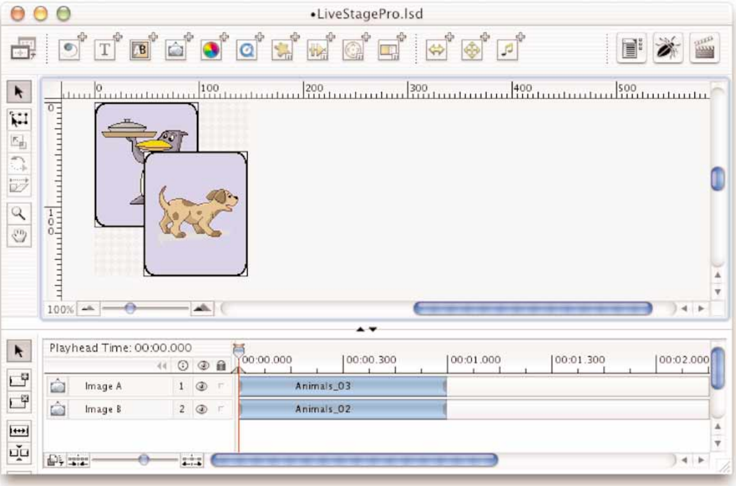
When multiple tracks have the same value for the Drawing Layer, the Index Number of the track will break the tie. The higher the Index Number, the closer the track is to the foreground.

A

B

Example: Image A has a Drawing Layer of 0 and is at Index Number 1
Image B has a Drawing Layer of 0 and is at Index Number 2

Image B appears in front of Image A



You can view your tracks by Index Number, by Layer, or by your own custom order by changing the Timeline View (Chapter 3 – Working in the LiveStage Professional Environment, page 86).

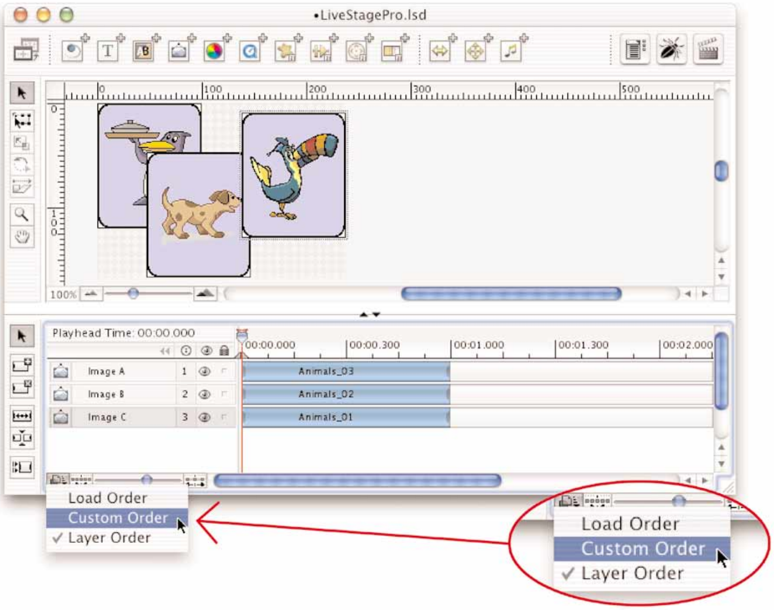
GROUPING TRACKS

[New in LiveStage Professional 4.0!]

In the Custom Timeline View, you can arrange tracks without affecting their Index Number or Drawing Layer. You can organize tracks according to whatever criteria you decide. In addition, you can now group tracks within the Timeline or the Stage to better organize your view, as well as manipulate multiple tracks simultaneously.

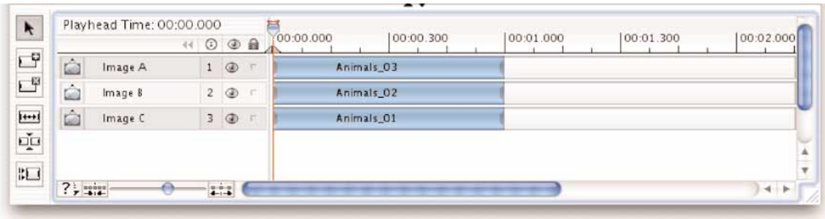
To group a track:

1. Select Custom View from the Timeline View options.



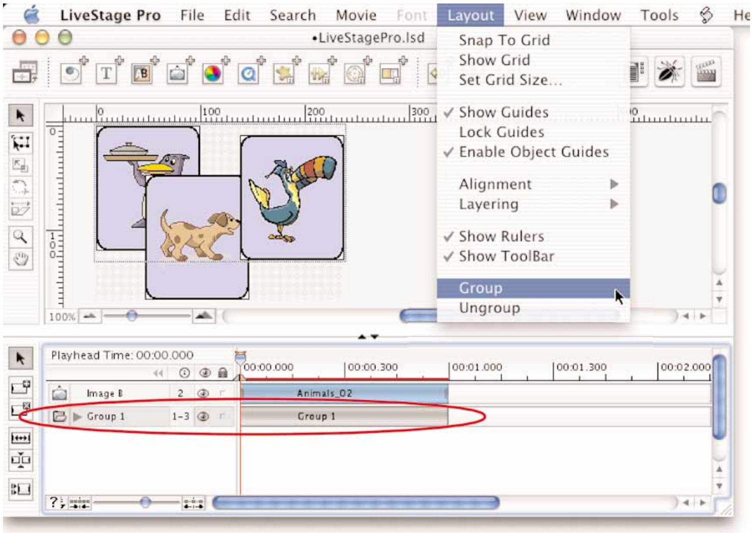
2. Select the tracks in the Timeline you want to group.

   Shift-click to select consecutive tracks, Option-Command-click to select non-consecutive tracks.



3. From the Layout Menu, select Group.

   The selected tracks are no longer visible in the Timeline and a single Group track appears.



4. Click the expansion triangle next to the Group track title to expand the track and view the grouped tracks.

Once tracks are grouped, you can adjust different parameters that will affect all the tracks included in the group.

- Use the Track Manipulation tool to change the Group Track's offset along the Timeline. With the Track Manipulation tool selected, click on the Group sample and drag it to a new point in the Timeline. All the grouped tracks will now be offset to this time. You will notice that there is no means to alter the Group Track's overall duration. Track duration remains independent of the Group, so you can adjust individual track duration without affecting other grouped tracks.

- Parameters set through the Group Track Inspector affect all the tracks within the group. You can alter the dimensions and position of the tracks, their Draw Mode, along with a number of other options explained in detail in Chapter 3—Working in the LiveStage Professional Environment, page 106.

tip

Be careful when grouping tracks that already have specific properties set. Any changes you make to the group track will overwrite the original settings of all tracks within the group.

- You can modify the name of a Group Track by clicking on its title in the Timeline and editing it in place.
- To ungroup a Group, click on the Group Track and select Ungroup from the Layout Menu.
- To delete a Group, select the Group Track and hit Delete. This removes all the tracks contained in the Group from your Timeline. Delete an individual track from the Group by selecting it (while the Group Track is expanded) and hitting the delete key.

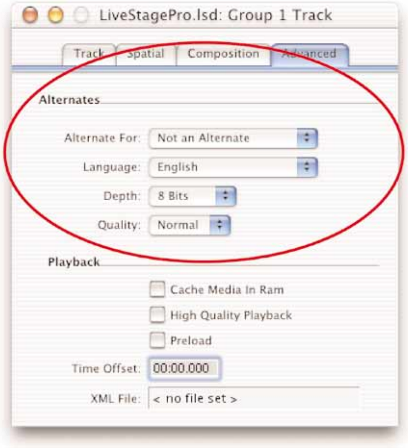
ALTERNATE TRACKS

Alternate Tracks can extend the reach of a project tremendously. Alternate Tracks allow you to deliver a tailored experience to each segment of your audience. You can author a single movie to provide an English version to your English-speaking audience and a French version to your French-speaking audience. If your target audience will have a range of CPU capacities and monitor resolutions, you can provide the best possible audio or visual experience for users at one end of the spectrum, without sacrificing quality for those at the other. Instead of creating several different movies to cover the range of audience you will have, you can author a single movie with tracks containing the alternate information. You can then script buttons that allow your users to select the presentation that best suits their system and language preferences, or QuickTime will seamlessly provide the appropriate track based on preferences set when the user first installed QuickTime.

Any track can have an Alternate Track. For example, you can have an English Text Track that has a German Alternate Text Track, and that German Track could have a Spanish Alternate Track. Each Alternate Track must, in turn, have an Alternate Track, and the entire sequence must be set up in a loop. In other words, the first track will be the alternate for the second; the second will be the alternate for the third, and so on. Finally, the last Alternate Track in the sequence must be the alternate for the first track. This way, each track has an alternate.

Example: English > German >
German > Spanish
Spanish > English

Alternate Tracks are specified in the Advanced tab of a track's Track Properties window. To open the Track Properties window, double-click on the Track Header.



LiveStagePro.lsd: Group 1 Track

Alternates

Alternate For: Not an Alternate

Language: English

Depth: 8 Bits

Quality: Normal

Playback

□ Cache Media In Ram

□ High Quality Playback

□ Preload

Time Offset: 00:00.000

XML File: < no file set >

ALTERNATE FOR

Through this popup menu, select the track for which you want the current track to be an alternate. If this track is a better match for the user's system, then it will be used instead.

LANGUAGE

Select the language of the current track. This track will be displayed when the user's language setting matches the selection in this popup.

DEPTH

Select the bit depth of the current track. The track that comes closest to the bit depth of the user's monitor will be displayed. For example, if your audience is likely to include those with older monitors, you might want to provide a track that conforms to 256 colors. QuickTime will automatically select the best quality track that will work on the user's system. This setting does not change your track, it merely describes it to QuickTime.

QUALITY

If multiple alternates fit the requirements of the user's system, this selection will determine the track that is displayed. QuickTime will select the best quality first. Again, this setting does not change your track, it merely describes it to QuickTime.

MEDIA SYNCHRONIZER

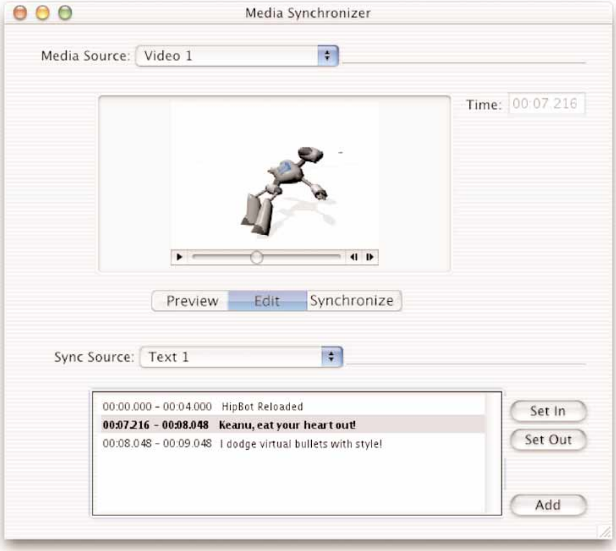
[NEW in LiveStage Professional 4.0!]

The new Media Synchronizer makes synchronizing images and text to audio and video streams quick and easy. Synchronizing media uses the standard Non-Linear Editor in/out metaphor, and incorporates shortcuts that video editors will find familiar.

You can use the Media Synchronizer to synchronize PowerPoint Slides to audio or video presentations for deployment to the Web or CD-ROM. You can also use it to quickly and easily add text captioning in multiple languages, all perfectly matched up to the audio/video content.

Working with the Media Synchronizer

The Media Synchronizer, accessed through the Movie Menu, has a straightforward interface.



Media Source: Video 1

Time: 00:07.216

Sync Source: Text 1

```
00:00.000 - 00:04.000   HipBot Reloaded
00:07.216 - 00:08.048   Keanu, eat your heart out!
00:08.048 - 00:09.048   I dodge virtual bullets with style!
```

MEDIA SOURCE

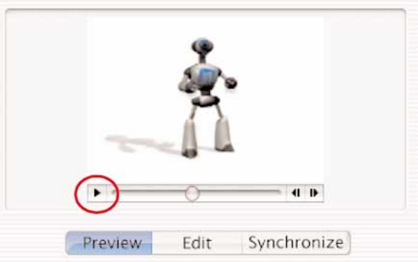
In this popup menu select the Video or Audio source with which you would like to synchronize your Picture or Text samples. If the source movie contains both a video and an audio track, there will be additional entries for these tracks so you can have both the audio and video play at the same time.

tip

If your Media Source Video was imported into LiveStage Professional with a Sound Track, you can play both tracks simultaneously while using the Media Synchronizer. Simply select Video Track 2 (Video + Sound) from the Media Source popul menu.

MEDIA SOURCE PREVIEW

The Media Source Preview displays the track(s) selected in the Media Source popup menu. To play the preview, hit the play button on the QuickTime controller.



TIME

This displays the source’s current playback time in Minutes:Seconds:Ticks.

PREVIEW BUTTON

The Preview button allows you to view how your video/audio source corresponds to its synchronized elements.

- Click the Play button in the Media Source Preview. Samples will be highlighted as their synchronized points come up in the course of the playback.

- Click on a sample to jump the video/audio to its synchronized point.

EDIT BUTTON

When this button is selected, the Media Synchronizer enters Edit Mode, which enables you to set start and end times for your synchronized source samples.

- Click on a Synchronized Source sample, then move the video/audio source to the desired ‘in’ or ‘out’ point for the sample, and click ‘Set In’ or ‘Set Out’.

SYNCHRONIZE BUTTON

The Synchronize mode allows you to set in and out points as the Media Source Preview plays.

1. Click the Synchronize Button to reset the media to be synchronized. Click the Play Button in the Preview.

2. Click ‘Set In’ (or press the letter ‘I’ on your keyboard) to set the selected sample’s in point. Click ‘Set Out’ (or press the letter “O” on your keyboard) to set the selected sample’s out point.

   Once an In Point is set, the next sample will automatically be selected. The subsequent setting of the In Point will apply to this sample. Continue to click on “Set In” until all samples are set. If you do not click “Set Out”, then the Out Point of the sample will be immediately preceding the In Point of the following sample.

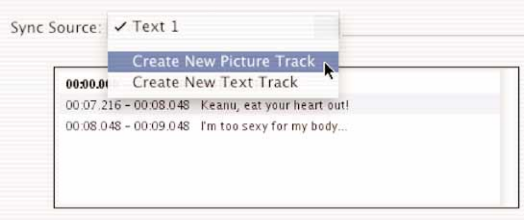
SYNC SOURCE

Select the Picture or Text Track you want to synchronize with the Video or Sound source from this popup menu. You can also create a new Picture or Text Track directly from the Media Synchronizer through this menu.

CREATING A NEW SYNC SOURCE PICTURE TRACK

You can create a new Sync Source Picture Track right in the Media Synchronizer.

1. From the Sync Source popup menu, select ‘New Picture Track’.



00:00.00
00:07.216 - 00:08.048 Keanu, eat your heart out!
00:08.048 - 00:09.048 I'm too sexy for my body...

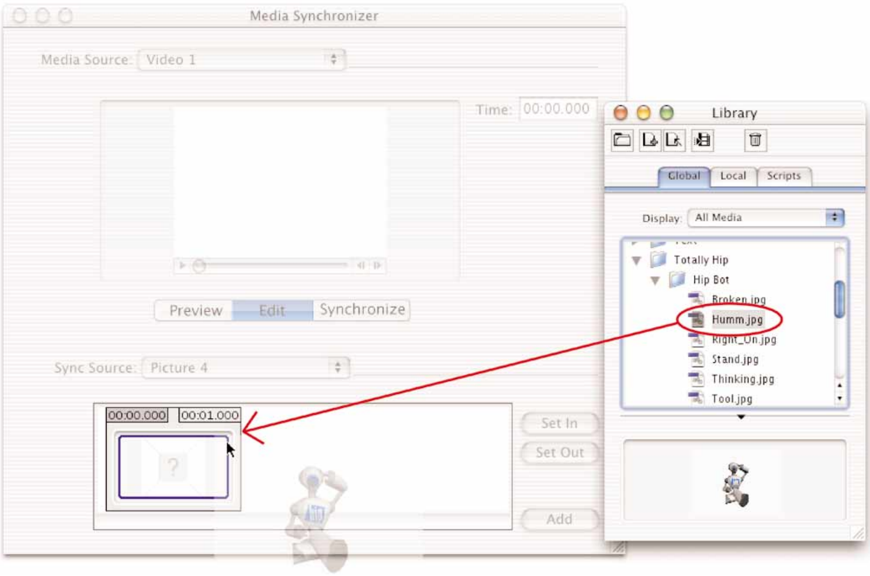
tip

You can use a number of keyboard shortcuts while working in the Media Synchronizer:

M	Create new Marker/Set In Point
N	Close new Marker/ Set Out Point
I	Set In Point
O	Set Out Point
J	Select next sample
K	Set In Point and select next sample
L	Set Out Point and select next sample

A new Picture Track appears in the Timeline, on the Stage, and in the Media Synchronizer Sync Source popup menu. The Sync Source Preview shows an empty sample.

2. Drag an image from the Library or your desktop to the empty sample in the Sync Source Preview. The image appears in the sample.



Creating a New Sync Source Text Track

1. From the Sync Source popup menu, select ‘New Text Track’.

   A new Text Track appears in the Timeline, on the Stage, and in the Media Synchronizer Sync Source popup menu. The Sync Source Preview shows an empty sample reading *no text*.

2. Double-click the new sample in the Sync Source Preview.

   The Text Sample Properties window opens.

3. Enter your text and close the Sample Properties window.

SET IN/SET OUT BUTTONS

These buttons are enabled when the Media Synchronizer is in edit and synchronize mode. These buttons are used to set the start and end times of the Picture or Text Track that is being synchronized to the Media Source track.

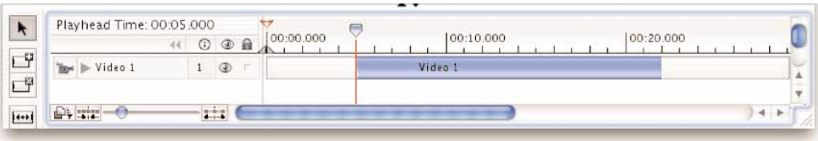
ADD BUTTON

You can add a new sample to the Sync Source directly from the Media Synchronizer by clicking on the Add button.

SYNC SOURCE PREVIEW

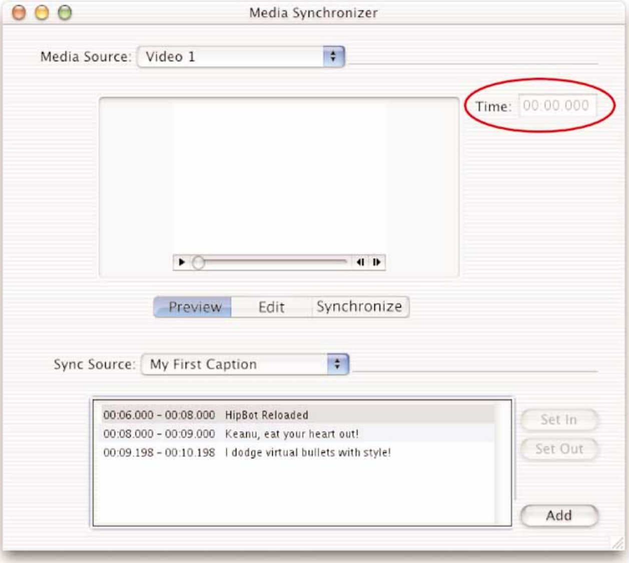
The Sync Source Preview displays the text or picture samples that can be synchronized to the Media Source. The time code range that appears with the sample is the actual Movie Time at which the sample will appear. If the Source track has a start time other than 00:00.000, the Time display to the right of the Preview Pane will display 00:00.000 indicating the time within this track. The Sync Source samples will take into consideration the offset and display their times relative to the start of the source media.

Example: `Video 1 is offset to 00:05.000`



Playhead Time: 00:05.000

Below the time display reads 00:00.000 (Source time) at start of Video 1 playback even though Video 1 track starts seconds into the movie.



Media Synchronizer

Media Source: Video 1

Time: 00:00.000

Sync Source: My First Caption

```
00:06.000 - 00:08.000   HipBot Reloaded
00:08.000 - 00:09.000   Keanu, eat your heart out!
00:09.198 - 00:10.198   I dodge virtual bullets with style!
```

In addition to displaying the Sync Source samples, you can also add and modify samples directly within this pane.

Adding and Modifying Picture Samples

There are a few different ways to add a new Picture sample to a Picture Track Sync Source. New samples come in with an automatic duration of one second that can be altered by adjusting the In and Out points in either synchronize or edit mode.

- Select the Sync Source Track and click the Add button. The Picture Sample Properties window opens. Drag a picture into the Sample Properties window and then close the window. The new picture sample appears at the end of the displayed samples. You may need to scroll to the end to view it. You can also keep this window open and use the New Sample button to add more samples.

- Drag one or more images from the Library or your desktop directly to the Sync Source Preview pane. The images appear in order at the end of the existing samples. You can also set the Preview's Playhead to a particular time and then drag an image in to have the sample begin at that point in the playback.

To change an image in a Sync Source sample:

- Drag an image from the Library or your desktop directly to the sample you want to replace. As you drag the image over the Sync Source Preview, the samples will outline in blue. When the sample you want replaced is outlined, drop the new image onto it. The new image will replace the previous one.

- Drag an image onto either the in or out time displayed just above an existing image. This will insert the image so that it starts or ends at the In or Out Point respectively.

- Double-click on the sample you want to change. The Sample Properties window opens. Drag a new image from your Library to the window, replacing the original image and close the window.

- Hold down the option key and drag one sample onto another to swap their images.

- You can also drag existing samples into new locations in the Sync Source. This will shift the samples to make room.

Example: Sample A has an In Point at 01:30.000 and an Out Point at 02:00.000.
Sample B has an In Point at 02:30.000 and an Out Point at 03:00.000.

Sample B is dragged before Sample A.

Sample B's new In Point is 01:30.000 and its new Out Point is 02:00.000.
Sample A's new In Point is 02:30.000 and its new Out Point is 03:00.000.

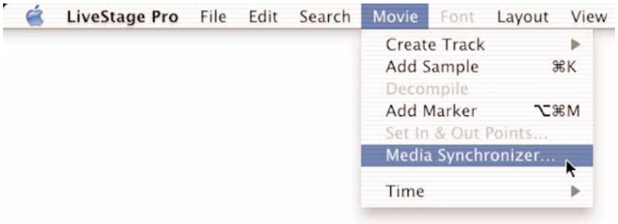
Adding and Modifying Text Samples

To add a new Text sample, select the Sync Source Text Track that you want the sample to be added to, and click the Add button. The Text Sample Properties window opens. Add the text you wish. Click on the New Sample Button to continue adding text samples, closing the window when you have finished. The new Text sample is added to the list in the Sync Source Preview. To modify a Text sample, double-click on it in the list in the Sync Source preview. Again, the Sample Properties window opens and you can modify the text.

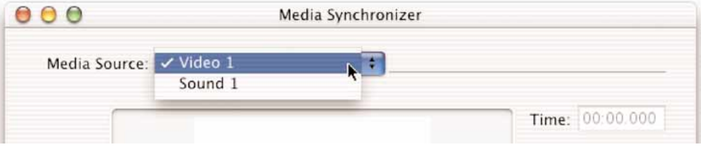
Synchronizing Media in Edit Mode

1. Select one of the tracks you want to synchronize.

2. From the Movie Menu, select Media Synchronizer.



3. From the Media Source popup menu, select the track with which you want to synchronize your Picture or Text Track.


Media Synchronizer

Media Source: ✓ Video 1

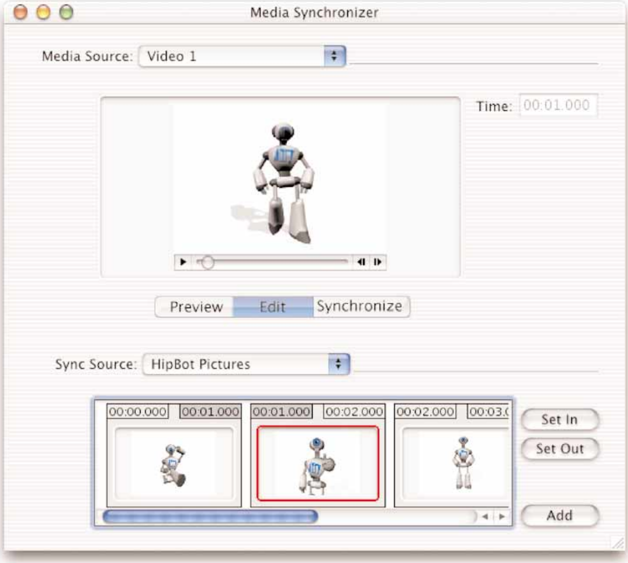
Time: 00:00.000

4. From the Sync Source popup menu, select the track you want to synchronize with your Media Source track, or create a new one.

5. Click the Edit button.

   The Set In and Set Out buttons become active.

6. Select a sample to synchronize by clicking on it within the Sync Source Preview pane.


Media Synchronizer

Media Source: Video 1

Time: 00:01.000

Sync Source: HipBot Pictures

7. Click the Media Source Preview's play button to view or listen to your source track.

8. Click the Set In button (or hit the 'I' key) at the point in the Source Media's playback where you would like your Sync Source sample to start.

9. Click the Set Out button (or hit the 'O' key) at the point in the Source Media's playback where you would like your Sync Source sample to end.

10. Repeat steps 6 – 9 for each sample within the Sync Source track that you would like to synchronize.

11. Close the Media Synchronizer and position your tracks on the Stage as you would like them to appear.

OPTIMIZING YOUR MOVIE

QuickTime renders media in real time. This is how QuickTime can maintain the integrity of all the media contained within it and at the same time keep file sizes small for projects such as slide shows. However, real-time rendering is a CPU-intensive process. For this reason, it is critical to construct your projects in a manner that minimizes real-time compositing wherever possible. This section will provide you with some fundamentals to optimize project construction.

Less is Better

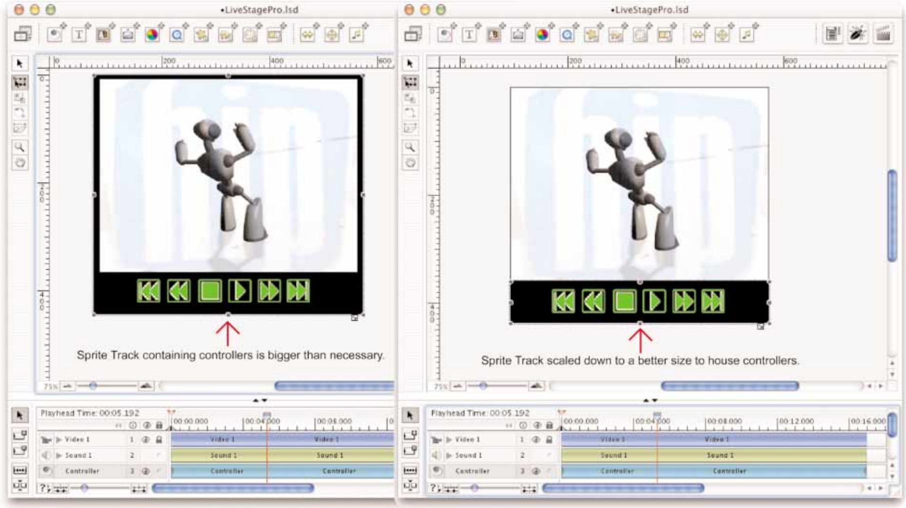
When constructing your LiveStage Professional projects, a good rule of thumb to follow is "Fewer Tracks, Smaller Dimensions". Fewer tracks as opposed to more; smaller tracks as opposed to larger. Economize wherever possible on the number of tracks and their dimensions, and try to limit the amount of overlap with other tracks, especially with video tracks.

FEWER TRACKS

Each track adds one or more memory buffers that QuickTime must create in RAM. Thus, the more tracks you have in a project, the harder QuickTime will have to work, and the more RAM it will use up. Wherever possible, you should attempt to constrain your media to as few tracks as is practical. If you can accomplish with one Sprite Track what you could accomplish with several, do it with one. Having multiple sprites in a single Sprite Track is preferable to having sprites spread out over multiple Sprite Tracks. Instead of having multiple Picture Tracks layered to create a background image, create the look you want in your graphics application and bring that image into LiveStage Professional 4.0 as a single Picture Track.

SMALLER DIMENSIONS

Keeping a track’s dimensions small is also important, especially when two or more tracks are being composited over one another. For example, say you are a building a full-screen movie trailer with a 640x480 Video Track. If you want to have a Flash or Sprite controller composited over the video using transparency or Alpha Channels, reduce the track size to fit the controller. As the controller will take up only a small fraction of the movie’s visual space, there is no need for its track to be 640x480. Making the controller track 640x480 would force QuickTime to engage in unnecessary compositing. Especially where Alpha Channels and transparency is concerned, it is preferable to have two or three smaller separate tracks, as opposed to a single large one.



Sprite Track containing controllers is bigger than necessary.

Sprite Track scaled down to a better size to house controllers.

WRONG

RIGHT

Alpha Channel vs. Masks

Alpha Channels are wonderful for achieving layered looks through multiple levels of transparency. However, they should be used intelligently as too many, or poor use can tax a CPU to the point where QuickTime starts dropping frames, or worse, the Alpha Channel itself. As with the above example, try to reduce the overall track size for tracks using Alpha channels, and optimize the number of tracks using Alpha Channels in a single project.

If you want to ‘frame’ your video, having it appear through a ‘hole’ in your movie, create a simple 1-bit black and white mask instead of using an Alpha Channel. Create the shape that you would like to frame the video. Areas in black will be rendered transparent so the video can show through, and areas in white will mask the video. Then drag this to the Matte well in the Video Track Properties window.

Loading

You can alter how QuickTime loads your tracks to help attain smooth overall playback.

For your movie to have the smoothest visual start, make sure critical tracks, such as those containing your interface, have the lowest Index Number and thus load first. Skin Tracks especially should always be at Index ‘1’. This way, the shape of the movie will already be established before the rest of the interface loads in.

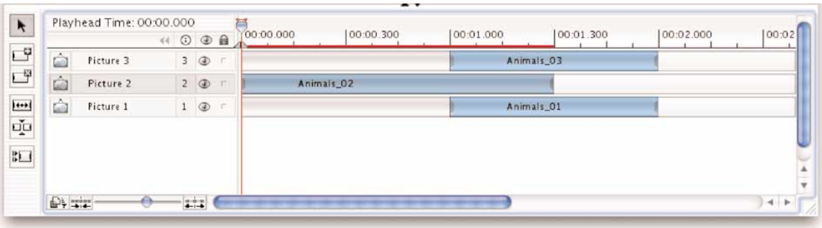
Loading tracks only when they are needed lessens the number of tracks being loaded simultaneously, reducing the number of memory buffers that QuickTime must create in RAM at any given time. Instead of just moving samples to the time when they appear in the Timeline, offset the entire track. You can offset tracks, and therefore change their load order, using the Track Manipulation tool.

1. In the Timeline, click the ⇥.

   You can now alter the duration and offset of your tracks.

2. Click and drag your tracks to the point along the timeline where they will first be needed.

3. Click the ↖ to return to the normal Timeline view.



The track of Index 2 will load before track of Index 1 as it comes before track of Index 1 in the Timeline. While both track of Index 1 and track of Index 3 are at the same time in the Timeline, track of Index 1 will load first because it has a lower Index number.

You can override all load considerations by preloading a track. This is done by selecting ‘Preload’ in the Advanced tab of the Track Inspector or Track Properties window. This will direct QuickTime to load the track prior to the movie starting. As a rule of thumb, preload Flash Tracks to ensure they have fully loaded before your movie starts.

Chapter 15

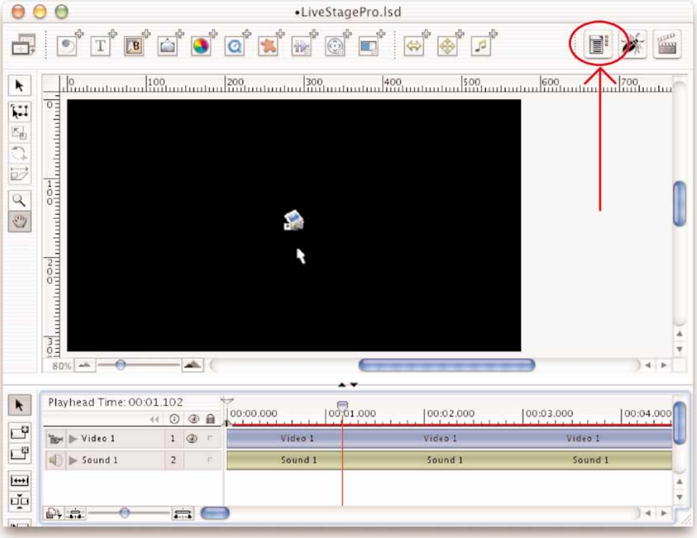
Exporting Your Movie

INTRODUCTION

The final step in the production of your project is exporting your finished QuickTime movie. Throughout the creation of your project, you have set various parameters for individual elements within your movie. Before you export it, you will want to also set parameters for the completed QuickTime movie to ensure it plays as you intend, and contains any relevant information you require.

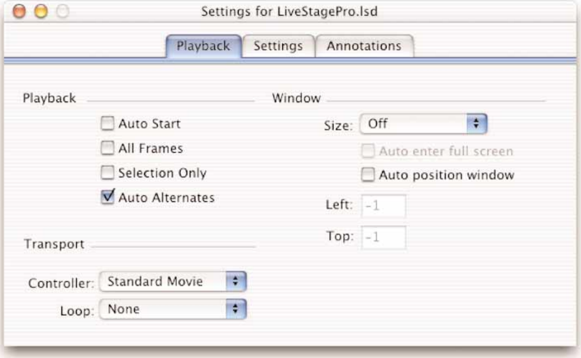
DOCUMENT SETTINGS WINDOW

The Document Settings Window provides a number of options that specify the conditions of playback, general movie settings, and any annotated information you want included with your finished QuickTime movie. To access the Document Settings Window, select “Document Settings” from the Edit menu (Option-Command-I), or click on the Document Settings button located at the top of the Document Window.



Playback Tab

In the Playback tab, you set parameters that determine how you would like your final movie to play on the end-user’s computer.



Settings for LiveStagePro.lsd

Playback | Settings | Annotations

Playback

□ Auto Start
□ All Frames
□ Selection Only
☑ Auto Alternates

Transport

Controller: Standard Movie

Loop: None

Window

Size: Off

□ Auto enter full screen
□ Auto position window

Left: -1

Top: -1

AUTO START

When you select this option, your QuickTime movie will begin to play as soon as it is opened. If you are presenting your movie online, Auto Start will allow the movie to begin to play while QuickTime continues to download the file. However, if your movie's data rate is faster than the user's connection speed, the movie will stop and the user will need to restart the playback.

ALL FRAMES

This option tells QuickTime to play every frame of the movie. This may sound like a good idea, but QuickTime's foremost objective is to synchronize any sound track with your visual media. To accomplish this task, QuickTime will drop frames as necessary to maintain this synch.

Having Play All Frames selected keeps QuickTime from maintaining synch, so it will drop any sound track present from the playback of the movie. It can also cause other undesirable side effects like a slow and inconsistent frame rate. As LiveStage Professional is designed to access all of QuickTime's features, this alternative is provided, but it is strongly recommended that you avoid using it.

SELECTION ONLY

If you have created a preset selection in your movie through QScript, check this box to play only that portion of your movie.

AUTO ALTERNATES

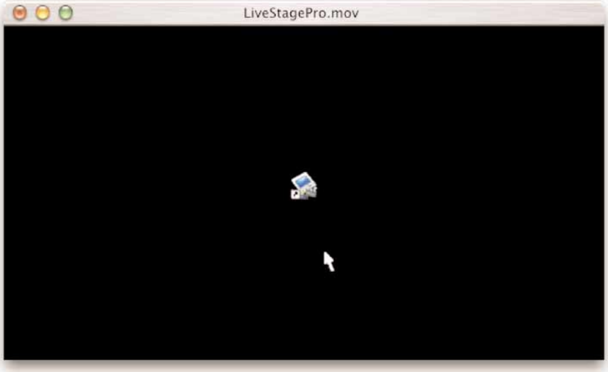
You can author alternate tracks within your project for users with different language and bandwidth requirements (see Chapter 14—Working with Multiple Tracks, page 285). Selecting this feature enables QuickTime to automatically select the appropriate Alternate track for playback on the end-user's computer.

CONTROLLER

Through this pop-up menu, you can elect to provide your user with QuickTime's standard or QTVR-specific controls, or choose to present your movie without a QuickTime controller.

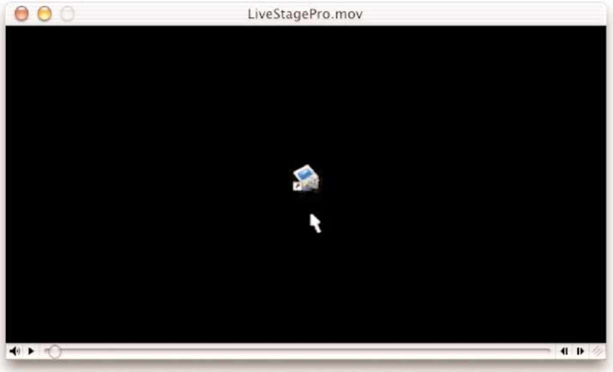
None

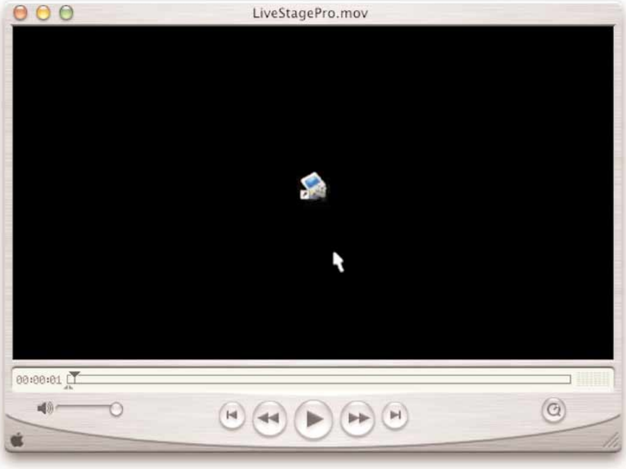
If you have created a controller of your own and/or would prefer not to have the QuickTime controller present, select None.



Standard Movie

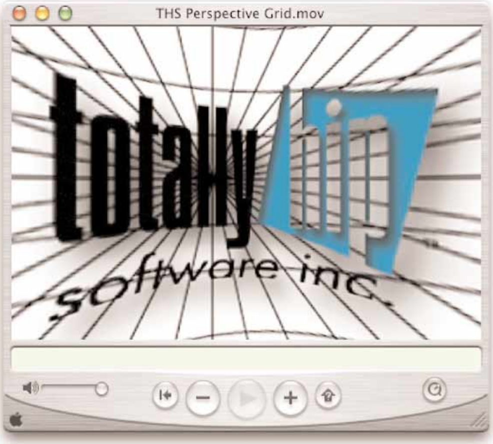
The Standard Movie controller provides the viewer with QuickTime's playback and volume controls.





QuickTime VR

If you are exporting a movie with QuickTime VR media, and would like the standard QuickTime VR controller, select this option.



LOOP

From this list, you can opt to repeatedly loop your movie until the movie is stopped.

From Beginning

This option plays the movie from start to finish, then returns the Playhead to the beginning to start again.

From End (Palindrome)

From End plays the movie from start to finish, then plays in reverse back to the beginning where the process starts again. Streaming movies cannot Palindrome.

WINDOW SIZE

Full Screen mode is a presentational means of displaying your movie within the QuickTime Player that is often used in online movie trailers. Depending on the setting you select (and the resolution of the user's monitor), the movie will either scale to fill the screen or will be centered on a black background. In some cases, QuickTime alters the monitor resolution during the playback of a full-screen movie, and resets the resolution when the movie is quit or taken out of full screen mode.

Off

This plays the movie in a window instead of in Full Screen mode.

Normal

The movie is played at its originally authored size.

Double

The movie plays full screen at double its normal size.

Half

The movie plays full screen on a black background at half its normal dimensions.

Full Screen

This setting scales the movie up or down to fill the user's screen as much as possible while preserving the movie's aspect ratio.

Current

This plays the movie in full screen at the size it was when last saved.

AUTO ENTER FULL SCREEN

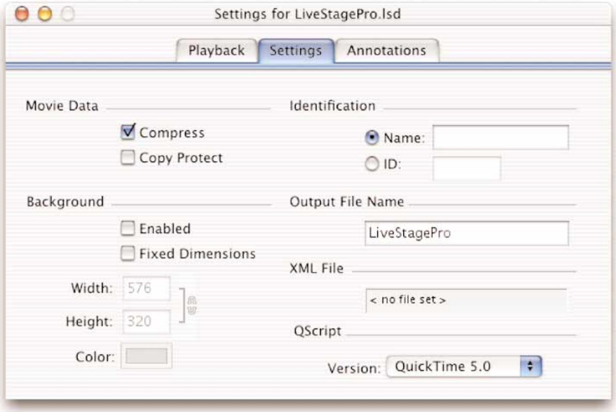
When a window size is selected, you can opt to have the movie automatically enter full screen mode when it is launched in the QuickTime Player. If you do not select this option, the movie will not play full screen until it is triggered through an action executed in a QScript.

AUTO POSITION WINDOW

By clicking this option and entering Left and Top coordinates, you can determine where on the user's screen your movie will be positioned when it is launched through the QuickTime Player. This is a new feature introduced in QuickTime 6.

The Settings Tab

The Settings tab is concerned with the identification and production parameters involved in the actual creation of your movie.



Movie Data

☑ Compress
□ Copy Protect

Background

□ Enabled
□ Fixed Dimensions

Width: 576
Height: 320
Color:

Identification

● Name:
○ ID:

Output File Name

LiveStagePro

XML File

< no file set >

QScript

Version: QuickTime 5.0

MOVIE DATA

This section is where you set up the properties involved in the actual creation of the movie.

Compress

This selection compresses the movie resource, making a movie with a smaller overall file size. This is particularly valuable for web deployment and is selected by default.

Copy Protect

This option adds a certain amount of copy protection to your movie by preventing users from saving it to their computers or modifying it. This is not complete copyright protection however, as some browsers can still save the movie in their caches. In addition, while this protects the movie from modification it does not prevent the movie file from being copied from one machine to another.

BACKGROUND

You can elect to have a colored background in your movie, and decide whether your movie should be constrained to precise dimensions.

Enabled

Checking this adds a background to your movie in the color that is specified in the Color field at the bottom.

Fixed Dimensions

To restrict your movie to set dimensions during construction, select this option and enter the desired Width and Height. If you wish to constrain the aspect ratio of your movie, click the Link icon to the right of the dimension fields.

Color

If you have enabled a background, click this to select its color.

IDENTIFICATION

Identification is required only for inter-movie communication. When one QuickTime movie communicates with another, it calls to that movie's Name or ID tag. This tag is completely separate from the movie file's name.

OUTPUT FILE NAME

This will be the name of your resulting QuickTime movie file. Entering your file name here will save you from having to continually input the movie's file name every time you export your project.

XML FILE

Drag an XML file to this field to include it in your QuickTime movie. This is used in conjunction with QTLists (see Chapter 18—XML and QTLists, page 356). To remove an attached file, select this field and hit the Delete key.

QSCRIPT

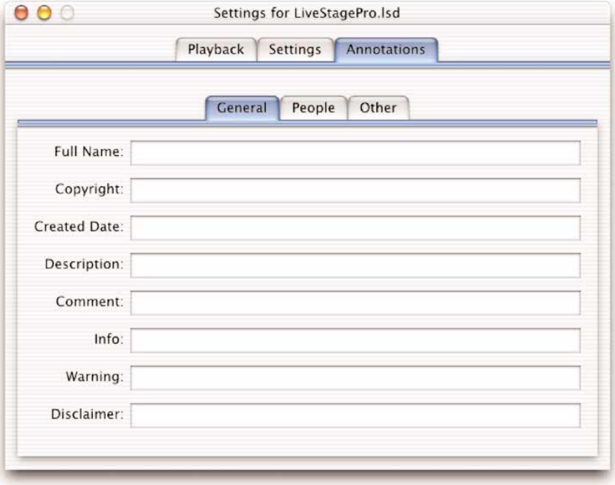
This sets the QuickTime version of the QScripts that will be available to your project. If you select QuickTime 4.0, and your project contains QScripts that were introduced in QuickTime 5.0, error messages will display and your movie will not compile when you attempt to preview or export it. Moreover, QScript commands that apply to higher versions will not be recognized in the Script Editor.

THE ANNOTATIONS TAB

This tab contains fields where you can specify credits, content, and copyright information to be embedded into your exported movie. The Movie Info window of the QuickTime Player (Command-I) will display the contents of the following fields:

- Full Name
- Copyright Info
- Info
- Artist
- Performer
- Album

The remaining information can be viewed in the QuickTime Player in the Movie Properties window (Command-J) under Annotations.



Settings for LiveStagePro.lsd

Playback | Settings | Annotations

General | People | Other

Full Name:

Copyright:

Created Date:

Description:

Comment:

Info:

Warning:

Disclaimer:

GENERAL

The General tab contains information related to the movie including its full name, copyright, creation date, and general information. The title you enter into the Full Name field will appear as the movie's heading in the QuickTime Player. If this is blank, the Output File Name you entered in the Settings tab ("myMovie.mov") will appear instead.

PEOPLE

The People tab contains information on the producers and creators of the movie's content.

OTHER

The Other tab contains system requirements or recommendations to help ensure proper playback, as well as other miscellaneous information related to the movie.

EXPORT OPTIONS

LiveStage Professional provides several options for exporting your completed movie to suit a range of deployment and project management needs. Select the export option you desire through the File>Export menu.

QUICKTIME MOVIE

This option compiles your project and exports it to your computer, provided there are no errors in your project's QScripts. The Output File Name you specified in the Settings tab of the Document Settings window will appear in the Save dialog and the .mov extension will be appended to the file name.

Use this option when you require only the QuickTime movie alone.

WEB

Export to Web provides you with both the QuickTime movie and a simple HTML document containing the code that embeds it. If you are deploying on the Internet, upload both files to a server to produce a simple web page containing your movie. Alternately, you can also copy and paste the OBJECT and EMBED tags from the provided code into your own HTML document or WYSIWYG editor.

CD

The Export to CD option meets the challenge of producing CDs for viewers who may not have QuickTime, or the correct version of QuickTime installed. Export to CD creates a "QuickStart" application for both platforms, an autostart file for Windows and a data file, in addition to the QuickTime movie. This function also creates a QuickTime file into which you place your QuickTime Installer. When included on the CD-ROM, these files ensure that the viewer has the most seamless launch possible, whether or not QuickTime is installed on their computer.

WHAT QUICKSTART DOES

QuickStart looks for the version of QuickTime you have specified on the user's computer. If it is installed, your movie is automatically launched. If QuickTime, or if the required version of QuickTime is not installed on the user's computer, the QuickTime installer is automatically launched. The user experience differs slightly from Mac and Windows:

WINDOWS

When the user inserts the CD-ROM into the CD drive:

- A file runs automatically, detecting whether or not the user has the specified version of QuickTime installed.

- If the user has the required version, your QuickTime movie is automatically launched.

- If the user does not have the required version, a dialog appears notifying the user that the particular version of QuickTime needs to be installed. When the user clicks the displayed "Install" button, the QuickTime installer is run.

- Once the installation is complete, the user closes out of the installation and your movie is lauched.

MAC

When the user inserts the CD-ROM into the CD drive and double-clicks on the QuickStart application on the CD-Rom:

- The QuickStart application will detect whether or not the user has the correct version of QuickTime installed.

- If the user has the required version, your QuickTime movie is automatically launched.

- If the user does not have the required version, a dialog appears, notifying the user they need to install the appropriate version of QuickTime. The user clicks on the “Install” button, and the QuickTime installer is run.

- Once the installation is complete, the user’s computer reboots.

- Once the computer is rebooted, the user double-clicks the QuickStart application again and your movie is launched.

EXPORTING TO CD

1. From the File Menu, select Export>CD.

2. A Save dialog window opens.

   Save your movie to the directory of your choice.

3. Select which version of QuickTime you want QuickStart to detect.

4. Click OK.

   The following files are created:

   QuickStart: The application that detects and launches the installer and/or your movie on Macintosh computers.

   QuickStart.exe: The application that detects and launches the installer and/or your movie on Windows computers.

   Autorun.inf: This automatically launches the QuickStart application on Windows machines when the user inserts the CD into the CD-ROM drive.

   QTConfig.ini: This is a data file that is used by QuickStart on both platforms.

   “MyMovie.mov”: This is your exported QuickTime movie.

   In addition to these files, a “QuickTime” folder will be created. This is where you will need to place the QuickTime standalone installer for the version of QuickTime you want your users to have installed. To obtain a copy of the QuickTime 6 standalone installer, visit: http://www.apple.com/quicktime/download/ standalone/.

5. Drag the QuickTime standalone installer into the QuickTime folder.

6. Place the above files and the QuickTime folder in the root directory of the CD you are burning.

XML

This option does not export a QuickTime movie, but instead exports the LiveStage Professional project to your computer as an XML project file. For more information, see Chapter 18—XML and QTLists, page 352.

Gathered Project

The Export Gathered Project option is very useful for archiving or sharing a LiveStage Professional project. This option does not export a QuickTime movie, but instead gathers all the media files associated with your project from your computer and copies them, along with your project file, into a new folder bearing the project file's name.

Chapter 16

Introduction to QScript

OVERVIEW

LiveStage Professional's QScript is a simple object-oriented scripting language created by Totally Hip Software. QScript puts the interactivity into LiveStage Professional movies. With QScript, elements of your movie can respond to user interaction such as a mouse click. Through QScript, movie elements and even individual movies themselves can act on and interact with each other.

QScript, unique to LiveStage Professional, is similar to other multimedia scripting languages like Lingo and JavaScript. Developers who are familiar with such programming languages will likely find QScript easy to learn. Even those with little to no scripting experience will find the basic functions in QScript straightforward and easy to grasp, though projects that are more intricate will pose greater challenges. LiveStage Professional provides assistance to seasoned programmers and newcomers alike through the QScript Reference Window. However, this reference is not a substitute for an understanding in scripting. While simple QScript can produce impressive effects, it is strongly recommended that you have a solid comprehension of scripting principles as you progress to projects that are more complex.

This chapter will introduce you to the tools in the LiveStage Professional interface that you will use while scripting, and then move on to cover some of the basic elements that make up QScript. For detailed information on individual functions, expressions, targets, actions, and properties, see the QScript Reference Appendix .

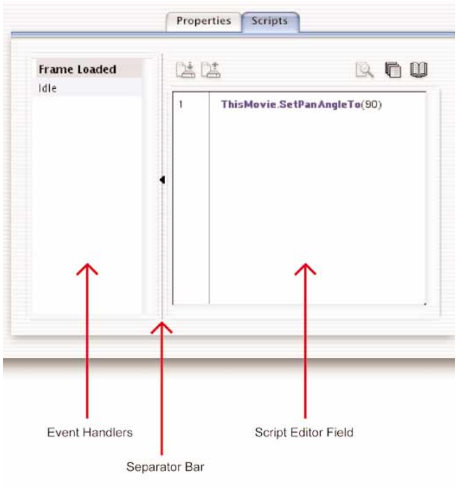
SCRIPT EDIT WINDOW

QScripts are added to scriptable tracks through their individual Script Edit Windows. There are presently four scriptable tracks in LiveStage Professional:

- Sprite Track
- Text Track
- Flash Track
- VR Track

The Script Edit Window is located in the Sample Properties window. While there are some variations between the Script Edit Window interfaces of each particular track type (see the individual Track chapters for more information), some features are common to all.

The Script Edit Window is divided into two sections. To the left, a list of the 'Event Handlers' for the sprite, HotSpot (VR and Text Tracks), or sample is displayed. Scripts attached to an Event Handler run when the event is executed (i.e., a mouse click). The right side contains a Script Editor field. This field is where you can view the scripts attached to the selected Event Handler as well as add new scripting. Event Handlers in boldface have scripts attached to them.



```
ThisMovie.SetPanAngleTo(90)
```

To add a script to an Event Handler, select the Event Handler in the list and then click in the Script Editor field to enter your script. To make more room for your scripts, you can hide the Event Handler list by clicking on the arrow on the resize bar that separates the two panes. Click it again to restore the Event Handler list. You can also enlarge the entire sample window.

Event Handlers

Event Handlers are specific events that execute QScripts within scriptable tracks. There are two types of events, Object Events, and Track Events. Object Events are events directed at a specific object within a scriptable track, such as a sprite, Text HotSpot, Flash Button, or QTVR HotSpot. These events are often generated by mouse actions. For instance, if you have placed a script into the Mouse Click Event Handler of a particular sprite, when the user clicks on that sprite your script will execute. Track Events trigger scripts according to the passage of time as well as some particular actions that impact on the track.

FRAME LOADED

The Frame Loaded event is triggered when the sample is loaded into the movie. Each sample has only one Frame Loaded Event. If you want to place Frame Loaded events on one or more sprites within a sample you must make sure to ‘target’ the sprite(s) in your script.

LIST RECEIVED

The List Received Event is a Track Event triggered when a QTList is received in response to an ExchangeList or QueryListServer QScript Action (see Chapter 18—XML and QTLists on page 358). This event is only available for Flash, Sprite, and Text tracks.

IDLE

The Idle Event is triggered whenever the movie is not occupied with another task. Setting a value in the Idle Delay field of the track’s Inspector or Properties window determines how often you would like the Idle Event to be executed. A rate of 0 executes the Idle as quickly as possible and a rate of -1 turns the Idle Event off. The default rate is 1, meaning one Idle

Event per 1/60th of a second - or one 'tick'. However, the actual rate can vary according to the speed of the user's computer, the complexity of the project, and the type of media, so an Idle Event should not be used to trigger an event that requires precise timing. Idle time processing can slow a movie so it is best to keep the Idle Delay to as large a value as possible. This event is only available for Sprite, Text, and VR Tracks.

KEY PRESSED

The Key Pressed Event is a Track Event that is triggered when the user presses a key on the keyboard. This event is only available for Flash, Sprite, and Text tracks.

MOUSE MOVED

The Mouse Moved Event is a Track Event triggered when the user moves the mouse within a Sprite or Text Track.

MOUSE CLICK

The Mouse Click Event is an Object Event that executes its script when the user presses and then releases the mouse button while the cursor is over the sprite or HotSpot.

MOUSE ENTER

The Mouse Enter Event is an Object Event and is triggered when the user moves the mouse over a sprite or HotSpot in a Scriptable Track.

MOUSE EXIT

The Mouse Exit Event is an Object Event that triggers when the user moves the mouse out of a sprite or HotSpot in a Scriptable Track.

MOUSE DOWN

The Mouse Down Event is an Object Event that is executed when the user presses the mouse button down while over the sprite or Hotspot.

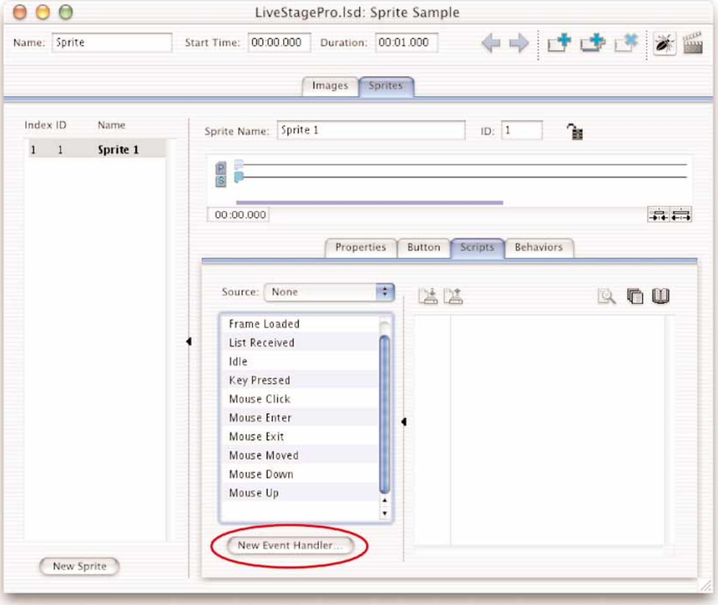
MOUSE UP

The Mouse Up Event is an Object Event triggered when the user releases the mouse button after pressing it over a sprite or Hotspot, regardless of where the cursor is when the mouse button is released.

CUSTOM EVENTS

In addition to the standard Event Handlers, you can create a Custom Event Handler and decide precisely when you want it to be triggered. Custom Event Handlers are available only in the Sprite Track and are created by clicking on the New Event Handler button found in the Sprite Sample Properties Scripts tab.

1. In the Scripts tab of the Sprite's Sample Properties window, click on the New Event Handler Button.



LiveStagePro.lsd: Sprite Sample

Name: Sprite
Start Time: 00:00.000
Duration: 00:01.000

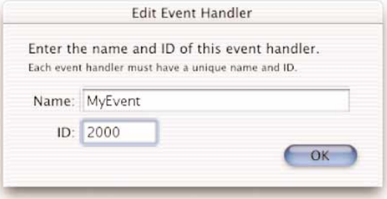
Index ID	Name
1 1	Sprite 1

Sprite Name: Sprite 1
ID: 1

00:00.000

Source: None

2. Enter a Name and/or ID Number for your Event.



Edit Event Handler

Enter the name and ID of this event handler.
Each event handler must have a unique name and ID.

Name: MyEvent
ID: 2000

Click OK.

3. Enter the script you would like to execute when this event is triggered.

Currently there is a bug in QuickTime 4.1 with Windows that does not convert Custom Event Handler ID numbers to a format that Windows understands. Use any of these event numbers if you run into this bug: 65792, 13154, 197376, 263168, 328960, 394752, 460544, 526336, 592128, 657920, 723712, 789504, 855296, 921088, 986880.

Triggering Custom Events

Custom Events are triggered through the ExecuteEvent QScript command or a Modifier track. You can trigger the Custom Event either by ID or by Name:

```
ExecuteEvent(ID) or ExecuteEvent($"NAME")
```

Example: `ExecuteEvent($"MyEvent")`

Different sprites can have Custom Event Handlers with the same Name and ID number, but with entirely different scripts, allowing them to respond differently to the same ExecuteEvent command. Simply target the sprite you would like to affect (see Targeting on page 320 of this chapter) in your QScript:

Example: `SpriteNamed("MySprite").ExecuteEvent($"MyEvent")`

Script Management

The Script Edit Window provides several time saving tools to manage and incorporate scripts.

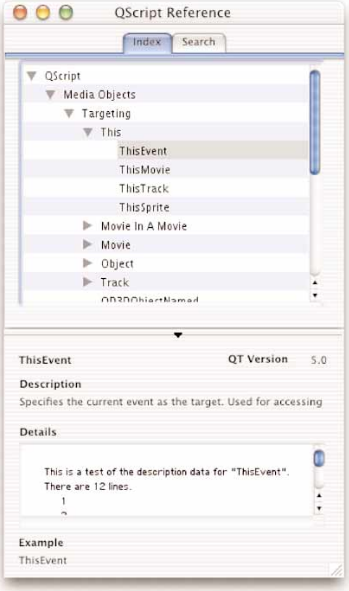
TOOL	NAME	FUNCTION
📥	Save QScript to File	Click to save the script currently in the Script Editor field for future use. This will save the script as a .qscript file to the directory of your choice.
📤	Load QScript from File	Click to incorporate a script saved in an external text or .qscript file. Select the file from the dialog that appears. This replaces any scripts currently in the Script Editor field with the contents of the selected file.
🔍	Check Syntax	Click to verify that the syntax of the QScript contained in the Script Edit field is valid. If the syntax is not valid, an error message will appear with an explanation of the error(s).
📖	Open Defines Window	Click to open the Defines Window (see the Defines Section, page 321).
🗐	Open QScript Reference Window	Click to access the QScript Reference window.

QSCRIPT REFERENCE WINDOW

The QScript Reference window is a powerful reference for all the QScript elements available to you in LiveStage Professional. It is easily accessed through the Window Menu by selecting “Show QScript Reference” (Command-T), or by clicking the QScript Reference window button in the Script Edit window. You can search for and drag the QScript elements you find within the QScript Reference window directly to your scripts, greatly reducing the risk of syntax and spelling errors. However, proper syntax does not guarantee that your QScript will execute as you expect.

The QScript Reference window is separated into two tabs. The Index tab contains the QScript keywords and the Search tab allows you to search for a particular QScript element.

The QScript Reference window can also help you to better understand individual script elements. Information regarding each element appears when you select an item in the window. You can view the item's detailed description, the minimum QuickTime version required to run it, an example of its syntax, and when available, links to further information online.



ThisEvent QT Version 5.0

Description

Specifies the current event as the target. Used for accessing

Details

This is a test of the description data for "ThisEvent".
There are 12 lines.
1

Example

ThisEvent

QT VERSION

This refers to the version of QuickTime that supports this scripting element. If you use a script in your project that is supported in QuickTime 5.0, but export your movie to comply only with QuickTime 4.0 your movie will not compile and will produce an error message.

DESCRIPTION

The Description is a brief summary of what the QScript element is or what it does.

DETAILS

This area offers you a more detailed explanation of the term or element you select. These explanations often provide information as to the parameters that you might need to include in the script element, and in what circumstances you might use it.

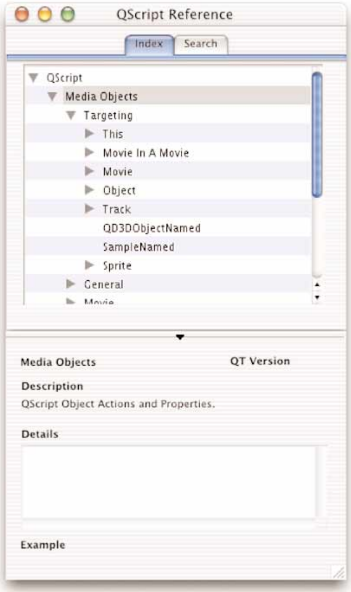
EXAMPLE

This displays an example of the valid syntax for the selected element.

MORE BUTTON

The More Button appears when additional online information is available for the item you have selected. Clicking this button when connected to the Internet will open a browser window containing the most recent information available on the selection along with any additional links that offer deeper explanations or examples.

Index Tab



Media Objects QT Version

Description

QScript Object Actions and Properties.

Details

Example

The Index tab is comprised of all available QScript elements categorized into various folders. Clicking on a triangle to the left of a list item expands its folder to reveal subfolders and individual QScript elements. Once an item is selected, you can also navigate the QScript Reference window using the arrow keys.

MEDIA OBJECTS

The Media Objects folder comprises all the targets that can be specified in QScript, as well as all the actions and properties that can be used on individual objects (see the Actions, Properties and Targeting sections in this chapter for more information).

CONSTANTS

This is where you can select from the pre-defined constants available in LiveStage Professional, including Boolean, Mathematical, Graphics Modes, and Movie Load Status, among others (see the Constants section in this chapter for more information).

FUNCTIONS

Mathematical and String functions for use within QScript expressions are housed in the Functions folder.

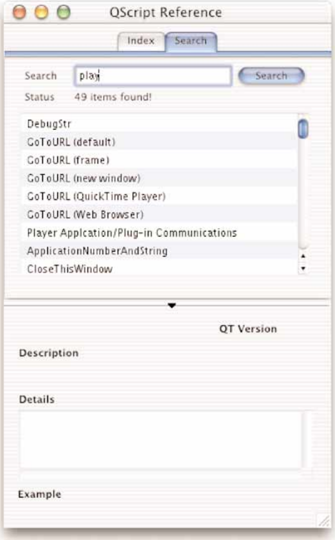
QSCRIPT COMPILER ERROR MESSAGES

When you attempt to preview or export your movie and the QScript in your project contains one or more errors, an error message will appear. This folder contains explanations of the different error messages that you might receive, along with suggestions as to how to correct the problem (see the Errors Window section in this chapter for more information).

QSCRIPT SYNTAX

In this folder, you will find various QScript syntax elements such as Control Statements, Variable Declarations, and a few helpful hints on QScript syntax itself.

Search Tab



Search play

Status 49 items found!

DebugStr
GoToURL (default)
GoToURL (frame)
GoToURL (new window)
GoToURL (QuickTime Player)
GoToURL (Web Browser)
Player Applcation/Plug-in Communications
ApplicationNumberAndString
CloseThisWindow

QT Version

Description

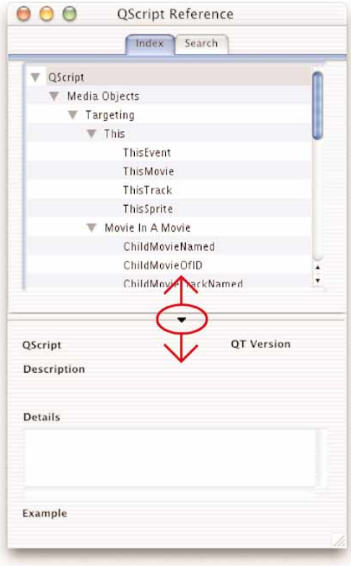
Details

Example

The QScript Reference window has a powerful search function that helps you find the element you need. Enter a keyword in the Search field and any QScript items that contain that word, whether in the QScript statement or in the description, will be displayed. You can select the displayed items and view their specific details as with the Index Tab.

Working in the QScript Reference Window

You can resize the QScript Reference window as needed to accommodate long lists or explanations. Clicking and dragging on the resize bar adjusts the proportions of the top and bottom sections. To maximize the upper section, click on the down triangle in the middle of the resize bar. Click again to return it to the previous position.



Incorporating QScripts into your Project

To incorporate QScript found in the QScript Reference window, you can either double-click on the element or drag it to the Script Edit Window. Double clicking will place the element at the cursor's location in the active Script Edit Window. Dragging the element will place it at the point where you release the mouse.

ERRORS WINDOW

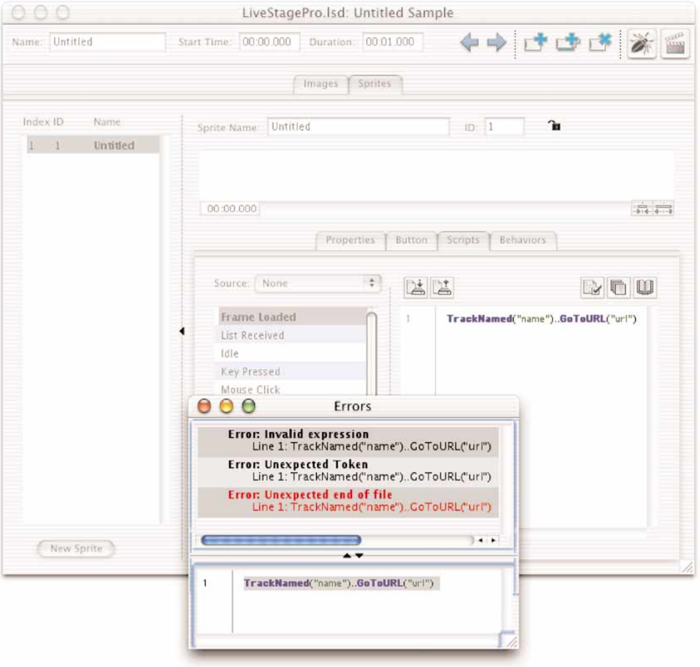
The Errors Window displays any errors in your project's QScript when you:

- check the syntax in your Script Edit Window.
- preview your movie.
- export your movie.

All QScript errors must be corrected in order for your movie to compile. The Errors Window assists in this process by listing the individual errors the compiler has encountered. To correct the QScript error, you can either:

- double-click on the error message. This will open the particular Script Edit Window in your project that contains the error so that you can correct the script.
- use the field below the Error Message list to correct the script in place.

Definitions of the Error Messages you might encounter can be found in the QScript Reference window, along with suggestions as to how to correct the problem.



LiveStagePro.lsd: Untitled Sample

Name: Untitled Start Time: 00:00.000 Duration: 00:01.000

Sprite Name: Untitled ID: 1

Index ID		Name
1	1	Untitled

Source: None

```
TrackNamed("name")..GoToURL("url")
```

Errors

```
Error: Invalid expression
    Line 1: TrackNamed("name")..GoToURL("url")
Error: Unexpected Token
    Line 1: TrackNamed("name")..GoToURL("url")
Error: Unexpected end of file
    Line 1: TrackNamed("name")..GoToURL("url")
```

```
TrackNamed("name")..GoToURL("url")
```

QSCRIPT STRUCTURE

QScript is a simple scripting language that uses a dot syntax structure. At its most basic, QScript takes the form

```
<Target>.<Action_or_Property>
```

The target is separated from the action or property by a "." , hence, "dot syntax".

A line of QScript can be composed of data and statements that comprise several basic elements.

Targets

A Target specifies the movie element you want the script to refer to, be it a track, sample, sprite, or even a movie. While it is not always mandatory to provide a target when working with actions and properties, it is a good habit to have some scripts will fail without one.

Example: `TrackNamed(“Sound 1”).SetVolumeTo(125)`

This targets the specific track by the name of "Sound 1" and directs it to adjust its volume to 125.

The target objects that can be specified in LiveStage Professional are listed in Appendix 1—QScript Reference.

Actions

Actions are statements that direct objects in the movie to perform tasks.

Example: `ThisSprite.Rotate(90)`

This action rotates the sprite containing this script 90 degrees clockwise.

A complete list of Actions available in LiveStage Professional can be found in Appendix 1—QScript Reference.

Properties

Individual movie objects have properties made up of data values. When used in QScript, a property returns the data value of a particular aspect of the targeted object.

Example: `ThisTrack.GetDuration`

This script returns the duration of the track that contains the script.

Constants

Constants, also called “Literals”, are elements of data that do not change when the script is executed. Constants are used in QScript statements and can be made up of numerical values, strings, or TRUE and FALSE values.

You can create your own constants in the Defines Window to easily insert or edit a defined value that is used in various places within your project’s QScript. When you compile your movie, LiveStage will go through all the QScripts in your projects and, wherever you have used your defined constant, LiveStage will replace the name with the value you assigned to it. Whenever possible, it is a good idea to use a defined constant name instead of an actual number. It can make your code easier to understand and allows you to quickly change its value without having to search throughout all your scripts for where you used that value.

DEFINES WINDOW

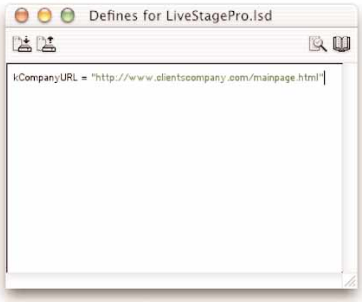
The Defines Window is where you enter constants or any other text that you would like to define for use in any scripts you write throughout your project. You can access the Defines Window either through the Edit>Document Defines on the Menu Bar, or through the 📖 in the Script Edit Windows.

In the Defines Window, enter a name and then assign this name a value that you will want to refer to in your QScripts. The format for creating constants is:

```
kConstantName = Value
```

Pre-defined constants in LiveStage Professional are prefixed with a ‘k’ and many programmers follow suit when defining their own constants. While it is not mandatory to add this prefix, it makes it easier to visually distinguish constants from other data in your QScript.

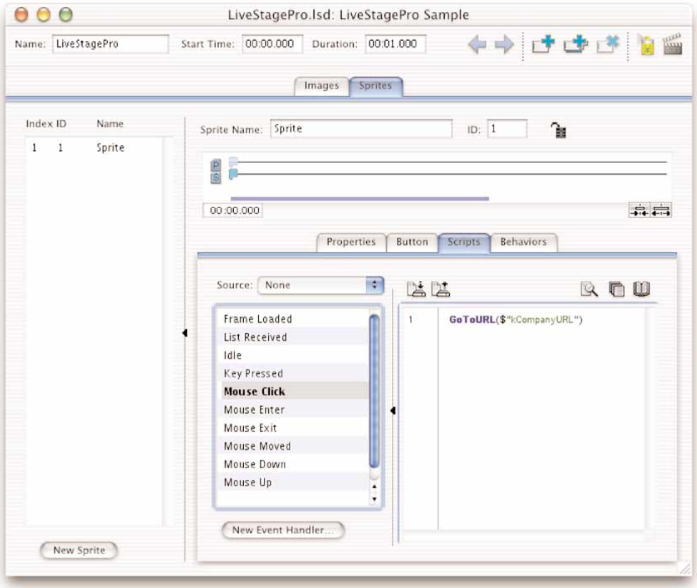
Example: `kCompanyURL = “http://www.clientscompany.com/mainpage.html”`



```
kCompanyURL = "http://www.clientscompany.com/mainpage.html"
```

Refer to the constant in your script by placing a $ before the “constant name”.

Example: `GoToURL($“kCompanyURL”)`



Name: LiveStagePro Start Time: 00:00.000 Duration: 00:01.000

1 1 Sprite

Sprite Name: Sprite ID: 1

00:00.000

Source: None

```
GoToURL($"kCompanyURL")
```

If the URL changes, you can edit it within the Defines Window. This will automatically update all the occurrences of kCompanyURL in your project’s script when you compile it.

You can also use spaces in your Define name, provided you enclose the words in quotes such as: “my number” = 5.

IMAGE DEFINES

LiveStage Professional automatically creates Defines for all the images used in your Sprite Track. LiveStage Professional always encloses these Defines in quotes. If you have an image named “Logo”, instead of referring to the Image Index in your QScript, you can instead refer to the name, as you would with other constants, like this: $“Logo”.

tip

You can also define a short portion of scripts in the Defines Window: TheTarget = TrackNamed(“the track”).SpriteNamed (“the sprite”). You can then simply enter “TheTarget” in your scripts instead of rewriting the entire line of code.

Variables

Unlike constants, variables are objects that contain values that can change. Variables store information while the script is running. This information can then be changed during the execution of statements. Variables can contain either numbers or strings that QuickTime will convert to the appropriate type depending on the parameter called for. In other words, if a variable contains a numerical value of 25 and the parameter called for when the variable is passed is a string, QuickTime will convert the numerical value of 25 to the string "25". This also works in reverse. If your variable is a string "25", any mathematical functions that operate on the string will read the string as a numeric value and will thus produce a numeric result.

VARIABLE DECLARATION

Variables must be introduced by name or 'declared' before you use them in a script. Declaring the variable tells LiveStage Professional what the variable's name is. It also defines what aspects of the project the variable will apply to, in other words, its 'scope'.

There are four types of scope available to QScript variables: Movie, Global, Sprite, and Local. Global, Sprite, and Local Variables are for use exclusively in sprites. Movie Variables can be used in sprites and must be used in Flash buttons, as well as in Text and VR HotSpots.

Movie

Movie Variables, declared as MovieVars, can be accessed from any script in the movie, as well as from other movies through Inter Movie Communication).

Example: `MovieVars TimesButtonClicked`

You can also declare a Movie Variable using an ID that can be used for inter-movie communication. To declare a Movie variable with an ID, add the ID to the end of the declaraton as follows:

MovieVars TimesButtonClicked : 3

This sets an ID of 3 to the TimesButtonClicked variable. You can use any number from 1-10,000 for the ID.

Global

Global Variables, declared as GlobalVars, are accessible from any script in the Sprite Track where the particular Global Variable is declared.

Example: `GlobalVars DistanceDragged`

Sprite

Sprite Variables, declared as SpriteVars, are available only to scripts contained in the same sprite that has declared the Sprite Variable.

Example: `SpriteVars SpriteIsRotated`

Local

A Local Variable, declared as LocalVars, can only be accessed during the execution of the script that declared it. The variable is only available to the script containing it.

Example:
```
LocalVars LoopCounter
```

ARRAY VARIABLES

Array Variables can store multiple values. The Variable name and scope are declared as usual followed by square brackets that enclose the number of elements that the array variable can contain.

Example:
```
GlobalVars MyArray[5]
```

The Global Variable called MyArray can contain five elements. These individual elements are then indexed sequentially starting at 0, and can be assigned values as follows:

```
MyArray[0] = 5
MyArray[1] = 7
MyArray[2] = 16
MyArray[3] = 4
MyArray[4] = 22
```

To access an element of an Array Variable in QScript, you will need to specify the numeric value in its square brackets.

Example:
```
LocalVars CurrentIndex
CurrentIndex = MyArray[3]
```

Once this script has executed, the variable “CurrentIndex” will contain a value of 4. If you specify a value beyond the range of array values, it may lead to unpredictable results when the movie is running.

You can also use a variable for an array index number.

Example:
```
LocalVars IndexVar, CurrentIndex
IndexVar = 2
CurrentIndex =  MyArray[IndexVar]
```

Once this script has executed, the variable “CurrentIndex” will contain a value of 16. If you specify a value beyond the range of array values in “IndexVar”, it may lead to unpredictable results when the movie is running. Make sure to constrain your values within the range you have allotted for your Array.

Assignments

Assignment statements establish how a variable’s data value is to be set. An Assignment Statement consists of two parts—the variable and an expression.

```
Variable = Expression
```

The expression can be a constant, variable, property, numeric or algebraic equation, or a combination of these components. The expression is evaluated and the result is then assigned to the particular variable.

Example: `CurrentImage = ThisSprite.ImageIndex`

This assigns the current sprite's image index to the variable CurrentImage.

Example: `MyNumber = MyNumber + 1`

This increments the variable by one. Because the expression on the right is evaluated first, you can refer to the variable itself within the expression. This expression can be used in conjunction with other variables to provide incremental values or calculations.

Expressions

Expressions are fundamental to constructing effective QScripts and can take many forms, ranging from literal values to algebraic operations.

Literal Value `SetVolumeTo(125)`

This sets the volume to a value of 125.

Properties `GoToTime(MaxLoadedTimeInMovie)`

This sends the Playhead to the point time in the movie that has been loaded.

Constant `GoToURL($“kCompanyURL”)`

This sends the user to the web page you specified for the kCompanyURL constant in the Defines Window.

Variables `SetImageIndexTo(counterNum)`

This sets the sprite's image index to the numerical value contained in the counterNum variable.

EXPRESSION OPERATORS

You can also use various operators to join expressions together.

+	Addition
-	Subtraction
*	Multiplication
/ or ÷	Division
DIV	Integer division
REM	Remainder after integer division
OR	Binary OR
AND	Binary AND
“(“ and ”)”	Parentheses can override the order of evaluation 1 + 2 * 3 - 4 the result is 3 (1 + 2) * (3 - 4) the result is -3

Note that if you are working in a Windows environment, you should use the “/” for Division as opposed to the “÷” as Windows machines do not read this particular symbol.

BOOLEAN EXPRESSIONS

Boolean expressions are not assigned to variables but are used to determine if a condition is TRUE or FALSE.

Example:

```
(TimesMouseClicked < 5)
```

In addition to the Expression Operators already listed, Booleans also employ these additional operators.

<	Left side is less than the right side
>	Left side is greater than the right side
<= or ≤	Left side is less than or equal to the right side
>= or ≥	Left side is greater than or equal to the right side
!= or ≠	Left side is not equal to the right side

tip

If you are working in a Windows environment, you will need to utilize the two-character operators (i.e., “<=”, “=>”, “!=”) and not the single-character symbols (≤, ≥, ≠).

CONTROL STATEMENTS

Control statements allow your scripts to make decisions based on criteria that you set up, through Boolean expressions, matching values, or looping. Control statements fall into three groups:

Conditional Statements
Switch Statements
Loop Statements

CONDITIONAL STATEMENTS

Conditional statements use Boolean expressions to determine if certain script elements should execute. Conditional statements make use of IF, ELSE, ELSEIF and ENDIF control statements.

Example:

```
IF(MovieTime>1200)
        TrackNamed(“controller”).SetEnabled(TRUE)
ENDIF
```

If the Playhead has passed 1200 (two seconds), the script directing the “controller” track to become enabled will execute. The script will not execute until the MovieTime is greater than 2 seconds. This script is generally contained in a sprite’s Idle Event.

Example:

```
IF(MovieTime>1200)
        SetImageIndexTo($”instructions”)
ELSE
        SetImageIndexTo($”welcome”)
ENDIF
```

If the Playhead has passed 1200, the sprite’s image will be set to the “instructions” image. If the Playhead has not passed 1200, the image will be set to “welcome”.

Example:

```
If(Version >=5 and Version < 6)
        GoToURL("YouHaveQT5.html")
ELSEIF (Version >= 6)
        GoToURL("YouHaveQT6.html")
ELSE
        GoToURL("UpdateQT.html")
ENDIF
```

If the user has QuickTime 5 installed on their computer, the browser will open the "YouHaveQT5.html" page. If the user has QuickTime 6, the "YouHaveQT6.html" page will launch. If the user has neither of these versions, they will be sent to a page informing them they need to update their QuickTime.

SWITCH STATEMENTS

Switch statements are a means of testing if a variable contains the same value as a constant. The script will only execute the case when the constant is true.

Example:

```
MovieVars mKeyCode

mKeyCode = GetEventKey
SWITCH mKeyCode
CASE kReturnKeyCode          // action triggered when
                             Return is hit
ENDCASE
CASE kBackSpaceKeyCode       // action triggered when
                             Backspace is hit
ENDCASE
ENDSWITCH
```

The variable mKeyCode supplies a value (GetEventKey) that is compared to the constants in each of the CASE statements. If the value stored in the variable matches a constant in a CASE statement, the scripts in that CASE statement are executed. At most, only one of the CASE statements will match the variable, though it is possible none will match. In this case, none of the scripts would be executed

LOOP STATEMENTS

Loop statements allow a sequence of statements to be executed a set number of times.

WHILE LOOP

The While Loop executes as long as a condition is true and then exits the script.

Example:

```
y = 0
WHILE (y < 10)
        TrackOfIndex(y).SetEnabled(FALSE)
        y = y + 1
ENDWHILE
```

The TrackOfIndex(y).SetEnabled(FALSE) will be executed for as long as the value of y is less than 10. The variable "y" starts of at a value of zero and then increments itself by one and executes the script until it has reached 10, at which point the script is exited. Thus, the tracks whose Index is between 0 and 9 will be disabled. Note that if the instruction Y = Y + 1 is missing, the condition Y <10 will always be satisfied and the loop will go on indefinitely. This is a common scripting error.

FOR LOOP

FOR loops follow this syntax:

```
FOR Variable = FromValue to ToValue
//Execute a Script
NEXT
```

In a FOR loop, 'FromValue to ToValue' controls how many times the FOR loop will be executed and 'Variable' is a variable that stores the current state of the FOR loop. The value stored in 'Variable' is used to determine if the FOR loop is to continue. The FromValue specifies the starting value that will be set into 'Variable' when the FOR loop first begins. The value in 'Variable' is then incremented by one every time the FOR loop "loops". When the value of the variable exceeds the value of ToValue the loop stops. When this happens, the FOR loop ends and QuickTime proceeds to the scripts, if any, following the NEXT statement.

Example:
```
FOR x = 5 to 10
        TrackOfIndex(x).SetEnabled(FALSE)
NEXT
```

Tracks with Indexes of 5 through 10 will be disabled when this script executes.

Note: The single determining factor in how many times the loop executes is how the loop control expression evaluates. If this expression never returns a FALSE (or if the variable in the FOR loop never exceeds the TO value), an infinite loop will occur and the loop will execute forever. This is a common scripting error.

Comments

Comments are lines of script that are hidden from the QScript compiler. They provide a convenient means for you to make notes and instructions within your script that you can refer to without affecting its execution. It is a good rule of thumb to regularly document your script in detail. This way, if you revisit your project later, you will better understand what you were trying to accomplish in your scripting.

Example:
```
// This is a comment
```

Any text following the double forward slash characters "//" through to the end of the line is ignored by the QScript compiler.

You can enter comments on the same line as your script, provided the "//" is placed after any critical code you want to have execute.

Example:
```
LocalVars count    // declare a local counter variable
```

The QScript compiler recognizes and runs the LocalVars count declaration, but ignores the text following the "//".

You can also use Comments to temporarily disable script statements. Type the “//” before a line of script and it will be ignored by the QScript compiler.

To make multi-line comments, use “/*” and “*/” to enclose the comments you want ignored.

Example:

```
/*
ignore this line
and this line
*/
```

Continuous Lines

If you need to put more than one QScript statement on a single line (i.e., FSCommands), use a semicolon where the line would normally break.

Example:

```
SpriteVars test; test = 0
```

Without the semicolon, you would have to write the QScript like this:

```
SpriteVars test
test = 0
```

LINE CONTINUATIONS

At times, your QScript commands can become quite long and will extend beyond the maximum viewing area of the Script Editor field. You can view the code by using the scroll bar, or you can use line continuations to keep the scripts broken into smaller, viewable pieces. Use the “\” character to continue a line of QScript that is separated onto multiple lines. Make sure that you choose a logical place to insert the “\” so that you are not breaking a string or target, etc. For instance, a good place to place a line continuation would be after the dot:

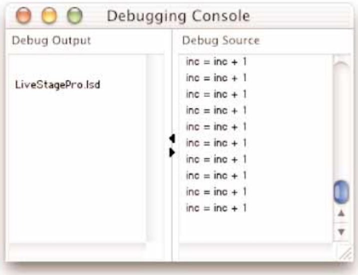
Example:

```
MovieName(“My long movie name”).\
TrackNamed(“My Long track name”).\
SpriteNamed(“My Long Sprite Name”).\
ExecuteEvent($”My Long Event Name”)
```

DEBUG CONSOLE WINDOW

The Debug Console Window, accessed via the Window>Show Debugging Console, or through the 🐞 is a troubleshooting aid for your scripting. The Debug Console Window displays information while your movie is running in LiveStage Professional. The output it produces can help you track down problems in your scripts in one of two ways. You can either view the scripts as they execute, or you can send your own debugging messages through the DebugStr (Debug String) QScript action.

To have the Debug Console Window display the QScript statements that are being executed, access the Preferences window through the LiveStage Professional menu (Command-,), and in the Compiler tab, check “Default to Show Source in Debugger”. When you run, or preview, your movie, the QScript statements within it will display as they are executed in the right side of the Debug Console Window.



LiveStagePro.lsd

```
inc = inc + 1
inc = inc + 1
inc = inc + 1
inc = inc + 1
inc = inc + 1
inc = inc + 1
inc = inc + 1
inc = inc + 1
inc = inc + 1
inc = inc + 1
```

#debug

The #debug preprocessor directive can also be used to show or hide portions of your script, helping you to isolate possible problems.

1. In LiveStage Professional's Preferences Window (Command -;), ensure "Default to Show Source in Debugger" is selected.

2. In the Script Edit Window of the sprite's sample, enter the following:

```
#debug off
GlobalVars inc
SpriteVars tempstring
#debug on
inc = inc + 1    // increment the counter
#debug off
SetString(tempstring, "counter value: ")   // create the string
AppendString(tempstring, inc, tempstring)  // append the counter
                                           // value, inc to the string
```

   The '#debug off' turns off the display of source code in the Debugger Console and the '#debug on' turns it back on again.

3. Click the Debug Window button to open the Debug Console.

4. Click the Preview button to run your movie.

   The Debug Console window shows:

```
inc = inc + 1    // increment the counter
```

You can also use this technique without the "Default to Show Source in Debugger" preference set. Just use the #debug on and #debug off to display the lines of script you want to see in the Debug Console.

DebugStr

You can also put debug messages into your scripts to indicate which parts of your script are being executed. Any information contained in the DebugStr command will display in the left side of the Debug Console Window. The DebugStr is not dependent on the “Default to Show Source in Debugger” preference.

1. Create a sprite (see Chapter 7—The Sprite Track, page 158).
2. In the Script Edit Window of the sprite’s sample, enter the following on the Mouse Click Event Handler:

   ```
   DebugStr(“Sprite was clicked on”)
   ```

3. Click the Debug Window to view the Debug Console and the Preview your movie.
4. Click on the sprite.

   The Debug Console Window shows the “Sprite was clicked on” message.

You can also use the DebugStr to display both a message and the value of a variable within your script.

1. Create a sprite.
2. In the Script Edit Window of the sprite sample select the “Frame Loaded” Event Handler and in the Script Editor field enter the following:

   ```
   GlobalVars inc
   inc = 0
   ```

   This defines a variable that will act as a counter.

3. In the “Mouse Enter” Event Handler, enter:

   ```
   GlobalVars inc            // inc will be the counter
   SpriteVars tempstring     // increment the counter
   inc = inc + 1             // create the string and then append
                             // the counter value to it
   SetString(tempstring, “counter value: “)
                             //the variable, tempstring, now has
                             the value of “counter value:”
   AppendString(tempstring, inc,tempstring)
                             // the variable, tempstring, now
                             has the value of  “Counter Value:
                             Inc” where inc is a numeric value.
   DebugStr(tempstring)      // send the debug string to the
                             debug window
   ```

   This increments the counter and then displays its value after a “counter value” message.

4. Open the Debug Console and Preview your movie.

5. Mouse over the sprite.

   The Debug Console displays “counter value: 1”

6. Mouse over the sprite again.

   The Debug Console displays “counter value: 2”

   Each time the mouse is moved into the sprite, the current value of the counter will be updated in the Debug Console Window.

For more information on the DebugStr QScript action, please refer to the QScript Reference.

Chapter 17

Behaviors

OVERVIEW

You do not have to be fluent in QScript to produce great interactivity in your project, you can simply drag and drop a Behavior. Behaviors are pre-made QScripts that you can drag and drop into your projects to add interactivity without writing a single line of code. As your skills in QScript progress, you can also create your own Behaviors to use in your projects and share with the rest of the developer community. Behaviors really come in handy when you want to save yourself the trouble of repeatedly writing the same code or continually cutting and pasting scripts.

PRE-AUTHORED BEHAVIORS

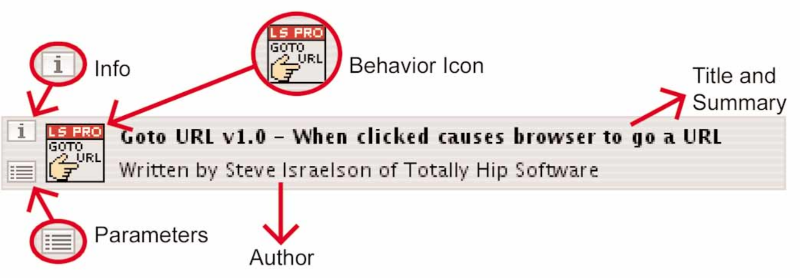
LiveStage Professional provides a number of ready-made Behaviors that have been created by Totally Hip team members and developers within the LiveStage Professional community. These behaviors can be found in the Scripts tab of the Library window. Additional Behaviors can be found on the LSDN (LiveStage Developer Network) web site at http://www.totallyhip.com/lsdn.

USING BEHAVIORS

Behaviors are added to sprites to make them 'behave' in a certain way. Some Behaviors are automatic and some need to be triggered by a specific event. Behaviors accomplish individual tasks such as making a sprite draggable or automatically panning a VR. You can also use several Behaviors in combination with one another on a single sprite to achieve complicated effects—all without writing a single line of QScript.

Behavior Components

Behaviors are text documents. You will not see the Behavior's actual QScript unless you use a text editor to open the Behavior (.lsb) document. In LiveStage Professional, the Behavior only provides information about itself and the means to adjust parameters in its script.



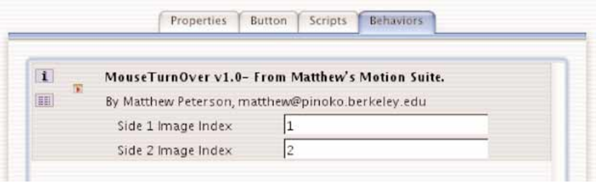
Goto URL v1.0 – When clicked causes browser to go a URL
Written by Steve Israelson of Totally Hip Software

INFO

Click on this icon to view information on the author, a description of what the Behavior does, and instructions on how to customize it for your use.

PARAMETERS

Some Behaviors allow you to enter various parameters to tailor how you would like the Behavior to act. These Behaviors provide a Parameters icon that you can click to open fields where you can alter any available options.



MouseTurnOver v1.0– From Matthew's Motion Suite.

By Matthew Peterson, matthew@pinoko.berkeley.edu

Side 1 Image Index: 1

Side 2 Image Index: 2

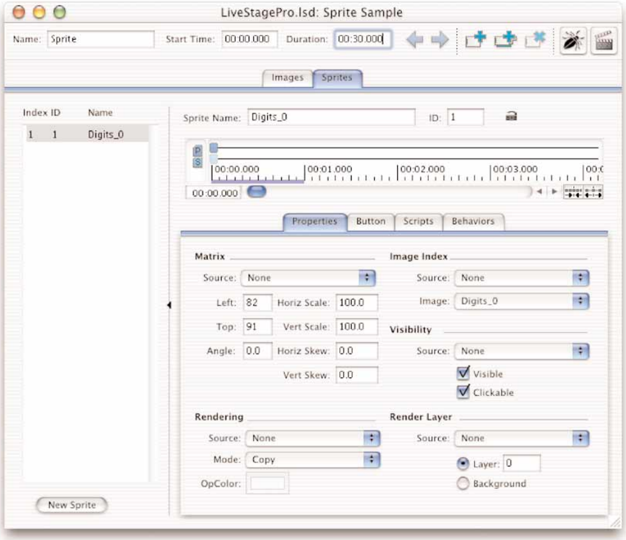
When you click the Parameters icon, you will be presented with a list of all the settings that you can adjust for this Behavior. Settings can take the form of numbers, names, IDs, as well as Defines (see Chapter 16—Introduction to QScript, page 321). You enter a Define in a Behavior by preceding its name with a "$", just as you would in a QScript. This is also how you can refer to image names.

Some Behaviors call Custom Events (see Custom Events in Chapter 16—Introduction to QScript, page 314) when certain conditions are met. You can enter either the Custom Event's ID number or its name like $myCustomEvent.

Adding a Behavior to a Sprite

To add a Behavior to a sprite:

1. Create a Sprite Track.
2. From the Images folder in the Global Library, drag one of the "Digits" images to the Sprite Track on the Stage.
3. Double-click on the new sprite on the Stage or in the Timeline to open the Sprite Sample Properties window.
4. In the Duration field, enter 00:30.000.



Name: Sprite
Start Time: 00:00.000
Duration: 00:30.000

1 1 Digits_0

Sprite Name: Digits_0
ID: 1

00:00.000

Matrix Source: None
Left: 82 Horiz Scale: 100.0
Top: 91 Vert Scale: 100.0
Angle: 0.0 Horiz Skew: 0.0
Vert Skew: 0.0

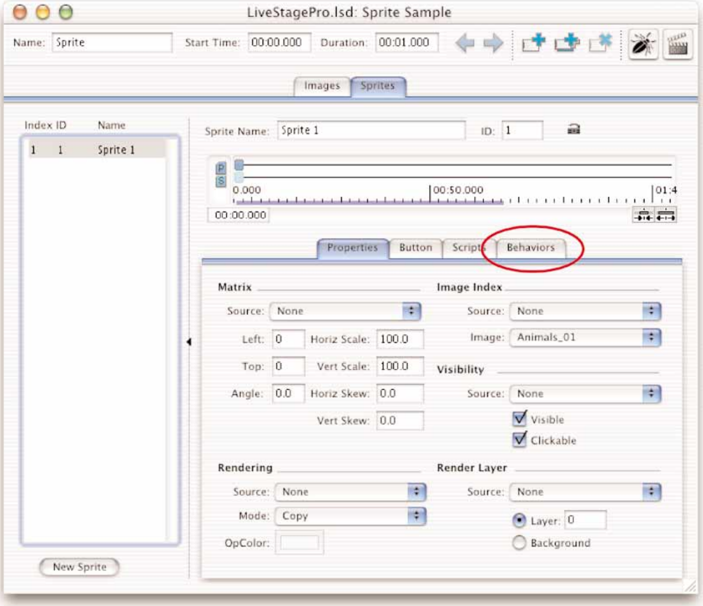
Image Index Source: None
Image: Digits_0

Visibility Source: None
☑ Visible
☑ Clickable

Rendering Source: None
Mode: Copy
OpColor:

Render Layer Source: None
● Layer: 0
○ Background

5. Click on the Behaviors tab.



LiveStagePro.lsd: Sprite Sample

Name: Sprite Start Time: 00:00.000 Duration: 00:01.000

Index ID Name
1 1 Sprite 1

Sprite Name: Sprite 1 ID: 1

Matrix
Source: None
Left: 0 Horiz Scale: 100.0
Top: 0 Vert Scale: 100.0
Angle: 0.0 Horiz Skew: 0.0
Vert Skew: 0.0

Image Index
Source: None
Image: Animals_01

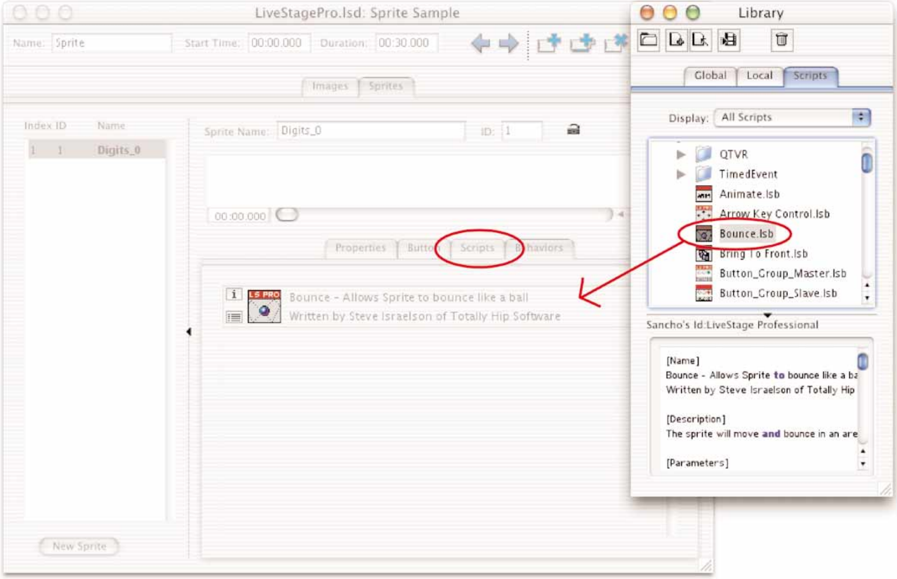
Visibility
Source: None
☑ Visible
☑ Clickable

Rendering
Source: None
Mode: Copy
OpColor:

Render Layer
Source: None
● Layer: 0
○ Background

6. Click on the Scripts tab of the Library Window.

7. In the Behaviors Folder, click on Bounce.lsb and drag it to the Behaviors tab.



LiveStagePro.lsd: Sprite Sample

Name: Sprite Start Time: 00:00.000 Duration: 00:30.000

Index ID Name
1 1 Digits_0

Sprite Name: Digits_0 ID: 1

Bounce - Allows Sprite to bounce like a ball
Written by Steve Israelson of Totally Hip Software

Library

Display: All Scripts

Sancho's Id:LiveStage Professional

[Name]
Bounce - Allows Sprite to bounce like a ba
Written by Steve Israelson of Totally Hip

[Description]
The sprite will move and bounce in an are

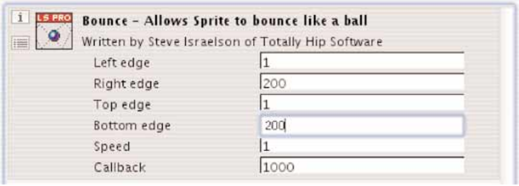
[Parameters]

8. Click on the Parameters icon ☰.

The parameters that can be modified for the Behavior are displayed.

9. Change the Right and Bottom Edge parameters to 200 and the Speed parameter to 4.

As the dimensions of the Sprite Track are 200x200 pixels, this will allow the sprite to bounce around the entire track. You could also set the bounce region dimensions to exceed the boundaries of the Sprite Track. This would cause the sprite to bounce in and out of the visible area.



Bounce - Allows Sprite to bounce like a ball
Written by Steve Israelson of Totally Hip Software

Left edge	1
Right edge	200
Top edge	1
Bottom edge	200
Speed	1
Callback	1000

10. Click on the 🎬 to preview your movie.

    The image bounces around the movie.

    Close the Preview and the Sprite Sample Properties window.

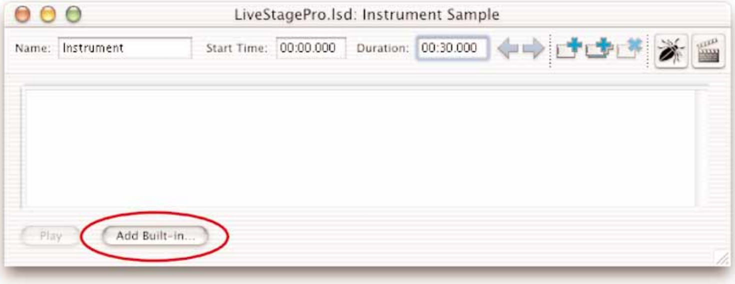
Triggering a Custom Event from a Behavior

You can enhance this Behavior by using the “Callback” parameter to trigger an event each time the image bounces. You can use this to trigger a sound, for instance.

1. Click on the 🎵 to create an Instrument Track.

2. Double-click the Instrument sample to open the Instrument Sample Properties window .

3. Click on “Add Built-in...”



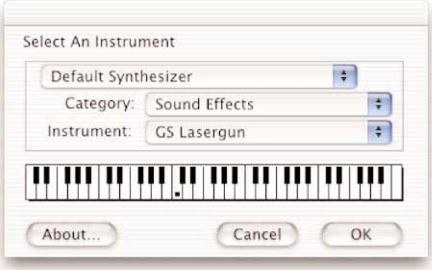
LiveStagePro.lsd: Instrument Sample

Name: Instrument Start Time: 00:00.000 Duration: 00:30.000

The Instrument Selection window opens.

4. From the Category popup menu, select “Sound Effects”.

5. From the Instrument popup menu, select GS Lasergun and click OK to close the window.



Select An Instrument

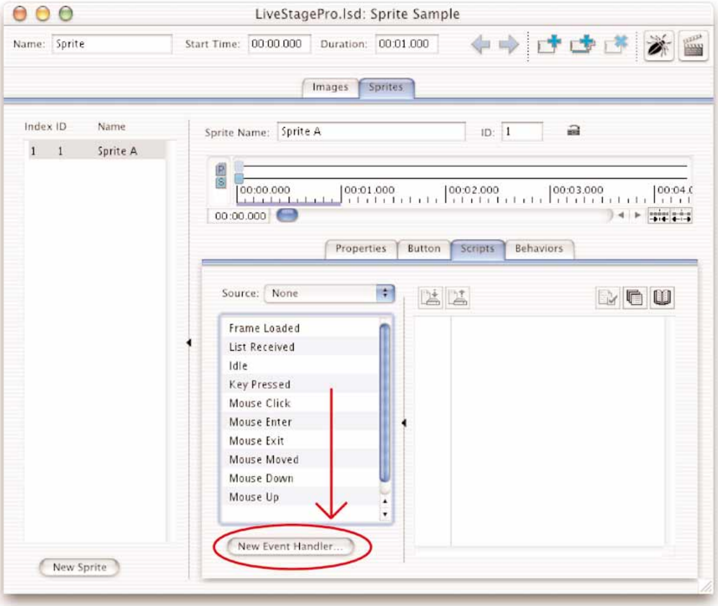
Default Synthesizer
Category: Sound Effects
Instrument: GS Lasergun

The GS Lasergun instrument appears in the Instrument List with an ID of 1.

6. Close the Instrument Sample Properties window .

7. Double-click on the Sprite sample to open the Sprite Sample Properties window again and click on the Scripts tab.

8. Click the “New Event Handler” button.


LiveStagePro.lsd: Sprite Sample

Name: Sprite Start Time: 00:00.000 Duration: 00:01.000

Sprite Name: Sprite A ID: 1

Source: None

The Edit Event Handler dialog opens.

9. Name the event “Bounce” and give it an ID of 5000.

Click OK to close the dialog

The new Custom Event shows up in bold in the Event Handler List.

10. Click in the Script Edit window and enter the following QScript:

TrackNamed(“Instruments 1”).PlayNote(1, 0, 60, 100, 1)

To learn more about the Instrument Track and to understand the various parameters for the PlayNote command, see Chapter 13—Additional LiveStage Professional Tracks, page 255.

11. Click on the Behaviors tab and open the parameters for the Bounce Behavior again.

12. In the Callback field, enter the Custom Event ID number: 5000. Alternately, you could also enter the Custom Event Name: $Bounce.

13. Click on the 🎬 to preview your movie.

The image bounces around the movie, making a noise with every bounce.

tip

To remove a Behavior, click on it in the Behaviors tab of the Sprite Sample Properties window to select it, and press the delete key.

AUTHORING BEHAVIORS

Once you are familiar with QScript, you will find that behaviors are simple to author. You just need to follow a simple format that will package your QScript into a single (.lsb) text file separated into different sections. In this format, the Behavior's name, description, and parameters have their own sections, followed by individual sections for each Event Handler your Behavior uses. To get an idea of what Behaviors look like on the 'inside', open one in a text editor. It will look much like the following:

```
[Name]
Bounce—Allows sprite to bounce like a ball
Written by Steve Israelson of Totally Hip Software

[Description]
The sprite will move and bounce in an area

Revision History:
V 1.0 1/1/01 Initial release
V 1.1 1/2/01 Added Callback parameter

Usage Instructions:
Set bounding box dimensions.
Callback will trigger on sprite bounce.

Parameters:
Speed parameter determines how fast the sprite will move across the bounding area.

Custom Events:
This behavior uses no custom events.

[Parameters]
Left edge, THW_BOUNCE_LeftEdge,1
Right edge, THW_BOUNCE_RightEdge,1
Top edge, THW_BOUNCE_TopEdge,1
Bottom edge, THW_BOUNCE_BottomEdge,1
Speed (1-10), THW_BOUNCE_Speed,1,1,10
Callback, THW_BOUNCE_EVENT,1000

[Frame Loaded]      Frame Loaded Handler
SpriteVars THW_BOUNCE_xvel THW_BOUNCE_yvel
THW_BOUNCE_xvel = Random(1,$THW_BOUNCE_Speed)
THW_BOUNCE_yvel = Random(1,$THW_BOUNCE_Speed)

[Idle]              Idle Handler
SpriteVars THW_BOUNCE_xvel THW_BOUNCE_yvel

// Move the sprite
MoveBy(THW_BOUNCE_xvel, THW_BOUNCE_yvel)
```

```
IF (BoundsLeft < $THW_BOUNCE_LeftEdge)
                MoveBy(($THW_BOUNCE_LeftEdge—BoundsLeft), 0)
                THW_BOUNCE_xvel = -THW_BOUNCE_xvel;
                ExecuteEvent($THW_BOUNCE_EVENT)
ELSEIF (BoundsRight > $THW_BOUNCE_RightEdge)
                MoveBy(($THW_BOUNCE_RightEdge—BoundsRight), 0)
                THW_BOUNCE_xvel = -THW_BOUNCE_xvel;
                ExecuteEvent($THW_BOUNCE_EVENT)
ENDIF
IF (BoundsTop < $THW_BOUNCE_TopEdge)
                MoveBy(($THW_BOUNCE_TopEdge—BoundsTop), 0)
                THW_BOUNCE_yvel = -THW_BOUNCE_yvel;
                ExecuteEvent($THW_BOUNCE_EVENT)
ELSEIF (BoundsBottom > $THW_BOUNCE_BottomEdge)
                MoveBy(($THW_BOUNCE_BottomEdge—BoundsBottom), 0)
                THW_BOUNCE_yvel = -THW_BOUNCE_yvel;
                ExecuteEvent($THW_BOUNCE_EVENT)
ENDIF
```

Behavior Sections

The text file that comprises the Behavior is split into several sections. The text enclosed in the square brackets (i.e. [Name]) determines what will be contained in the section that follows it. You can also add a comment on the same line as the section header.

tip

Long sentences are not automatically word-wrapped in the Behaviors tab, so it is a good idea to keep the length of each line in this section to about 50 characters. This way your users will not have to enlarge the Behaviors tab to fully view your instructions.

[NAME]

This is the section where you provide the name, version number, and brief description of the Behavior. You can also add the author's name and contact number in this section.

Example: `[Name]`
Bounce—Allows Sprite to Bounce Like a Ball
Written by Steve Israelson of Totally Hip Software

[Description]

This section is where you provide detailed information about the Behavior. This is the information that will appear when the user clicks on the ℹ in the Behaviors tab.

The information you include in this section should clarify what the Behavior does as well as offer instructions on how the Behavior should be used.

Description

Detail what the Behavior does.

Example: `[Description]`
The sprite will move and bounce in an area

Revision History

Provide a list of the Behavior's versions, dates, and revision notes.

Example: Revision History:
V 1.0 1/1/01 Initial release
V 1.1 1/2/01 Added Callback parameter

Usage Instructions

Describe how to attach the Behavior to a sprite for it to function properly.

Example: Usage Instructions:
Set bounding box dimensions.
Callback will trigger on sprite bounce.

Parameters

Describe each of the Behavior's parameters.

Example: Parameters:
Speed parameter determines how fast the sprite will move across the bounding area.

Custom Event

List the custom events that the Behavior provides. Detail and describe each event, parameters used by the event, as well as any return variables or errors.

Example: Custom Events:
This behavior uses no custom events.

[PARAMETERS]

This is the section where you list the parameters you are including that will enable the user to customize your Behavior. List each parameter on a separate line, arranged as follows:

Parameter Name (value range), Define, Default Value, Minimum Value, Maximum Value

```
[Parameters]
Left edge, THW_BOUNCE_LeftEdge,1
Right edge, THW_BOUNCE_RightEdge,1
Top edge, THW_BOUNCE_TopEdge,1
Bottom edge, THW_BOUNCE_BottomEdge,1
Speed (1-10), THW_BOUNCE_Speed, 1, 1, 10
Callback, THW_BOUNCE_EVENT,1000
```

Parameter Name

This label will appear in the Behavior parameters window. Provide a range of values in parentheses with the Name to give the user an idea of what values are possible for the particular parameter.

Example: **Speed (1-10)**, THW_BOUNCE_Speed, 1, 1, 10

Define

The Define is the name that will be used to hold this parameter's value. As with any other Defines (see Chapter 16—Introduction to QScript), you can refer to it in your Behavior script by preceding the name with a "$" .

Make sure to select a name that is unique and not contained in other Behaviors, as this name will become global to your sprite. If another Behavior attached to the sprite has the same parameter name defined, the two Behaviors will conflict. Developers often prefix their Defines with their initials to prevent such issues. Note that Totally Hip has reserved the prefix "THW_" for variable and define names.

Example: Speed (1-10), **THW_BOUNCE_Speed**, 1, 1, 10

Default Value, Minimum Value, Maximum Value

These components are not mandatory, but if they are used, they must appear in order. In other words, if you are going to assign a Minimum Value, you will need to specify a Default Value first.

The Default Value can be textual or numerical, and will be the value that is used if the user does not enter a value. While this is optional, it is strongly recommended that you always include a Default Value. The Minimum and Maximum Values are the range of numbers that the user can enter.

Example: Speed (1-10), THW_BOUNCE_Speed, **1, 1, 10**

[EVENT HANDLERS]

Event Handlers are separated into their individual sections and are defined by the name of the event handler in square brackets.

```
[Mouse Down]
[Mouse Up]
[Mouse Click]
[Mouse Enter]
[Mouse Exit]
[Mouse Moved]
[Frame Loaded]
[Key Pressed]
[Idle]
[List Received]
```

Each section includes the QScript that the specific handler will execute. This QScript is identical to what you would write in a Script Editor window.

Example: `[Idle]` Idle Handler
SpriteVars THW_BOUNCE_xvel THW_BOUNCE_yvel

You can also use Custom Events. Provide an ID number followed by the Name you want to use for the custom Handler.

Example: [123456 Fade In]

The Custom Event Handler ID is 123456 and the name is Fade In. You can refer to this Event Handler in your scripts either by the ID or by the name, as $Fade In.

Guidelines

You can lessen the risk of your Behaviors conflicting with each other and with other variables in your project in a number of ways.

VARIABLES

It is imperative that you exercise care when naming the variables and Defines in your Behaviors. You must ensure that each variable and Define name is unique and not duplicated by other Behaviors. The variable and Define names within your Behaviors are accessible by all of the scripts within the Behavior as well as the user's scripts within this sprite.

For instance, imagine you have a Behavior that rotates your sprite clockwise using a variable called RotateSprite. If you had a script elsewhere in your sprite also with a variable called RotateSprite that was used to rotate the sprite counter-clockwise, the two would obviously conflict.

While Using Global Variables may be advantageous for some advanced Behaviors, you should almost exclusively use Sprite or Local variables (see Chapter 16—Introduction to QScript, page 323), as opposed to Global Variables. SpriteVars are local to the sprite that contains them, so they will not conflict with other sprites that may have the same variable names. On the other hand, GlobalVars are shared by all sprites in the Sprite Track, so Behaviors that are applied to different sprites but use the same Variable names would conflict with one another.

Naming Conventions

Creating unique names for your variables is an obvious necessity when names are shared. You can employ certain naming conventions to decrease risk of a conflict as well as provide a means to distinguish one type of variable from another.

Example: YourInitials_BehaviorName_VariableName

When declaring a Global Variable, this naming convention would result in this:

```
GlobalVars THW_MyBehavior_Counter
```

The “THW_” prefix identifies the Behavior's author—in this case Totally Hip Software. The “MyBehavior_” is the name of this Behavior and the last section “Counter” is the actual variable name.

SCOPE IDENTIFICATION

Prefix your variables with a single letter that stands for the variable's scope. This will help you easily differentiate between different types of variables in the same script.

Example:

```
MovieVars mTHW_MyBehavior_Counter
GlovalVars gTHW_MyBehavior_Value
SpriteVars sTHW_MyBehavior_Status
```

The lower-case "m", "g" and "s" denote Movie, Global, and Sprite variables respectively.

PASSING PARAMETERS

In order to pass parameters to Custom Events, you need to utilize Global or Sprite variables, making sure to name the variables very carefully. When creating Behavior scripts that will be receiving and returning values there are a few additional guidelines you should follow.

CACHE VALUE LOCALLY

When you are passing values into a Custom Event in your Behavior, you should copy the value received to a Local Variable. This way, the parameter value in your Behavior will not be changed when the Custom Event is executed.

Example

```
// Calling Script on Event Handler
GlobalVars gTHW_MyBehavior_Parameter1, gTHW_MyBehavior_Parameter2

gTHW_MyBehavior_Parameter1 = 100
gTHW_MyBehavior_Parameter2 = 200
SpriteNamed("Target").ExecuteEvent($CustomEvent)

// Custom Event
GlobalVars gTHW_MyBehavior_Parameter1, gTHW_MyBehavior_Parameter2
LocalVars param1, param2

param1 = gTHW_MyBehavior_Parameter1
param2 = gTHW_MyBehavior_Parameter2
```

Transferring the Global Variable parameter values to Local Variables keeps the values constant within the target Custom Event handler. This eliminates the risk of undesirable results from other Event Handlers altering the Global Variables.

Return Variables

If your Custom Event is going to return a value, define a Global variable for this purpose:

Example:

```
GlobalVars gTHW_MyBehavior_Return
```

Through parameter passing, you can use Custom Events to return a value of a complex calculation or even Error codes:

Example:

```
// CalcSquare Custom Event Handler
// Calculate the Square of the input parameter

GlobalVars gTHW_MyBehavior_SquareParameter, gTHW_MyBehavior_SquareResult
LocalVars squareParam

squareParam = gTHW_MyBehavior_SquareParameter
gTHW_MyBehavior_SquareResult = squareParam * squareParam

// Calling Script
GlobalVars gTHW_MyBehavior_SquareParameter, gTHW_MyBehavior_SquareResult

gTHW_MyBehavior_SquareParameter = 1234
SpriteNamed("Target").ExecuteEvent($CalcSquare)
// gTHW_MyBehavior_SquareResult now contains
the value of 1234 * 1234
```

The result could easily have been a numeric value indicating 0=Success, 1=General Failure, and so on. Use the value 0 to indicate success. Any value other than 0 would then indicate a failure. The non-zero value will indicate the error code.

CUSTOM EVENTS

You can use Custom Events to trigger your Behavior's functionality. Chapter 16—Introduction to QScript details Custom Events on page 313. As with variables, Event Names and IDs are also shared so ensure your names are unique.

Event IDs

As it is critical that all Event ID's are unique, Totally Hip Software assigns ID's on request to minimize ID conflicts between published Behaviors. If you write a Behavior for public use contact Totally Hip Software to receive a range of Event ID numbers that you can use. To receive your ID allocation, e-mail your request to livestage@totallyhip.com.

Getting and Setting Parameters through a Custom Event

Behavior parameters cannot be modified while the movie is running. For more flexibility at run-time, you can set a Behavior's parameter values by a QScript. By using parameter passing, you can create Custom Events that accept parameter values and then set them into Sprite Variables to be used exclusively within the Behavior:

Example: [Frame Loaded Event]

```
SpriteVars sTHW_MyBehavior_Parameter1

//Initialize the variable with the value from the Behavior Parameters

sTHW_MyBehavior_Parameter1 = $Behavior_Parameter1
```

This sets up the Sprite Variable with the value from the Behavior Parameters. All Behavior operations will then use this variable.

```
[1000 SetParameter1]      Custom Event

GlobalVars gTHW_MyBehavior_Parameter1
SpriteVars sTHW_MyBehavior_Parameter1

sTHW_MyBehavior_Parameter1= gTHW_MyBehavior_Parameter1

[1001 GetParameter1]             Custom Event

GlobalVars gTHW_MyBehavior_Parameter1

SpriteVars sTHW_MyBehavior_Parameter1

gTHW_MyBehavior_Parameter1= sTHW_MyBehavior_Parameter1
```

The above two Custom Events handle the setting and getting of the Sprite Variable sParameter1. Because they provide a way to access the working parameters of a Behavior, they are called Accessors.

Set these Accessors prior to activating the Behavior (many Behaviors provide a Custom Event Handler that starts the Behavior) to override the Behavior's parameter settings that are set during script compilation.

To call these events, for example using an Idle Event Handler:

```
[Idle]

//pass the parameter

GlobalVars gTHW_MyBehavior_Parameter1
SpriteVars sTHW_MyBehavior_Parameter1

gTHW_MyBehavior_Parameter1=100

//set the parameter

ExecuteEvent($SetParameter1)

//get the parameter

ExecuteEvent($GetParameter1)

// gTHW_MyBehavior_Parameter1 now equals the setting
```

Initialize Event

Instead of putting all your initialization code in a Frame Loaded event, you should put it in a Custom Event and then call this event from the Frame Loaded event. This way, your behavior can be reset to its default settings at any time by calling the custom event.

Use of Standard Events

When you have a script that you want to be executed by a standard Event Handler such as Idle, Mouse Click, Frame Loaded and so on, use only Local Variables and place the script into a Custom Event. Then call this Custom event from the standard Event Handler. This allows the action to be easily triggered by other scripts simply by executing the custom event.

Chapter 18

XML and QTLists

WHAT IS XML?

XML (Extensible Markup Language) is a platform-independent format for describing, structuring, storing, and sending data. XML files are simple text documents that structure data into a series of tags, much like HTML. Elements have both Start (<element>) and End (</element>) tags.

Example: `<title>`This is the title`</title>`

Unlike HTML however, XML's tags are not predefined and are therefore completely customizable. This allows a great degree of control and flexibility in describing data elements and dictating how they should be categorized. In XML, data is categorized hierarchically, allowing you to place elements within other elements.

Example:

```
<year>
          <month>
                    <day></day>
          </month>
</year>
```

XML documents must be 'well-formed' to operate. Here are a few rules that will help you to build well-formed XML:

- All elements must have closing tags.

  Example: `<element>`This is valid`</element>`

- Elements must be properly nested.

  Example:

  ```
  <element>
            <sub-element>This is properly nested
            </sub-element>
  </element>

  <element>
            <sub-element>This is NOT properly nested
            </element>
  </sub-element>
  ```

- As opposed to HTML, XML tags are case sensitive. Therefore, your opening and closing tags must be written in the same case.

  Example: `<Element>`This is not valid`</element>`
  `<element>`This is valid`</element>`

- Element names cannot contain spaces. Use an underscore to separate words instead.

  Example: `<my_element>`

- Unlike HTML, programs processing XML documents will stop if a validation error is encountered. It is a good rule of thumb to validate your XML documents. There are a number of free XML validation sites online that you can use for this purpose.

It is advisable that you gain at least a basic understanding of XML prior to using it with LiveStage Professional.

Storing LiveStage Professional Projects in XML

Your entire LiveStage Professional project can be stored as an XML file. Your LiveStage Professional project can then be edited using any available text editing application, without the need for additional tools. Storing your project as XML allows you to compare differences between two projects. Additionally, you can generate changes to the project programmatically (usually on a server), and then create a movie from this new XML file.

To store your LiveStage Professional project as an XML document, you simply export it as XML:

1. Create a new LiveStage Professional project from the File menu.

2. From the File menu, select Export->XML...

   A File Save dialog displays.

3. Enter a name for your XML project file, and ensure the “.xml” extension is added.

   Example: MyProject.xml

   You can view the new XML document in a text editor.

tip

Be careful when altering an XML project file. Altering the document needs to be precise or it simply will not work.

DECIPHERING THE LIVESTAGE PROFESSIONAL XML PROJECT FILE

Here is what you will see if you open the newly exported document with a text editor:

```
<LSProject>
        <Movie>
                <Movie_XML_File>
                        <Media_File_Frame>0</Media_File_Frame>
                        <Media_File_Type>1852796517</Media_File_Type>
                        <Media_File_Folder_ID>-1</Media_File_Folder_ID>
                        <Media_File_Spec>
                                <VRefNum>0</VRefNum>
                                <ParID>0</ParID>
                                <FileName>””</FileName>
                        </Media_File_Spec>
                </Movie_XML_File>
                <Output_Name>”untitled”</Output_Name>
                <Window_Left>15</Window_Left>
                <Window_Top>67</Window_Top>
                <Window_Width>555</Window_Width>
                <Window_Height>390</Window_Height>
                <Window2_Left>0</Window2_Left>
                <Window2_Top>0</Window2_Top>
                <Window2_Width>0</Window2_Width>
                <Window2_Height>0</Window2_Height>
                <Window_Scale>0</Window_Scale>
        <Play_Full_Screen>false</Play_Full_Screen>
        <Play_All_Frames>false</Play_All_Frames>
        <Play_Selection_Only>false</Play_Selection_Only>
        <Copy_Protect>false</Copy_Protect>
        <Compress_Resource>true</Compress_Resource>
        <QT_Version>5</QT_Version>
        <Controller_Type>1852796517</Controller_Type>
        <AutoPlay>false</AutoPlay>
        <Loop>1313820229</Loop>
        <Poster_Time>
            <Time>0</Time>
            <TimeScale>600</TimeScale>
        </Poster_Time>
        <Preview_Time>
            <Time>0</Time>
            <TimeScale>600</TimeScale>
        </Preview_Time>
        <Preview_Duration>
            <Time>0</Time>
            <TimeScale>600</TimeScale>
        </Preview_Duration>
        <Poster_Box>
            <Left>0</Left>
            <Top>0</Top>
            <Right>0</Right>
            <Bottom>0</Bottom>
        </Poster_Box>
        <Time_Scale>600</Time_Scale>
        <Auto_Alternates>true</Auto_Alternates>
        <Tag_By_ID>false</Tag_By_ID>
    </Movie>
    <DocPreferences>
        <Current_Time>
            <Time>0</Time>
            <TimeScale>600</TimeScale>
        </Current_Time>
    </DocPreferences>
    <Version>4</Version>
</LSProject>
```

These are the tagged elements generated from a blank project. Projects containing media and scripting will generate considerably larger XML documents. The element tag `<LSProject>` comprises the entire contents of the XML file. All other elements in the XML project file are housed hierarchically within the LSProject element. Most of what you see in this example is taken from the Document Settings window in the LiveStage Professional project file.

To help you interpret and edit your LiveStage Professional XML project file, in addition to the rules listed in the “What is XML” section, there are a few principles to follow:

- Hexadecimal data must be enclosed in quotes

- File names must be enclosed in quotes

- Tags for data like colors are broken into their R, G, and B components and these items must be in order.

- Fixed Point or Point data must be in the following form:

  ```
  <tag_name><X>xposition</X><Y>yposition</Y></tag_name>
  ```

  Example:

  ```
  <Image_Registration_Point><X>0</X><Y>0</Y></Image_Registration_Point>
  ```

While it is possible to edit your project by editing the XML file, it is recommended that you perform any modifications to your project within the LiveStage Professional environment.

ELEMENT TAGS

The tags in the exported LiveStage Professional XML document define the information required to create the various objects and contents in the movie.

<Movie>

Anything within the <Movie> element itemizes the attributes of the LiveStage Professional movie. Within the root element <Movie>, you will find information concerning the QuickTime window size and settings.

<Track>

Each track within the LiveStage Professional project appears in the XML document within the <Track> and </Track> tags. Depending on the type of track being represented, different information elements are described. The elements within the <Track> element correspond to the settings found in the Track Inspector and Properties window, as well as the Sample Properties window in LiveStage Professional.

COMMON SUB-ELEMENT TAGS

There are two sub-element groups common to all tracks in the <Track> element. The first describes information about the image files used to provide the Matte effect for the track.

Example:

```
<Track_Matte_File>
          <Media_File_Frame>0</Media_File_Frame>
          <Media_File_Type>1852796517</Media_File_Type>
          <Media_File_Folder_ID>-1</Media_File_Folder_ID>
          <Media_File_Spec>
                    <VRefNum>0</VRefNum>
                    <ParID>0</ParID>
                    <FileName>""</FileName>
          </Media_File_Spec>
</Track_Matte_File>
```

As no Matte File has been associated with this track, there is no <FileName> value. The second sub-element group provides information on the QTList file attached to the track. Again, no QTList file is attached to the track so there is no <FileName> value.

Example:

```
<Track_XML_File>
          <Media_File_Frame>0</Media_File_Frame>
          <Media_File_Type>1852796517</Media_File_Type>
          <Media_File_Folder_ID>-1</Media_File_Folder_ID>
          <Media_File_Spec>
                    <VRefNum>0</VRefNum>
                    <ParID>0</ParID>
                    <FileName>””</FileName>
          </Media_File_Spec>
</Track_XML_File>
```

TRACK SAMPLE SUB-ELEMENT TAGS

The information described in the Track Sample sub-element group pertains to media data that is specific to individual tracks. There can be multiple groups of track sample sub-elements, corresponding to multiple samples being contained within a track. Each track sample has its own start and end tags that describe the type of sample being stored. The tag names for the track samples used in LiveStage Professional are:

Sprite_Sample
Text_Sample
Effect_Sample
Picture_Sample
Color_Sample
Movie_Sample
Skin_Sample_Parent
Tween_Sample
Music_Sample
External_Track_File
Flash_Media_File
External_Movie_File

TRACK INFORMATION

The Track Information sub-elements describe how an individual track is displayed and positioned, as well as providing additional track information. The track ID is contained in the `<Track_Type>` tag within the `<Track>` element. Each track has a specific ID:

SPRT	Sprite Track
TXTt	Text Track
EFCt	Effect Track
CLRt	Color Track
MOVt	Movie Track
SKNt	Skin Track
TWEN	Tween Track
MODI	Modifier Track

MUSI	Music Track
Strm	Streaming Track
flsh	Flash Track
VRtk	VR Track

QTLISTS

QTLists, or QuickTimeLists, enable you to structure and associate data with each track in your movie, as well as with the movie itself. This 'meta data' information can be used to create tailored presentations based on different objectives or audiences. For example, your movie can access a large repository of visual or audio media and provide unique supporting text content for these elements depending on a user's login or category selection.

QTLists have a similar structure to XML, with elements, sub-elements, and values that can be accessed by, or provided to, the user. Because of this common standard structure, QTLists can communicate directly with a database that understands XML, or through a CGI script or other middle ware to servers that are not XML-aware. The QuickTime movie can send requests to a URL via HTTP and receive back XML data. This enables QTLists to be updated on demand in real-time.

LOADING XML DOCUMENTS

Each track within your QuickTime movie can have its own XML document associated with it.

Before the content of an XML document can be displayed or parsed, it must first be loaded into the track's QTList. To load an XML document, use the following syntax on a scriptable track.

TrackNamed("Track_Name").SetListFromURL("URL", "Element_Path")

The URL is the location of the XML file you want to load. The Element_Path tells QuickTime how to find the information you want in the XML document's hierarchy. This path is structured from the root element through each sub-element to the element you wish to access, separated by a "."

Example:

```
<Cookbook>
          <Section>
                    <Recipe>Recipe Content</Recipe>
          </Section>
<Cookbook>
```

The path to the recipe content would be (Cookbook.Section.Recipe)

If you want to load the entire XML document, leave the Element_Path blank.

Example: TrackNamed("recipes").SetListFromURL("MyRecipes.xml","")

This action will place the XML string at the beginning of the track's QTList. Prior to loading your XML, you will likely want to clear any existing XML data from your track to avoid multiple files appearing in the same space. This line of QScript placed before the one above will accomplish this:

```
TrackNamed("Track_Name").RemoveListElement("element_path",1,TrackNamed("Track_Na
me").GetListElementCount("Element_Path"))
```

This removes the XML data specified by the element path from the first data element to the last. When you do not know how many data elements your XML file will contain, use the "GetListElementCount" property. Again, if you are working with an entire XML document, as opposed to a portion of one, you can leave the "Element_Path" blank.

Example: TrackNamed("recipes").RemoveListElement("",1,TrackNamed("recipes"). GetListElementCount(""))

Displaying XML Data

Once your XML document has been loaded into the track's QTList, you can either display it as raw XML or use QuickTime to parse the data. The following script will place a string containing the raw XML data into a Movie Variable that can then be used to display the data:

```
SetString(mVariable_Name, TrackNamed("Track_Name").
GetListAsXML("Element_Path",1,TrackNamed("Track_Name").
GetListElementCount("Element_Path")))
```

Note: The above script should appear on a single line.

This collects the XML data attached to the particular track and loads it into a variable that can then be used to display the data.

For more information on Variables, see Chapter 16—Introduction to QScript, page 323.

PARSING XML DATA

It is in QuickTime's ability to parse XML data that you can truly take advantage of what XML can accomplish. Instead of a raw XML document, you can access the precise information in the XML file you require, and format it accordingly. Using the GetListElementValue property in conjunction with variables, you can display individual elements from your track's XML document and apply any formatting you like.

To retrieve a single element from an XML document associated with a track, the following syntax is used (all on one line):

```
SetString(mVariable_Name, TrackNamed("Track_Name").
GetListElementValue(Element_Path))
```

Example: SetString(mMyRecipe, TrackNamed("recipes"). GetListElementValue(Cookbook.Section.Recipe))

This script collects the data located in the "Recipe" sub-element, contained in the sub-element "Section" in the root element "Cookbook" in the XML document.

This is only a basic example. To access multiple elements within an XML document, you will need to utilize more advanced scripting. This is covered in detail in StageDoor tutorial #08 (http://stagedoor.totallyhip.com).

Server Query Actions

There are three means of accessing and receiving XML data from an online database, or flat file. The first is SetListFromURL, covered in the "Loading XML Documents" section. SetListFromURL simply loads the XML string from a file or URL into the targeted track's QTList. The SetListFromURL action is known as a "synchronous" action. This means that the query action will be completed before the movie will continue. While the movie is contacting and receiving the XML information from a URL, it will pause.

The second two actions, ExchangeList and QueryListServer, are "asynchronous" actions. The movie will continue as the actions query the server and no pause will occur.

EXCHANGELIST

ExchangeList is used when you want to send a track's QTList to a server and then receive a returned list from the server. When a list is received, a List Received Event is triggered in the track that sent the ExchangeList.

ExchangeList(URL, Element_Path)

Example: TrackNamed("recipes").\\
ExchangeList("http://www.myrecipes.com","")

This sends the contents of the list that is associated with the "recipes" track to the URL. The contents are specified by the Element Path, which in this case, being empty, is the entire QTList.

In order for ExchangeList to operate there must be an XML document associated with the target track.

When a list is returned from the server, it is saved in the temporary list of the List Received Event handler using the following format:

```
<event>
          <listName>
                    Name of the List
          </listName>
          <list>
                    Contents of the returned list.
          </list>
</event>
```

When targeting an element in the returned list, you will need to follow the path to that element taking the above elements into consideration (event.list.element).

While SetListFromURL automatically loads the XML string into the target track, with ExchangeList you will need to load it manually from the List Received temporary list using a QScript command. To do so, use the “ThisEvent” target:

ThisEvent.GetListElementValue(Element_Path)

Then load it into the track using:

TrackNamed(Track_Name).SetListElement(Element_Path, Value)

Example:

```
TrackNamed(“recipes”).\\
SetListElement(“recipe”,ThisEvent.GetListElementValue\\
(“event.list.recipes”))
```

QUERYLISTSERVER

QueryListServer allows you to exert more control over how your movie will communicate with the server. While not mandatory, parameters added to the command allow you to specify key/value pairs to be appended to the URL, and flags offer further control the query action.

QueryListServer(URL,Key_Value_Pairs,Flags,Element_Path)

Example:

```
TrackNamed(“recipes”).QueryListServer\\
(“http://www.recipes.com/database.cgi”,”type=recipe_list”,\\
kListQueryWantCallBack,””)
```

This targets the “recipes” track and sends the QTList associated with it to the URL’s CGI script. The “type=recipe_list” specifies which URL to return, and the kListQueryWantCallBack flag triggers the ListReceived Event when the data is returned from the server.

Flags

Four flags can be used as parameters in the QueryListServer command:

kListQuerySendListAsXML
This sends the targeted track’s QTList as XML appended to the URL.

Example: www.myURL.com/mycgi.cgi?qtlist=<qtlist>XML list content</qtlist>

kListQuerySendListAsKeyValuePairs
This sends the targeted track’s QTList as key/value pairs appended to the URL.

Example: www.myURL.com/mycgi?myKey=myValue

kListQueryWantCallback
This instructs QuickTime to trigger the List Received event when the server returns data.

kListQueryDebugTrace
This sends the complete URL generated to the Debug Console for viewing.

You can use multiple flags in the same QueryListServer command by separating them with a “+”.

Example:
```
TrackNamed(“recipes”).QueryListServer\\
(“http://www.recipes.com/database.cgi”,”type=recipe_list”,\\
kListQueryWantCallBack+kListQueryDebugTrace,””)
```

XML & QTLISTS WRAP UP

While these are advanced features in LiveStage Professional, the capabilities of XML and QTLists can be combined to produce powerful and dynamic Internet applications for delivering content. A good example of QTLists in action can be found at www.toddishere.com. “TODD” accesses an online repository of video, image, and text data and supplies it on-demand through a skinned player. On the StageDoor site, you will find several tutorials that will take you step-by-step through building QTList-enabled projects. You can also find more information on the various QTList commands available in Appendix 1—QScript Reference.

Chapter 19

Setting LiveStage Professional Preferences

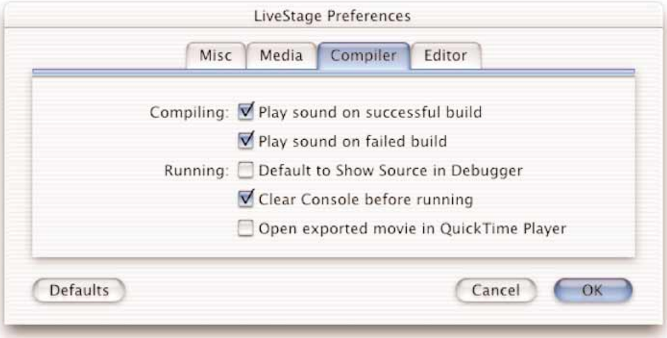
OVERVIEW

LiveStage Professional 4.0 affords a great deal of customization through the setting of Preferences. There are some settings, such as those in the View menu, that will automatically be saved in the Preferences for you. Others must be specified in the Preferences Window. To access the Preferences Window, select Preferences... from the LiveStage Pro Menu (Command-,).

The LiveStage Professional Preferences window is divided into four categories, Misc, Media, Compiler, and Editor. Each tab holds preferences for specific LiveStage Professional functions. At the bottom of the Preferences window is a 'Defaults' button that will reset all the Preferences parameters to their original settings.

MISC TAB

The Misc tab contains general functionality settings in LiveStage Professional.


LiveStage Preferences

Compiling: ☑ Play sound on successful build
☑ Play sound on failed build
Running: □ Default to Show Source in Debugger
☑ Clear Console before running
□ Open exported movie in QuickTime Player

STARTUP ACTION

Using this menu, you can select from a number of options to determine how LiveStage Professional will start up. These options include:

Ask What to Do

On startup, LiveStage Professional will ask if you would like to open an existing project or create a new project. This is the default setting.

Create New Document

On startup, LiveStage Professional will automatically create a new untitled Project.

Open Existing Document

On startup, LiveStage Professional will present a standard file open dialog through which you can select the project you would like to have opened.

Reopen Last Document

On startup, LiveStage Professional automatically opens the last saved LiveStage project.

Do Nothing

On startup, LiveStage Professional will run but will neither prompt to open nor will create a new document.

AUTO CREATE NEW SPRITE PROPERTIES

The Auto Create New Sprite Properties setting determines what will happen when a sprite is moved on the Stage. When this setting is selected and a sprite is moved, a new Sprite Property Chip will be created (if needed) at the current time. You can view this in the Sprite Timeline located in the Sprite Sample Properties window. This can be used to do simple animation by moving a sprite, changing the time, moving the sprite again, and so on. For information on the Sprite Timeline, see Chapter 7—The Sprite Track.

ALLOW MULTIPLE PREVIEW WINDOWS

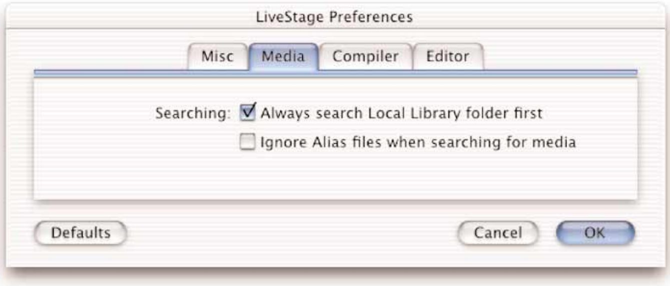
When Allow Multiple Preview Windows is selected, you can have multiple Previews of your movie open at the same time. When this is deselected, any existing Preview windows will be closed and the new preview will be displayed.

AUTO SAVE [NEW TO LIVESTAGE PROFESSIONAL 4.0!]

You can choose to have LiveStage Professional automatically save your project as you work. Select this setting and enter the number of minutes you want between save operations. Once you have initially saved your project, the Auto Save feature will save a backup of your project to a file called myProjectName.bak in the same directory as the project file.

MEDIA TAB

The settings in the Media tab determine how LiveStage Professional will search for your project's media.



Searching: ☑ Always search Local Library folder first

☐ Ignore Alias files when searching for media

ALWAYS SEARCH LOCAL LIBRARY FOLDER FIRST

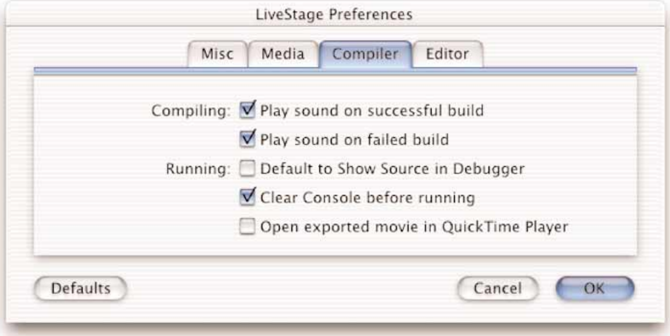
This selection tells LiveStage Professional to look for all your project's media in the Local Library prior to looking elsewhere. Media files in LiveStage Professional projects can be located practically anywhere, including network severs, CD-ROMs etc. This setting is useful when you have copied your media files into the Local Library folder and you do not want LiveStage to use any media files that are not present in the Local (or Global) Library.

IGNORE ALIAS FILES WHEN SEARCHING FOR MEDIA

This setting will ignore a shortcut or alias to a media file or folder. This is useful when you have a project that has been copied from another computer, or if you have aliases to a CD-ROM or network that is not currently available.

COMPILER TAB

The options in the Compiler tab control the compiler and Debugger Console settings.



Compiling: ☑ Play sound on successful build
☑ Play sound on failed build
Running: □ Default to Show Source in Debugger
☑ Clear Console before running
□ Open exported movie in QuickTime Player

COMPILING

Play Sound on Successful Build

This selection enables an audio confirmation that your scripts have compiled without errors.

Play Sound on Failed Build

This selection enables a distinct audio warning that your scripts have not compiled due to errors.

RUNNING

Default To Show Source In Debugger

When this option is checked, LiveStage Professional will display all executed QScripts in the Debugger Console at runtime. You can also control the display of the script source in the Debugger Console through QScript (see Chapter 16—Introduction to QScript. page 329).

Clear Console Before Running

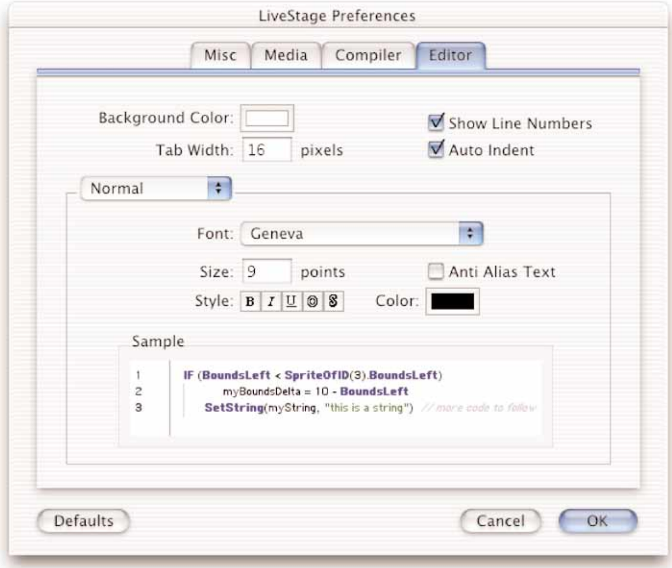
This option clears the Debugger Console of any previous debug data and text prior to running the current movie.

OPEN EXPORTED MOVIE IN QUICKTIME PLAYER

When this option is checked, your movie will open in the QuickTime Player when it has been exported.

EDITOR TAB

The Editor tab allows you to customize the text that appears in the Script Edit windows in all scriptable tracks. Any alterations you make in this tab are previewed in the Sample pane below.



Background Color: □

Tab Width: 16 pixels

☑ Show Line Numbers

☑ Auto Indent

Normal

Font: Geneva

Size: 9 points

□ Anti Alias Text

Style: B I U ◎ S

Color: ■

Sample

```
IF (BoundsLeft < SpriteOfID(3).BoundsLeft)
        myBoundsDelta = 10 - BoundsLeft
    SetString(myString, "this is a string")  // more code to follow
```

BACKGROUND COLOR

You can select a background color for your Script Edit windows by clicking on this swatch and selecting a color from the Color Picker.

TAB WIDTH

This option allows you to specify the size in pixels of a tab in the Script Edit windows.

SHOW LINE NUMBERS

You can elect to display the line numbers along the left margin of the Script Editor.

AUTO INDENT

This setting automatically indents your QScript as you type.

SYNTAX STYLE POPUP MENU

This popup provides the different syntax styles that can be modified through the formatting options in this tab. To alter a style, first select it from the popup and then apply the desired formatting from the Font, Size, Style, Color and Anti Alias options provided.

Normal

Normal is the default text setting for any text that does not fall into the other style categories.

Reserved

Select this setting to modify the text formatting for any keyword that is reserved within the QScript language.

Constants

Select this setting to modify the text formatting for any constant that is used within your QScript.

Comments

Select this setting to modify the text formatting for the comments in your QScript.

Strings

Select this setting to modify the text formatting for strings in your QScript.

Highlighted

Select this setting to modify the text formatting for highlighted lines of QScript.

FONT

The Font menu lists all available fonts that can be used to modify the syntax styles.

Size

Specify the point size to use.

Anti Alias Text

When selected, all the text in the Script Edit window is anti-aliased (Note: OSX text is already anti-aliased, so no difference will be apparent).

Style

Use any combination of Bold, Italics, Underline, Outline, and Shadow to format your text.

Color

Click the swatch and the select the desired text color from the Color Picker.

Appendix 1

QScript Reference

INTRODUCTION

Welcome to the LiveStage Professional QScript Reference Guide. This guide is designed to help you and provide detailed information on the various QScript statements utilized by LiveStage Professional. It includes correct Syntax, Descriptions, and Examples of each statement to assist you in understanding what they do and how to use them.

Conventions Used

`language element` Text shown in this font must be entered exactly as shown.

`placeholder` Text shown in italics indicates a place holder that you must replace with an appropriate value.

[optional] Brackets indicate that the enclosed language elements are optional. However, brackets in bold are part of the syntax and must be entered as shown.

A | B | C A vertical bar separates a group of elements from which one must be chosen.

<> Enclose a group of multiple choice elements.

For Example `SetImageIndexTo(`*`index`*`)` **`[min(`***`number`***`) max(`***`number`***`)Wraparound]`**

You can replace "*`index`*" with any numeric expression you want, such as

```
SetImageIndexTo(1)
```

Here are some examples:

```
// causes the image index to increase by one and when
// it passes 10 it starts over at 1
SetImageIndexBy(1) MIN(1) MAX(10) Wraparound
// causes the image index to be set to the value of x
// and limited to a number from 8 to 10
SetImageIndexTo(x) MIN(8) MAX(10)
```

QSCRIPT SYNTAX

This section of the QScript Reference contains descriptions of the various types of QScript syntactical constructs.

Data Specifiers

Data is the life blood of scripting languages. QScript is no exception. The following section describes the syntax for specifying data within QScript.

Constants

Description Constants, as the name implies, are data that can not be altered when the script is executed. Constants are also called “Literals”. The following are some examples of constants in QScript:

Example

`1234`	Numeric constant indicating the whole number 1234
`1234.56`	Numeric constant indicating the real number 1234.56
`-1`	Negative numeric constant indicating the whole number -1
`“Hi”`	Text or String constant containing the characters “Hi”
`TRUE`	Predefined boolean constant for TRUE
`FALSE`	Predefined boolean constant for FALSE
`$”Image One”`	The index of the image named “Image One”

Constants can be used any place in a script where a number or string is required.

See also Numeric Expression, String Literal

Numeric Expression

Description A Numeric Expression is an expression that, when evaluated, generates a numeric result. Numeric Expressions are made up of numeric elements (i.e. numeric constants, variables and properties) joined with numeric operators (i.e. +, -, *, /, etc.).

Numeric Expressions can be used any place in a script where a numeric value is required.

Example

```
1 + 2 * 3
ThisSprite.ImageIndex + 1
ABS -1
```

See also Constants, Operators, Properties, Variables

String Literal

Description A String Literal is made up of text characters enclosed within quotation marks (“). String Literals are used to specify textual information any place within scripts where a textual value is required. (QT Version 4.0 or later)

Example

```
“Hello World!”
“http://www.totallyhip.com”
```

See also Constants

Boolean Literal

Description A Boolean Literal can be one of two values, TRUE or FALSE. Boolean Literals can be used any place in a script where a boolean value is required.

Example

```
SetVisible(TRUE)
```

See also Constants

Defines

Description A define is an abstract representation of a constant. Defines allows you to assign a name to a number or string of characters. Please refer to the section on the “Defines Tab” in Chapter 3 of the LiveStage Professional User Manual for information on how to create your own custom defines.

By creating and using custom defines in your scripts, you can centralize the location for your custom constant values.

Example

In the Defines Tab:

```
kHomeURL = “http://www.totallyhip.com”
```

In the script:

```
GotoURL($kHomeURL)
```

The example above illustrates how defines are created and used. In the first part of the example, a define is added to the project in the Defines Tab to refer to a URL using the name “kHomeURL”.

In the second part of the example, the custom define “kHomeURL” is used in a QScript Action that requires a string parameter.

The “kHomeURL” custom define can be used throughout the project. If you wish to change the value of the “kHomeURL” custom define, you simply change it in the Defines Tab. Once you make the change in the Defines Tab, every instance where “kHomeURL” is used in your script will now use the new value.

To refer to a define, put a “$” in front of its name. If the name of the define has spaces in it, then the name must be enclosed in quotes.

Example

```
SpriteOfID($ThisSpriteID).SetVisible(TRUE)
SetImageIndexTo($”Up Arrow Image.qif”)
ExecuteEvent($”Reset Game”)
```

There are several defines created for you automatically. These are:
ThisSpriteID: Set to the ID of the sprite containing the script.
Images: Each image has a define created that represents the image index.
Custom Handlers: Each custom event handler has a define created that represents the Handler ID for that handler.

BUILT-IN DEFINES

There are also built-in defines for various numeric constants used by QScript and QuickTime. You do not need to add a “$” in front of Built-in Defines.

The following is a list of the QScript Built-in Defines. For more details on each of these built-in defines, please refer to the QScript Actions where they are used elsewhere in the QScript Reference.

CURSOR IDS

kQTCursorOpenHand
kQTCursorClosedHand
kQTCursorPointingHand
kQTCursorRightArrow
kQTCursorLeftArrow
kQTCursorDownArrow
kQTCursorUpArrow
kQTCursorArrow

DRAWING MODES DEFINES

srcCopy
srcOr
srcXor
srcBic
notSrcCopy
notSrcOr
notSrcXor
notSrcBic
blend
addPin
addOver
subPin
transparent
addMax
subOver
adMin
grayishTextOr
hilite
ditherCopy
graphicsModeStraightAlpha
graphicsModePreWhiteAlpha
graphicsModePreBlackAlpha
graphicsModeComposition
graphicsModeStraightAlphaBlend
graphicsModePreMulColorAlpha

EVENT IDS

kQTEventFrameLoaded
kQTEventIdle
kQTEventKey
kQTEventMouseClick
kQTEventMouseClickEnd
kQTEventMouseClickEndTriggerButton
kQTEventMouseEnter
kQTEventMouseExit
kQTEventMouseMoved
kQTEventListReceived
kQTEventMovieLoaded
kQTEventRequestToModifyMovie
Flash Mouse Transitions
kIdleToOverUp
kOverUpToIdle
kOverUpToOverDown
kOverDownToOverUp
kOverDownToOutDown
kOutDownToOverDown
kOutDownToIdle
kIdleToOverDown
kOverDownToIdle
Key Code Defines
kReturnKeyCode
kEnterKeyCode
kTabKeyCode
kBackSpaceKeyCode
kDeleteKeyCode
kInsertKeyCode
kUpArrowKeyCode
kDownArrowKeyCode
kLeftArrowKeyCode
kRightArrowKeyCode
kEscapeKeyCode
kPageUpKeyCode
kPageDownKeyCode
kHomeKeyCode
kEndKeyCode

KEY STRING DEFINES (USE KEYISDOWN OR STRING COMPARISONS ONLY)

kReturnKey
kTabKey
kDeleteKey
kUpArrowKey
kDownArrowKey
kLeftArrowKey
kRightArrowKey
kDoubleQuoteCharacter
kLineFeedCharacter
kCRLFCharacters
kCRCharacter
kLFCharacter

KEY MODIFIER DEFINES

kModifierActiveFlag
kModifierButtonState
kModifierCommandKey
kModifierShiftKey
kModifierAlphaLock

kModifierOptionKey
kModifierControlKey
kModifierRightShiftKey
kModifierRightOptionKey
kModifierRightControlKey

KEY MODIFIER FLAGS (USED ONLY WITH KEYISDOWN)

kNone
kOptionKey
kShiftKey
kCommandKey
kControlKey
kCapsLockKey

MOVIE LOOPING CONTROL DEFINES

kNoLoop
kLoop
kLoopPalindrome

MEDIA TYPE DEFINES

kBaseMediaType
kFlashMediaType
kMPEGMediaType
kMusicMediaType
kQTVRQTVRType
kQTVRObjectType
kQTVROldPanoType
kQTVROldObjectType
kQTVRPanoramaType
kSpriteMediaType
kStreamingMediaType
kSkinMediaType
kSoundMediaType
kTextMediaType
kThreeDeeMediaType
kTimeCodeMediaType
kTweenMediaType
kVideoMediaType

MUSIC TRACK DEFINES

kControllerModulationWheel
kControllerBreath
kControllerFoot
kControllerPortamentoTime
kControllerVolume
kControllerBalance
kControllerPan
kControllerExpression
kControllerLever1
kControllerLever2
kControllerLever3
kControllerLever4
kControllerLever5
kControllerLever6
kControllerLever7
kControllerLever8
kControllerPitchBend
kControllerAfterTouch
kControllerPartTranspose
kControllerTuneTranspose
kControllerPartVolume
kControllerTuneVolume
kControllerSustain
kControllerPortamento
kControllerSostenuto
kControllerSoftPedal
kControllerReverb
kControllerTremolo
kControllerChorus
kControllerCeleste
kControllerPhaser
kControllerEditPart
kControllerMasterTune
kControllerMasterTranspose
kControllerMasterVolume
kControllerMasterCPULoad
kControllerMasterPolyphony
kControllerMasterFeatures

NUMERIC DEFINES

EPSILON_FLOAT
MIN_FLOAT
MAX_FLOAT
MIN_LONG
MAX_LONG
MIN_SHORT
MAX_SHORT
MAX_USHORT
PI or π

QUERYLISTSERVER DEFINES (QT5 OR LATER)

kListQueryDebugTrace
kListQuerySendListAsKeyValuePairs
kListQuerySendListAsXML
kListQueryWantCallback

SENDAPPMESSAGE DEFINES (QT5 OR LATER)

```
kAppMsgDisplayChannels
kAppMsgRequestEnterFullScreen
kAppMsgRequestExitFullScreen
kAppMsgSoftwareChanged
kAppMsgWindowClose
```

STATE DEFINES

MOVIE LOAD STATES (QT5 OR LATER)

```
kLoadStateError
kLoading
kLoaded
kPlayable
kPlaythroughOK
kComplete
```

NETWORK STATES (QT5 OR LATER)

```
kConnected
kNoNetwork
kNotConnected
kUncertain
```

TEXT TRACK DEFINES

JUSTIFICATION (QT5 OR LATER)

```
kCenterJustify
kLeftJustify
kRightJustify
```

TEXT DISPLAY FLAGS (QT5 OR LATER)

```
kDontDisplay
kDontAutoScale
kClipToTextBox
kUseMovieBGColor
kShrinkTextBoxToFIt
kScrollIn
kScrollOut
kHorizScroll
kReverseScroll
kContinuousScroll
kFlowHoriz
kContinuousKaraoke
kDropShadow
kAntiAlias
kKeyedText
kInverseHilite
kTextColorHilite
```

TEXT EDIT STATES (QT5 OR LATER)

```
kDirectEditing
kNoEditing
kScriptEditing
```

TEXT STYLES (QT5 OR LATER)

```
kNormalFace
kBoldFace
kItalicFace
kUnderlineFace
kOutlineFace
kShadowFace
kCondenseFace
kExtendedFace
```

TEXT FONTS (QT5 OR LATER)

```
kAthensFont
kCairoFont
kCourierFont
kGenevaFont
kHelveticaFont
kLondonFont
kLosAngelesFont
kMobileFont
kMonacoFont
kNewYorkFont
kSanFranciscoFont
kSymbolFont
kTimesFont
kTorontoFont
kVeniceFont
```

TEXT SEARCH DEFINES (QT5 OR LATER)

```
kSearchAgain
kSearchCaseSensitive
kSearchCurrentSample
kSearchReverse
kSearchWraparound
```

See also Constants

Preprocessor Directives

Preprocessor Directives are added to scripts to control the compilation of a project.

#debug

Syntax `#debug on | off`

on | off Indicates if debug output is to be turned on or off

Description This directive turns on or off the output of QScript source code to the Debugging Console.

Use this compiler directive to limit the amount of source code that is sent to the Debugging Console by selectively turning this directive on and off.

Example Use the following syntax to restrict the lines of script that are sent to the Debugging Console:

```
#debug on

LocalVars myNum
myNum = 100
        ...

#debug on
// Compute the value of myNum
        ...
#debug off
        ...
```

The above script sends only the script lines enclosed between the “#debug on” and “#debug off” statements to the Debugging Console.

#define

Description This directive creates a Value Definition that associates a name (*defineName*) with a value (*defineValue*). Once defined, a Value Definition can be use anywhere within the scope of the current script.

Value Definitions are designed to work with the `#include` directive. Commonly used Value Definitions can be placed in a script file external to the LiveStage Professional Project and be included in the compilation process by the use of the `#include` directive. This allows you to develop script language segments that can be used in many projects.

To use a Value Definition in a QScript statement, add a “$” to the front of the Value Definitions.

Syntax `#define` *defineName* = *defineValue*

defineName the name of the value definition

defineValue the value of the value definition

```
#define kStartTime = 0
ThisMovie.GoToTime($kStartTime)
```

Example Create Value Definitions in a file with the name of “defines.qsf”:

```
#define kMyURL = “http://www.totallyhip.com”
        ...
```

Add the following line to the beginning of any script where the Value Definitions in the file “defines.qsf” is to be used:

```
#include “defines.qsf”
GotoURL($kMyURL)
```

See also #include

#include

Syntax `#include “filename”`

Description This directive specifies a file containing additional QScript statements that are to be included at compilation time. This allows code segments to be placed in external files (external to the LiveStage Professional project) and shared between multiple projects.

The file specified must be saved in the Library folders (Global or Local) as this is the only place where these include files can be located.

Example

```
#include “mycode.qsf”
```

See also #define

#RegionFromImageFile

Syntax `#RegionFromImageFile(filename)`

filename A string literal specifying the name of the image file

Description This directive is used to specify a region defined using an image file. A black and white image file is supplied to provide the shape of the clipping region. A black and white image file must be used or the region created would be invalid.

The file specified must be saved in the Library folders (Global or Local) as this is the only place where these files can be located.

Example

```
ThisTrack.SetClipRegionTo(
#RegionFromImageFile(0,”myfile.pct”)
// Sets the clipping region to the
// region specified by the image file
```

See also #RegionFromRect

#RegionFromRect

Syntax `#RegionFromRect(left, top, right, bottom)`

left, top Numeric literals specifying the coordinates for the top left corner of the rectangle

right, bottom Numeric literals specifying the coordinates for the bottom right corner of the rectangle

Description This directive is used to specify a rectangular region. The parameters specify the location of the top left and bottom right corners of the region.

Example

```
ThisTrack.SetClipRegionTo(
         #RegionFromRect(0, 0, 100, 100))
// Sets the clipping region to the rectangle
// Specified
```

See also #RegionFromImageFile

#StringFromFile

Syntax `#StringFromFile(`*`filename`*`)`

`filename` A string literal specifying the name of the file

Description This directive specifies that a text file (specified) contains the text used in a string literal. The preprocessor directive is replaced with the contents of the specified file when the script is compiled.

The file specified must be saved in the Library folders (Global or Local) as this is the only place where these files can be located.

Example

```
ReplaceText(#StringFromFile(“mytext.txt”),
1, GetTextLength)
// Replace the text in the current text sample
// with the characters in “mytext.txt”
```

See also #include

Targets

Most actions and properties require a target that specifies what is to execute the action or what is to return a property. If no target is specified, or a partial target is specified, then the default targets are used which are the current movie, track and sprite that is executing the script. Targets consist of three portions, the movie, the track within the movie and the sprite within the track. These can be completely or partially specified. You can specify just the sprite, or just the sprite and the track or just the track or just the movie. Whatever part of the target you do not specify will be replaced with the default target.

Event Target

Syntax `ThisEvent`

Description The `ThisEvent` target is used to access the temporary QTList accessible during the execution of an event handler. The temporary QTList attached to an event contains information specific to the event handler currently being executed.

The following are the layout of the event handler local QTLists for the various built-in events:

Key Pressed:

```
<event>
         <key>key value</key>
         <modifiers>key modifier flags</modifiers>
         <scancode>key scan code</scancode>
</event>
```

List Received:

```
<event>
         <listName>Name of the list</listName>
         <list>
                  Contents of the returned list
                           ...
         </list>
</event>
```

Mouse Events (Click, Down, Enter, Exit, Moved, Up):

```
<event>
         <where>
                  <h>X Position</h>
                  <v>Y Position</v>
         </where>
</event>
```

All other events have empty temporary QTLists.

Example

```
LocalVar theValue
SetString(theValue, ThisEvent.GetListElementValue("", 1))
// return the value of the first element in the
// event handler's temporary list
```

QT Version 5.0 or later

See also List Properties and Actions

Object Target

Syntax

```
ObjectNamed(name)
ObjectOfID(id)
ObjectOfIndex(index)
```

name A string literal specifying the name of the object.

id A numeric literal specifying the ID of the object.

index A numeric literal specifying the index of the object.

Description Objects within various Track (i.e. Sprite, QT3D, Flash, etc.) can be targeted by using its name, ID or index and corresponding Object target specifier.

QT Version 5.0 or later

Movie Target

Syntax

```
MovieOfID(id)
MovieNamed(name)
ThisMovie
```

id A numeric literal specifying the ID of the movie.

name A string literal specifying the name of the movie.

Description A movie can be targeted by using its ID or name to identify it. You can set the name or ID of the movie in the Info tab of the document window. In an html document you can also specify the name or ID of the move as part of the embed tag. You can do this using the free “Plug-in Helper” application from the following URL:

```
            http://www.apple.com/quicktime/developers/tools.html
```

or you can enter it yourself as follows:

```
       <embed src=”mymovie.mov” moviename=”name” ...>
```

Example

```
MovieOfID (1)
MovieNamed (“My Movie”)
```

QT Version 4.0 or later

Movie Target (Parent Movies)

Syntax

```
RootMovie
ParentMovie
```

Description A child movie can use these to target the root movie, which is the movie that contains all other child movies. The parent movie is the movie that contains this child movie. Use these only from within a child movie.

Example

```
RootMovie.StartPlaying
ParentMovie.StartPlaying
```

QT Version 4.1 or later

See also Movie Target (Child Movies)

Movie Target (Child Movies)

Syntax

```
ChildMovieTrackNamed(name)
ChildMovieTrackOfID(id)
ChildMovieTrackOfIndex(index)
ChildMovieNamed(movie_name)
ChildMovieOfID(movie_id)
```

`name`	A string literal specifying the name of the movie track.
`id`	A numeric literal specifying the ID of the movie track.
`index`	A numeric literal specifying the index of the movie track.
`movie_name`	A string literal specifying the name of the child movie.
`movie_id`	A string literal specifying the ID of the child movie.

Description These all target a child movie in some way. They do not target a track even though they mention tracks in their names. The movie that is targeted will be the currently loaded movie in the child movie track. If a movie is added to the child movie list of a track, but it is not loaded, then it can not be targeted because it does not exist yet. The targets that mention a child movie track all target the currently loaded movie in the child movie track specified. The targets that specify the movie name and movie ID will target a child movie that has the matching movie name or movie ID. This name or ID can be set for a movie in the info tab when you are creating the movie.

Example

```
ChildMovieTrackNamed(“My Track”).StartPlaying
ChildMovieNamed(“My First Movie”).StartPlaying
```

QT Version 4.1 or later

See also Movie Target (Parent Movies)

QD3DObject Target

Syntax
```
QD3DObjectNamed(name)
```

name A string literal specifying the name of the QD3D Object.

Description A QD3D Objects can be targeted by using its name.

QT Version 3.0 or later

Sample Target

Syntax
```
SpriteNamed(name)
```

name A string literal specifying the name of the sample.

Description A media sample can be targeted by using its name. You can set the name of a media sample in the Sample Editor for each of the editable track types.

Example
```
LocalVars theTime
theTime = SampleNamed(“MySample”).StartTime
```

QT Version 4.0 or later

Sprite Target

Syntax
```
SpriteOfID(id)
SpriteOfIndex(index)
SpriteNamed(name)
ThisSprite
```

id A numeric literal specifying the ID of the sprite.

index A numeric literal specifying the index of the sprite.

name A string literal specifying the name of the sprite.

Description A sprite can be targeted by using its ID, index or name to identify it. You can set the name and ID of the sprite from the Sprite Editor within LiveStage Professional. The index is set by LiveStage Professional.

Example
```
SpriteOfID(1).SetVisible(FALSE)
SpriteNamed(“MySprite”).SetVisible(TRUE)
```

QT Version 3.0 or later

Track Target

Syntax
```
TrackOfIndex(index)
TrackNamed(name)
TrackOfID(id)
TrackOfType(type [, index] )
ThisTrack
```

index A numeric literal specifying the index of the track.

name A string literal specifying the name of the track.

id A numeric literal specifying the ID of the track.

type A numeric constant specifying the type of the track.

Description A track can be targeted by using its index, ID, name or type to identify it. The index and name of a track is shown in the Tracks tab of the document window. The ID of the track is set by QuickTime when the movie is created and is not known until then, but it is usually the same as its index.

Example

```
TrackOfIndex(1)
TrackNamed(“Picture Track 1”)
TrackOfID(1)
TrackOfType(kVideoMediaType)
TrackOfType(kTextMediaType, 2)
```

QT Version 3.0 or later

FUNCTIONS

Functions perform tasks that are independent of other objects in a LiveStage Professional project. A function process values supplied via the parameter list and return a value. The following example demonstrates the use of Functions:

```
LocalVars theSquareRoot
theSquareRoot = Sqr(4)
```

In the above example, “Sqr(4)” calculates the square root of the parameter “4”. The result is assigned to the local variable “*theSquareRoot*”.

Functions are similar to Properties and can be used within expressions in place of numeric and string literals.

Math Functions

ArcTan

Syntax `ArcTan(`*`value`*`)`

value A numeric literal, variable or expression specifying the input value.

Description This function calculates the Trigonometric ArcTan of the supplied value.

Return Returns a numeric value in radians.

Example

```
LocalVars theValue
theValue = ArcTan(0.5)
```

QT Version 5.0 or later

See also ArcTan2, Cos, Sin, Tan

ArcTan2

Syntax `ArcTan2(y, x)`

y A numeric literal, variable or expression specifying the y component or a rectangular vector.

x A numeric literal, variable or expression specifying the x component or a rectangular vector.

Description This function calculates the Trigonometric ArcTan2 of the supplied rectangular vector. A rectangular vector is supplied (y, x) which is used to return a full-range (-2 PI—2 PI) angle value.

Return Returns a numeric value indicating the angle in radians calculated by the ArcTan2 function.

Example

```
LocalVars theValue
theValue = ArcTan2(10, 10)
```

QT Version 5.0 or later

See also ArcTan, Cos, Sin, Tan

Cos

Syntax `Cos(angle)`

angle A numeric literal, variable or expression specifying the input angle in radians.

Description This function calculates the Trigonometric Cosine of the supplied angle.

Return Returns a numeric value.

Example

```
LocalVars theValue
// Calculate the Cosine of 90 degrees
theValue = Cos(0.5)
```

QT Version 5.0 or later

See also ArcTan, Sin, Tan

DegreesToRadians

Syntax `DegreesToRadians(degrees)`

degrees A numeric literal, variable or expression specifying the input value in degrees.

Description This function converts the specified value in degrees to the equivalent value in radians.

Return Returns a numeric value.

Example

```
LocalVars theValue
// Convert 360 degrees to radians
theValue = DegreesToRadians(2.0 * PI)
```

QT Version 5.0 or later

See also RadiansToDegrees

Exp

Syntax Exp(*value*)

value A numeric literal, variable or expression specifying the input value.

Description This function calculates the Exponential of the specified value.

Return Returns a numeric value.

Example

```
LocalVars theExponent
// Calculates the exponential
theValue = Exp(100.0)
```

QT Version 5.0 or later

See also Log, Sqr

Log

Syntax Log(*value*)

value A numeric literal, variable or expression specifying the input value.

Description This function calculates the Natural Logarithm of the specified value.

Return Returns a numeric value.

Example

```
LocalVars theLog
// Calculates the Log
theValue = Log(100.0)
```

QT Version 5.0 or later

See also Exp, Sqr

RadiansToDegrees

Syntax RadiansToDegrees(*radians*)

radians A numeric literal, variable or expression specifying the input value in radians.

Description This function converts the specified value in radians to the equivalent value in degrees.

Return Returns a numeric value.

Example

```
LocalVars theValue
// Convert 32 PI radians to degrees
theValue = RadiansToDegrees(32.0)
```

QT Version 5.0 or later

See also DegreesToRadians

Random

Syntax Random(*minimum, maximum*)

minimum A numeric literal or variable specifying the lowest number that can be returned.

maximum A numeric literal or variable specifying the highest number that can be returned.

Description This function generates and returns a random numeric value. The numeric value generated ranges from the minimum to the maximum values specified as parameters.

The Random function was reclassified a Math Function instead of a General Property for LiveStage Professional 3.0.

Return Numeric value ranging from the specified minimum and maximum values.

Example

```
Random(1, 100)
// Generate a random number from 1 to 100
If (Random(1, 100) < 50)
// Script that will execute 50% of the time
```

QT Version 3.0 or later

Sin

Syntax `Sin(angle)`

angle A numeric literal, variable or expression specifying the input angle in radians.

Description This function calculates the Trigonometric Sine of the supplied angle.

Return Returns a numeric value.

Example

```
LocalVars theValue
// Calculate the Sine of 180 degrees
theValue = Sin(1.0)
```

QT Version 5.0 or later

See also ArcTan, Cos, Tan

Sqr

Syntax `Sqr(value)`

value A numeric literal, variable or expression specifying the input value.

Description This function calculates the Square Root of the specified value.

Return Returns a numeric value.

Example

```
LocalVars theSquareRoot
// Calculates the square root
theValue = Sqr(16)
```

QT Version 5.0 or later

See also Exp, Log

Tan

Syntax `Tan(angle)`

angle A numeric literal, variable or expression specifying the input angle in radians.

Description This function calculates the Trigonometric Tangent of the supplied angle.

Return Returns a numeric value.

Example

```
LocalVars theValue
// Calculate the Tangent of 45 degrees
theValue = Tan(0.25)
```

QT Version 5.0 or later

See also ArcTan, Cos, Sin

String Functions

StrLength

Syntax `StrLength(`*`string`*`)`

`string` A string literal or variable specifying the input string.

Description This function returns the length of the specified string.

Return Returns a numeric value.

Example

```
LocalVars TheLength
// Calculates length of the string
heLength = StrLength("This is a test")
```

QT Version 5.0 or later

See also StrCompare, StrConcat, SubString

StrCompare

Syntax `StrCompare(`*`string1, string2, caseSensitive, diacSensitive`*`)`

`string1` A string literal or variable specifying the first input string.

`string2` A string literal or variable specifying the second input string.

`caseSensitive` A boolean literal or variable specifying if the strings are to be compared in a case sensitive manner.

`diacSensitive` A boolean literal or variable specifying if special handling for diacritical marks and special characters is to be used.

Description This function compares the two input strings and returns TRUE if the two strings are the same. Set *`caseSensitive`* to TRUE if you want to find an exact match. Set *`diacSensitive`* to TRUE if you are working with strings containing diacritical marks and special characters.

Return Returns a boolean value.

Example

```
LocalVars theSame
// Compares the two supplied strings
theSame = StrCompare("String1", "STRING1", TRUE, FALSE)
// theSame contains FALSE as the two strings are
// not the same using case sensitive comparison
```

QT Version 5.0 or later

See also StrConcat, StrLength, SubString

StrConcat

Syntax `StrConcat(`*`string1, string2`*`)`

`string1` A string literal or variable specifying the first input string.

`string2` A string literal or variable specifying the second input string.

Description This function returns the concatenation of input strings.

Return Returns a string value.

Example

```
LocalVars theString
// Concatenate two strings
SetString(theString, StrConcat("Part 1 ", "Part 2"))
// theString now contains the string
// "Part 1 Part 2"
```

QT Version 5.0 or later

See also StrCompare, StrLength, SubString

SubString

Syntax `SubString(`*`string, offset, length`*`)`

`string` A string literal or variable specifying the input string.

`offset` A numeric literal or variable specifying the offset from the beginning of the string where the substring starts.

`length` A numeric literal or variable specifying the length of the substring.

Description This function returns a portion of the input string. The portion returned is controlled by the parameters *`offset`* and *`length`*.

Return returns a string containing the specified substring.

Example

```
LocalVars theSub
// Get the first 2 characters of the string
SetString(theSub, SubString("Hi there?", 0, 2))
// theSub contains "Hi"
```

QT Version 5.0 or later

See also StrCompare, StrConcat, StrLength

PROPERTIES AND ACTIONS

When you script in LiveStage Professional, you are working with QuickTime Objects such as Movies, Tracks, Samples, etc.

The way to access these QuickTime Objects is via Properties and Actions.

Properties

A property is a characteristic of an object. A car has a color, engine size, top speed. These are all properties of the car. Most properties of objects in QuickTime movies can only be changed by executing actions. They cannot be changed by assigning values directly to the properties.

Here is the syntax for accessing a Property of a QuickTime Object in QScript:

```
target.property [ (parameters...) ]
```

target This is the specification indicating which QuickTime Object the property is attached to.

. A "." separates the two parts of the property access syntax. This is the standard syntax for object property access for contemporary scripting languages.

property This is the name of the property that you want to access.

[(parameters...)] Some properties accept parameters. These parameters must be placed between parentheses at the end of the property access statement.

Here are some examples of property access statements:

```
ThisMovie.MovieTime    // Accesses the current movie time
SpriteOfIndex(1).ID    // Accesses the sprite ID of the first sprite
```

You can use a property in any place that you would normally use a number. For example:

```
LocalVars theMovieTime
// Save the current movie time in a local variable
theMovieTime = ThisMovie.MovieTime
// Goto the latest downloaded time in the movie
ThisMovie.GotoTime(ThisMovie.MaxLoadedTimeInMovie)
```

You can also use them in expressions. For example:

```
If (ThisMovie.MovieTime < ThisMovie.MaxLoadedTimeInMovie)
// Go to the latest downloaded time in the movie
// if we are not there already
EndIf
```

Actions

Action are commands issued to QuickTime Objects to direct them to perform a task. Some Actions accept information in the form of parameters. Actions do not return a result so they can not be used where you would normally use a number or a property.

Here is the syntax for accessing an Action of a QuickTime Object in QScript:
`target.action [ (parameters...) ] [ min(number) max(number) wraparound ]`

`target` This is the specification indicating which QuickTime Object the Action is attached to.

`.` A "." separates the two parts of the action access syntax. This is the standard syntax for object action access for contemporary scripting languages.

`action` This is the name of the action that you want to perform.

`[ (parameters...) ]` Some actions accept parameters. These parameters must be placed between parentheses at the end of the property access statement.

`[ min(number) max(number) wraparound ]`

Some Actions can have their parameters limited to a range of values. This allows you to have a variable contain the value and not have to check it to see if it is in range. It also allows you to limit the range of relative actions (those that end in "by") so that the resultant setting falls within the range. To do this you need to add the optional `min` and `max` keywords after the action. The ranges should be enclosed in () and must be numeric literals or constants and cannot be variables or expressions. You can also add the `wraparound` keyword to any relative action so that when the resultant setting passes outside of its range it will then enter the range at the opposite end. Actions that can use any of these will show them in the syntax section of their descriptions.

GENERAL PROPERTIES AND ACTIONS

The general Properties and Actions do not require that a target be specified. These Properties and Actions access the properties and actions of the environment that is playing the movie.

General Properties

ComponentVersion

Syntax `ComponentVersion(type, subtype, manufacturer)`

type A string literal identifying the type of component.

subtype A string literal identifying the sub-type of the component.

*manufacturer*A string literal identifying the manufacturer of the component.

Description This property returns the installed version number for the component of the given type, subtype and manufacturer. Each of the three parameters is a four character string literal.

Return Returns a numeric value indicating the installed version of the component. Returns 0 if no component is installed.

Example `If (ComponentVersion(“fire”, “”, “Appl”) > 2)`

QT Version 4.0 or later

See also LoadComponent

ConnectionSpeed

Syntax `ConnectionSpeed`

Description This property returns the speed of the current internet connection in bits per second. A 14.4 modem speed would be returned as 1400 for instance.

Return A numeric value indicating the speed of the current internet connection in bits per second.

The following lists the connection speed reported for various connection settings:

SETTING	VALUE
14.4 Kbps	1400
28.8/33.6 Kbps	2800
56 Kbps Modem/ ISDN	5600
112 Kbps ISDN/DSL	11200
256 Kbps ISDN/DSL	25600
384 Kbps ISDN/DSL	38400
512 Kbps ISDN/DSL	51200
768 Kbps IDSN/DSL	76800
1 Mbps Cable	100000
1.5 Mbps T1	150000
Internet/LAN	2147483648

Example `If (ConnectionSpeed < 28800)`

QT Version 4.0 or later

CustomHandlerID

Syntax `CustomHandlerID`

Description This property returns a unique custom action handler ID.

Return A numeric value representing a unique custom action handler ID.

QT Version 5.0 or later

See also IsCustomHandlerOpen

IsCustomHandlerOpen

Syntax `IsCustomHandlerOpen(`*`handler_ID`*`)`

`handler_ID` A numeric value identifying the custom handler.

Description This property returns TRUE if the specified action handler is open otherwise it returns FALSE.

Return Returns a boolean value of TRUE if the handler is open otherwise it returns FALSE.

QT Version 5.0 or later

See also CustomHandlerID

GetEventKey

Syntax

```
GetEventKey
```

Description This property returns the key code of the key that triggered the Key Pressed Event.

This property should only be used within the KeyPressed Event handler otherwise the returned value may be invalid.

KEY CODE DEFINES

```
kReturnKeyCode
kEnterKeyCode
kTabKeyCode
kBackSpaceKeyCode
kDeleteKeyCode
kInsertKeyCode
kUpArrowKeyCode
kDownArrowKeyCode
kLeftArrowKeyCode
kRightArrowKeyCode
kEscapeKeyCode
kPageUpKeyCode
kPageDownKeyCode
kHomeKeyCode
kEndKeyCode
```

For key codes not defined in the above list, simply refer to the key’s ASCII value.

Do not use the Key String Defines with the GetEventKey property as these constants define strings instead of key codes.

Return A numeric value indicating the key code when the KeyPressed event was triggered.

Example

```
LocalVars keyCode
keyCode = GetEventKey
```

QT Version 5.0 or later

See also GetEventMouseX, GetEventMouseY, GetEventModifiers, GetEventScanCode

GetEventModifiers

Syntax

```
GetEventModifiers
```

Description This property returns the key modifiers at the time the KeyPressed Event was triggered.

The following Modifier Key defines are available:

MODIFIER DEFINE	MEANING
`kModifierCommandKey`	Command key held
`kModifierShiftKey`	Shift key held
`kModifierAlphaLock`	CapsLock key held
`kModifierOptionKey`	Option key held
`kModifierControlKey`	Control key held
`kModifierRightShiftKey`	Right Shift key held
`kModifierRightOptionKey`	Right Option key held
`kModifierRightControlKey`	Right Control key held

The above modifier defines can be used together to specify that two or more modifier keys are pressed at the same time (i.e. kModifierCommandKey + kModifierShiftKey)

This property should only be used within Mouse or Keyboard Event handler otherwise the returned value may be invalid.

Return A numeric value indicating the key modifier state when the Mouse or Keyboard event was triggered.

Example

```
If (GetEventModifiers = (kModifierCommandKey +
    kModifierShiftKey))
// Script executes only if the Command and Shift
// keys are pressed
Endif
```

QT Version 5.0 or later

See also GetEventMouseY, GetEventModifiers, GetEventKey, GetEventScanCode

GetEventMouseX

Syntax `GetEventMouseX`

Description This property returns the X coordinate of the mouse when the event was triggered. The coordinate returned is in the coordinate space of the track.

Return A numeric value indicating the X position of the mouse.

Example

```
LocalVars origMouseX
origMouseX = GetEventMouseX
```

QT Version 5.0 or later

See also GetEventMouseY, GetEventModifiers, GetEventKey, GetEventScanCode, GetEventMouseY

GetEvenMouseY

Syntax `GetEventMouseY`

Description This property returns the Y coordinate of the mouse when the event was triggered. The coordinate returned is in the coordinate space of the track.

Return A numeric value indicating the Y position of the mouse.

Example

```
LocalVars origMouseY
origMouseY = GetEventMouseY
```

QT Version 5.0 or later

See also GetEventMouseX, GetEventModifiers, GetEventKey, GetEventScanCode

GetEventScanCode

Syntax

```
GetEventScanCode
```

Description This property returns the keyboard scan code of the key that triggered the KeyPressed Event.

This property should only be used within the KeyPressed Event handler otherwise the returned value may be invalid.

Return A numeric value indicating the key scan code when the KeyPressed event was triggered.

Example

```
LocalVars scanCode
scanCode = GetEventScanCode
```

QT Version 5.0 or later

See also GetEventMouseX, GetEventMouseY, GetEventModifiers, GetEventKey

GetMemoryFree

Syntax

```
GetMemoryFree
```

Description This property returns the number of bytes of free memory available.

Return A numeric value indicating the number of bytes of free memory available.

Example

```
LocalVars memoryFree
memoryFree = GetMemoryFree
```

QT Version 5.0 or later

GetNetworkStatus

Syntax

```
GetNetworkStatus
```

Description This property returns the current Network Status.

The following Network Status defines are available:

NETWORK STATUS	MEANING
`kNoNetwork`	Network not available
`kNotConnected`	Network available but not connected
`kConnected`	Network available and connected
`kUncertain`	Unknown Network availability

Return A numeric value indicating the Network Status.

Example

```
If (GetNetworkStatus = kConnected)
// Launch URL only if already connect to
// to the network
GotoURL(“http://www.totallyhip.com”)
```

QT Version 5.0 or later

GetSystemVersion

Syntax `GetSystemVersion`

Description This property returns the BCD (binary coded decimal) encoded system version number.

Return A numeric value indicating the BCD encoded system version number. The following values are returned by various OS versions:

OS	VALUE RETURNED
MacOS 8.x	2144
MacOS 9.x	2308
MacOS X	0
Windows 9x	65536
Windows NT	131072

QT Version 5.0 or later

See also Version

GMTDay

Syntax `GMTDay`

Description This property returns the current day in GMT (Greenwich mean time). The day is returned as a number from 1 to 31. You can use this with the LocalDay property to determine what time zone the user is in.

Return A numeric value specifying the current day in GMT.

Example

```
LocalVars theDay
theDay = GMTDay
```

QT Version 4.0 or later

See also GMTMonth, GMTYear, LocalDay, LocalYear

GMTHours

Syntax `GMTHours`

Description This property returns the current hour in GMT (Greenwich mean time). The hour is returned as a number from 0 to 23. You can use this with the LocalHours property to determine what time zone the user is in.

Return A numeric value specifying the current hour in GMT.

Example

```
LocalVars localHours
localHours = GMTHours
```

QT Version 4.0 or later

See also GMTMinutes, GMTSeconds, LocalHours, LocalMinutes, LocalSeconds

GMTMinutes

Syntax `GMTMinutes`

Description This property returns the current minute in GMT (Greenwich mean time). The minute is returned as a number from 0 to 59. You can use this with the LocalMinutes property to determine what time zone the user is in.

Return A numeric value specifying the current minute in GMT.

Example

```
LocalVars localMins
localMins = GMTMinutes
```

QT Version 4.0 or later

See also GMTHours, GMTSeconds, LocalHours, LocalMinutes, LocalSeconds

GMTMonth

Syntax `GMTMonth`

Description This property returns the current month in GMT (Greenwich mean time). The month is returned as a number from 1 to 12. You can use this with the LocalMonth property to determine what time zone the user is in.

Return A numeric value specifying the current month in GMT.

Example

```
LocalVars theMonth
theMonth = GMTMonth
```

QT Version 4.0 or later

See also GMTDay, GMTYear, LocalDay, LocalMonth

GMTSeconds

Syntax `GMTSeconds`

Description This property returns the current second in GMT (Greenwich mean time). The second is returned as a number from 0 to 59.

Return A numeric value specifying the current seconds in GMT.

Example

```
LocalVars localSeconds
localSeconds = GMTSeconds
```

QT Version 4.0 or later

See also GMTHours, GMTMinutes, LocalHours, LocalMinutes, LocalSeconds

GMTYear

Syntax `GMTYear`

Description This property returns the current year in GMT (Greenwich mean time). The year is returned as a 4 digit number.

Return A numeric value specifying the current year in GMT.

Example

```
LocalVars theYear
theYear = GMTYear
```

QT Version 4.0 or later

See also GMTDay, GMTMonth, LocalDay, LocalMonth

HandlerRef

Syntax `HandlerRef`

Description This property returns the ID of the sprite that is executing this script. This is used when you create a sprite using the MakeNewSprite action. When you call the MakeNewSprite action you specify a sprite that contains the scripts for the new sprite. When those scripts are executed for the new sprite you can check this property to see what sprite is executing them.

Return Numeric value that is the ID of the sprite executing this script.

Example

```
SpriteOfID(HandlerRef).SetVisible(FALSE)
```

QT Version 4.0 or later

See also MakeNewSprite

IsRegistered

Syntax `IsRegistered(version)`

version A numeric literal or variable indicating the QuickTime major version number to be tested (i.e. 5)

Description This property returns TRUE if the copy of QuickTime is registered (i.e. QuickTime Pro).

Return Boolean value indicating TRUE if the specified version of QuickTime Pro is being used.

Example

```
If (IsRegistered(5))
// Script executes only if QT Pro version 5.x
// is begin used
EndIf
```

QT Version 5.0 or later

See also Registered

KeyIsDown

Syntax `KeyIsDown(modifiers, key)`

modifiers One of the modifier constants listed below.

key A string literal containing the key to be tested.

Description This property returns a boolean value of TRUE if the keyboard key specified by the lowercase key is being held down. Modifiers is used to specify keyboard modifier keys (i.e. Shift key, Control key, etc.) that are held down in conjunction with the key specified by key. Extra modifiers can be pressed and TRUE will still be returned.

The following key string defines are available for specialized keys:

KEY STRING DEFINE

```
kReturnKey
kTabKey
kDeleteKey
kUpArrowKey
kDownArrowKey
kLeftArrowKey
kRightArrowKey
kDoubleQuoteCharacter
kLineFeedCharacter
kCRLFCharacters
kCRCharacter
kLFCharacter
```

Don’t use these Key String Defines with the GetEventKey action. This action works with key codes and not key strings.

The following modifier constants are available:

KEY MODIFIER FLAG	MEANING
`kNone`	No keyboard modifiers used
`kOptionKey`	Option key held down (Alt for Windows)
`kShiftKey`	Shift key held down
`kCommandKey`	Command key held down (Mac)
`kControlKey`	Control key held down
`kCapsLockKey`	Caps lock key held down

Don’t use these Key Modifier Flags with the GetEventModifiers action. This action works with a different set of key modifier flags (see GetEventModifiers).

Multiple modifier constants can be specified by separating them with the “|” operator.

Return Boolean value of TRUE if the specified key is held down, FALSE otherwise.

Example

```
If (KeyIsDown(kNone, ‘a’) = TRUE)
// Script to execute if the “a” key is held down // by itself
EndIf

If (KeyIsDown(kShiftKey, ‘a’) = TRUE)
// Script to execute if the “a” key is held down
// along with the shift key
EndIf

If (KeyIsDown(kShiftKey | kControlKey, ‘a’) = TRUE)
// Script to execute if the “a” key is held down
// along with the shift and control keys
EndIf
```

QT Version 3.0 or later

See also Boolean Literal, String Literal, Built-in Constants

LocalDay

Syntax `LocalDay`

Description This property returns the current day in local time. The day is returned as a number from 1 to 31.

Return A numeric value specifying the current day in local time.

Example

```
LocalVars localDayVar
localDayVar = LocalDay
```

QT Version 4.0 or later

See also GMTHours, GMTMinutes, LocalHours, LocalMinutes, LocalSeconds

LocalHours

Syntax `LocalHours`

Description This property returns the current hour in local time. The hour is returned as a number from 0 to 23.

Return A numeric value specifying the current hour in local time.

Example

```
LocalVars localHourVar
localHourVar = LocalHours
```

QT Version 4.0 or later

See also GMTMinutes, GMTHours, GMTSeconds, LocalMinutes, LocalSeconds

LocalMinutes

Syntax `LocalMinutes`

Description This property returns the current minute in local time. The minute is returned as a number from 0 to 59.

Return A numeric value specifying the current minute in local time.

Example

```
LocalVars localMinutesVar
localMinutesVar = LocalMinutes
```

QT Version 4.0 or later

See also GMTMinutes, GMTHours, GMTSeconds, LocalMinutes, LocalHours

LocalMonth

Syntax `LocalMonth`

Description This property returns the current month in local time. The month is returned as a number from 1 to 12.

Return A numeric value specifying the current month in local time.

Example

```
LocalVars localMonthVar
localMonthVar = LocalMonth
```

QT Version 4.0 or later

See also GMTDay, GMTMonth, GMTYear, LocalDay, LocalYear

LocalSeconds

Syntax `LocalSeconds`

Description This property returns the current second in local time. The second is returned as a number from 0 to 59.

Return A numeric value specifying the current second in local time.

Example

```
LocalVars localSecondsVar
localSecondsVar = LocalSeconds
```

QT Version 4.0 or later

See also GMTMinutes, GMTHours, GMTSeconds, LocalMinutes, LocalHours

LocalYear

Syntax `LocalYear`

Description This property returns the current year in local time. The year is returned as a 4 digit number.

Return A numeric value specifying the current year in local time.

Example

```
LocalVars localYearVar
localYearVar = LocalYear
```

QT Version 4.0 or later

See also GMTDay, GMTMonth, GMTYear, LocalDay, LocalMonth

MouseButtonDown

Syntax `MouseButtonDown`

Description This property returns a boolean value of TRUE indicating that the mouse button (left mouse button for Windows) is pressed, otherwise it returns FALSE.

Return Boolean value of TRUE if the mouse button is pressed, otherwise FALSE.

Example

```
If (MouseButtonDown)
// do something when button is pressed
EndIf
```

QT Version 3.0 or later

See also Boolean Literal

Platform

Syntax `Platform`

Description This property indicates what platform the movie is being run on. Currently only the Macintosh and Windows platforms are recognized.

Return A value of 1 is returned to indicate that the user is running on a Macintosh and a value of 2 is returned to indicate that they are running on a Windows machine.

Example

```
If (Platform = 1)
 // its a Mac
EndIf
```

QT Version 4.0 or later

Registered

Syntax `Registered`

Description This property returns TRUE if QuickTime is registered. This means that the user has QuickTime Pro installed.

Return A boolean value of TRUE if QuickTime is registered otherwise FALSE is returned.

Example

```
If (Registered = TRUE)
 // do something
EndIf
```

QT Version 4.0 or later

See also IsRegistered, Version

Subscription

Syntax `Subscription(`*`channel_name`*`)`

`channel_name` A string literal identifying the channel by its URL.

Description This property returns TRUE if the named channel is currently subscribed to, otherwise it returns FALSE. Channel subscriptions show up in the pull out tray in the QuickTime player. Each channel is shown as an icon.

Return A boolean value of TRUE if the specified channel is subscribed to otherwise FALSE is returned.

Example

```
If (Subscription("My Test Channel") = TRUE)
```

QT Version 4.0 or later

See also AddSubscription, RemoveSubscription

TickCount

Syntax `TickCount`

Description This property returns the number of ticks (1/60 second) that have elapsed since the machine was started. You can use this for more accurate timing than you can get by using the idle handler.

Return A numeric value specifying how many ticks (1/60 second) have elapsed since the machine started up.

Example

```
x = TickCount + 5
While (x > TickCount)
```

```
// do something for 5 ticks
EndWhile
```

QT Version 4.0 or later

Version

Syntax `Version`

Description This property returns the version of QuickTime installed. You can use this to determine if you can execute QScripts that require QuickTime 4 in order to work. You should only use greater than or less than comparisons so that your scripts will still work when a new version of QuickTime becomes available.

Return A numeric value indicating QuickTime version number.

Example

```
If (Version = 3)
// QuickTime 3
ElseIf (Version > 4)
// QuickTime version 4 or greater
EndIf
```

QT Version 4.0 or later

See also Registered

General Actions

AddSubscription

Syntax `AddSubscription(name, URL, pictures_URL)`

name A string literal or variable containing the name of the channel.

URL A string literal or variable containing the URL to the channel.

pictures_URL A string literal or variable containing the URL to a picture for the channel.

Description This action subscribes the user to the named channel with the given URL. The name parameter designates what name will be used for the subscription. The URL parameter refers to the media which the user is subscribing to. The pictures_URL parameter refers to a Gif image that is 33 x 28 pixels in size.

Example

```
AddSubscription("Hip", "www.totallyhip.com",
"www.totallyhip.com/channel.qif")
```

QT Version 4.0 or later

See also String Literal, RemoveSubscription, SetString, AppendString

AppendString

Syntax `AppendString(Variable_1, Variable_2, Variable_Result)`

Variable_1 A variable name as declared in a variable declaration.

Variable_2 A variable name as declared in a variable declaration.

Variable_Result A variable name as declared in a variable declaration.

Description This action sets the contents of the result variable to the concatenation of Variable_1 and Variable_2. All three parameters must be variable names.

Example

```
LocalVars firstString, secondString
LocalVars resultString
SetString(firstString, “I am “)
SetString(secondString, “Locutous of Borg.”)
AppendString(firstString, secondString, resultString)
```

In this case the result string will contain the string “I am Locutous of Borg.”

QT Version 4.0 or later

See also String Literal, SetString

ApplicationNumberAndString

Syntax `ApplicationNumberAndString(number,string)`

`number` A numeric expression.

`string` A string literal or variable.

Description This action sends the number and string to the application playing the movie.

Example The following statement causes the number 1 and the string “Here I am !” to be sent to the application playing the movie:

```
ApplicationNumberAndString(1, “Here I am!”)
```

QT Version 3.0 or later

See also Numeric Expression, String Literal, DebugStr

CloseThisWindow

Syntax `CloseThisWindow`

Description This action sends a message to the application playing the movie (QuickTime Player) indicating that the current movie window is to be closed.

QT Version 5.0 or later

See also DisplayChannels, EnterFullScreen, ExitFullScreen, SendAppMessage, SoftwareWasChanged

DebugStr

Syntax `DebugStr(`*`message`*`)`

`message` A string literal or variable.

Description This action sends a string to the application playing the movie. The parameter message contains the debug string to send. The string can be no longer than 255 characters.

Example The following statement causes the string “Hello World!” to be sent to the application playing the movie:

```
DebugStr(“Hello World!”)
```

QT Version 3.0 or later

See also String Literal, ApplicationNumberAndString

DisplayChannels

Syntax `DisplayChannels`

Description This action sends a message to the application playing the movie (QuickTime Player) indicating that the QuickTime TV channels should be displayed.

QT Version 5.0 or later

See also CloseThisWindow, EnterFullScreen, ExitFullScreen, SendAppMessage, SoftwareWasChanged

EnterFullScreen

Syntax `EnterFullScreen`

Description This action sends a message to the application playing the movie (QuickTime Player) indicating that the current movie window is to enter full-screen mode.

QT Version 5.0 or later

See also CloseThisWindow, DisplayChannels, ExitFullScreen, SendAppMessage, SoftwareWasChanged

ExecuteAppleScript

Syntax `ExecuteAppleScript(`*`string`*`)`

`string` A string literal or variable that contains the text of the AppleScript to execute.

Description This action is used only by LiveStage Professional at this time. With it you can create Tool Movies that can be placed in the Tools menu and used to control LiveStage Professional.

Example `ExecuteAppleScript("quit")`

QT Version 4.0 or later

See also String Literal, Constants, SetString, AppendString

ExecuteGenericScript

Syntax `ExecuteGenericScript(`*`cmd, args`*`)`

`cmd` A string literal or variable containing the command.

args A string literal or variable containing the arguments to be passed into the command being executed.

Description It is up to the application that is playing the QuickTime movie to interpret this command.

Example The following statement causes the command "DoAction" to be sent to the application playing the movie:

```
ExecuteGenericScript("DoAction", "argument")
```

QT Version 4.1 or later

See also ExecuteVBScript, ExecuteJavaScript, ExecuteLingoScript

ExecuteJavaScript

Syntax `ExecuteJavaScript(cmd, args)`

cmd A string literal or variable containing the command.

args A string literal or variable containing the arguments to be passed into the command being executed.

Description It is up to the application that is playing the QuickTime movie to interpret this command.

Example The following statement causes the command “DoAction” to be sent to the application playing the movie:

```
ExecuteJavaScript(“DoAction”, “argument”)
```

QT Version 4.1 or later

See also ExecuteGenericScript, ExecuteVBScript, ExecuteLingoScript, ExecuteLingoScript

ExecuteLingoScript

Syntax `ExecuteLingoScript(cmd, args)`

cmd A string literal or variable containing the command.

args A string literal or variable containing the arguments to be passed into the command being executed.

Description It is up to the application that is playing the QuickTime movie to interpret this command.

Example The following statement causes the command “DoAction” to be sent to the application playing the movie:

```
ExecuteLingoScript(“DoAction”, “argument”)
```

QT Version 4.1 or later

See also ExecuteGenericScript, ExecuteJavaScript, ExecuteVBScript

ExecuteProjectorScript

Syntax `ExecuteProjectorScript(cmd, args)`

cmd A string literal or variable containing the command.

args A string literal or variable containing the arguments to be passed into the command being executed.

Description It is up to the application that is playing the QuickTime movie to interpret this command.

Example The following statement causes the command “DoAction” to be sent to the application playing the movie:

```
ExecuteProjectorScript(“DoSoAction”argument”)
```

QT Version 4.1 or later

See also ExecuteGenericScript, ExecuteJavaScript, ExecuteLingoScript

ExecuteVBScript

Syntax `ExecuteVBScript(cmd, args)`

cmd A string literal or variable containing the command.

args A string literal or variable containing the arguments to be passed into the command being executed.

Description It is up to the application that is playing the QuickTime movie to interpret this command.

Example The following statement causes the command “DoAction” to be sent to the application playing the movie:

```
ExecuteVBScript(“DoAction”, “argument”)
```

QT Version 4.1 or later

See also ExecuteGenericScript, ExecuteJavaScript, ExecuteLingoScript

ExitFullScreen

Syntax `ExitFullScreen`

Description This action sends a message to the application playing the movie (QuickTime Player) indicating that the current movie window is to exit full-screen mode and return to the window mode.

QT Version 5.0 or later

See also CloseThisWindow, DisplayChannels, EnterFullScreen, SendAppMessage, SoftwareWasChanged

GoToURL

Syntax `GoToURL(URL)`

URL A string literal or variable that contains the URL to go to.

Description The GoToURL action is used to get content from either a Web Address (Internet/LAN) or local storage (CD-ROM or hard disk, etc.). Although its name indicates that its primary purpose is for Web Addresses there is a great deal of functionality incorporated into this action for non Web based operations.

By using the GoToURL action you can have the default system Web Browser display the specified URL, target a frame in your Web Page, launch the QuickTime (c) Player, etc. One caveat regarding this action is that all file specifications must be fully qualified (i.e. no referential paths).

This example will launch the default Web Browser on the target system and display the contents of the Totally Hip Web Site.

```
GoToURL(“http://www.totallyhip.com”)
```

You can also target a specific frame in the Web Browser (if it is already running) by using the following syntax:

```
GoToURL("<URL>T<Frame>")
```

Where Frame is the frame name.

To have QuickTime open the URL in a new window use the following GoToURL call.

```
GoToURL("<URL>T<_blank>")
```

A variation on the above tells QuickTime to open the URL in the QuickTime Player.

```
GoToURL("<URL>T<QuickTimePlayer>")
```

Another variation tells QuickTime to open the URL in the Default Web Browser.

```
GoToURL("<URL>T<webbrowser>")
```

To reference a file using the GoToURL action you would enter a call using a file specification. The syntax for the call is:

```
GoToURL("file:///<filename>")
```

Where filename is the platform specific file path. On the Mac this is volume:folder:file and on Windows it is volume:\folder\file.

Example The following statement causes the default Web Browser to be launched and selects "www.totallyhip.com" to be the current URL.

```
GotoURL("http://www.totallyhip.com")
```

The following statement causes the specified URL (www.totallyhip.com) to be loaded into the specified Frame (SideBar):

```
GotoURL("<http://www.totallyhip.com>T<SideBar>")
```

QT Version 3.0 or later

See also String Literal, SetString, AppendString, SetStatusString

LoadComponent

Syntax `LoadComponent(component)`

component A numeric literal indicating the ID of the component to load.

Description This action causes the specified component to be loaded by QuickTime. This action initiates the loading of the specified QuickTime component. ComponentVersion should be called at a later time to ensure that the specified component is loaded.

This action allows the management of the automatic component download process provided by QuickTime 5.

QT Version 5.0 or later

See also ComponentVersion

RemoveSubscription

Syntax `RemoveSubscription(URL)`

URL A string literal or variable containing the URL to the channel.

Description This action removes the subscription from the specified URL. The user must be subscribed to the specified URL already, otherwise no action is performed.

Example `RemoveSubscription("http://www.totallyhip.com")`

QT Version 4.0 or later

See also String Literal, AddSubscription

SendAppMessage

Syntax `SendAppMessage(`*`message`*`)`

`message` An integer expression indicating the application message ID to send.

Description This action sends the specified message ID to the application playing the movie.

AVAILABLE APPLICATION MESSAGE IDS

```
kAppMsgSoftwareChanged
kAppMsgRequestWindowClose
kAppMsgRequestExitFullScreen
kAppMsgDisplayChannels
kAppMsgRequestEnterFullScreen
```

All of the above message IDs causes the QuickTime Player to perform the specified task. If you are not using QuickTime Player (QT 5 or later) to play the movie, your results will vary depending on the level of support offered by your player application.

Example

```
SendAppMessage(kAppMsgRequestWindowClose)
// Tells QuickTime Player to close
```

QT Version 5.0 or later

See also DisplayChannels, EnterFullScreen, ExitFullScreen, SoftwareWasChanged, WindowThisClose

SetCursor

Syntax `SetCursor(`*`cursor_ID`*`)`

`cursor_ID` ID of the cursor to use.

Description This action sets the current cursor. The available cursor IDs are as follows:

```
kQTCursorOpenHand
kQTCursorClosedHand
kQTCursorPointingHand
kQTCursorRightArrow
kQTCursorLeftArrow
kQTCursorDownArrow
kQTCursorUpArrow
kQTCursorArrow
```

Example `SetCursor(kQTCursorPointingHand)`

QT Version 4.0 or later

See also String Literal, Constants, SetString, AppendString

SetStatusString

Syntax `SetStatusString(status, Flags)`

`status` A string literal or variable that contains the status string.

`Flags` Flags for the status string.

Description This action instructs the QuickTime plug-in to display the status string in the status area of the browser. The flags should be set to kStatusURL to indicate that this status string is a URL link. You can set it to kStatusStreaming if the string is a streaming status.

Example

```
SetStatusString(“Loading...”, kStatusURL)
```

QT Version 4.0 or later

See also String Literal, Constants, SetString, AppendString

SetString

Syntax `SetString(Variable, String)`

`Variable` A variable name as declared in a variable declaration.

`String` A string literal or numeric expression that will be converted to a string.

Description This action sets the contents of the variable to the provided string or numeric expression converted to a string.

Example

```
LocalVars myString
SetString(myString, “I am Locutous of Borg”)
```

QT Version 4.0 or later

See also String Literal, AppendString, SetStatusString

SoftwareWasChanged

Syntax `SoftwareWasChanged`

Description This action sends a message to the application playing the movie (MoviePlayer) indicating that the installed QuickTime software has been updated.

QT Version 5.0 or later

See also CloseThisWindow, DisplayChannels, EnterFullScreen, ExitFullScreen, SendAppMessage

MOVIE PROPERTIES AND ACTIONS

Movie properties and actions require a movie target. If no movie target is specified then the current movie is considered to be the target. You can specify a movie target by using MovieNamed(name) or MovieOfID(id) and pass in either the name of the movie or its ID. See the section on targets for a complete description and some examples of how to specify a target.

Movie Properties

GetDuration

Syntax `GetDuration`

Description This property returns the duration of the movie in the time scale of the movie (typically 600 units per second).

Return A numeric value representing the duration of the movie.

Example

```
LocalVars movieLengthInSeconds
movieLengthInSeconds = GetDuration / GetTimeScale
// Calculate the length of the movie in seconds
```

QT Version 5.0 or later

See also GetTimeScale

GetHeight

Syntax `GetHeight`

Description This property returns the height of the movie in pixels.

Return A numeric value representing the height of the movie.

QT Version 5.0 or later

See also GetWidth

GetLoadState

Syntax `GetLoadState`

Description This property returns the current Load State of the movie.

The following defines are available for the different load states:

LOAD STATE DEFINE	MEANING
`kLoading`	Movie is still loading
`kPlayable`	Movie is now playable
`kPlaythroughOK`	Movie has cached enough data to play through to the end
`kComplete`	Movie is completely loaded
`kLoadStateError`	Load error

Return A numeric value indicating the Load State of the movie.

Example

```
// Start playing the movie as soon as it is playable
If (GetLoadState = kPlayable AND MovieRate <> 0)
StartPlaying
EndIf
```

QT Version 5.0 or later

See also MaxLoadedTimeInMovie

GetID

Syntax `GetID`

Description This property returns the ID of the movie. The movie ID of a movie can be set in the Info tab of LiveStage Professional.

Return A numeric value indicating the ID of the movie.

Example

```
LocalVars movieID
movieID = ThisMovie.GetID
// Retrieve the ID of the movie
```

QT Version 5.0 or later

See also GetName

GetName

Syntax `GetName`

Description This property returns the name of the movie. The movie name of a movie can be set in the Info tab of LiveStage Professional.

Return A string value indicating the name of the movie.

Example

```
LocalVars movieName
movieName = ThisMovie.GetName
// Retrieve the name of the movie
```

QT Version 5.0 or later

See also GetID

GetTimeScale

Syntax `GetTimeScale`

Description This property returns the Time Scale of the movie. This value is typically 600 units per second. The value of all movie and track durations are scaled by the Time Scale of the movie.

For Example, a duration of 4 seconds in a movie with the Time Scale of 600 would be represented as follows:
Scaled Duration = 4 seconds * 600 units per second = 2400

Return A numeric value representing the Time Scale of the movie.

Example

```
LocalVars movieLengthInSeconds
movieLengthInSeconds = GetDuration / GetTimeScale
// Calculate the length of the movie in seconds
```

QT Version 5.0 or later

See also GetDuration

GetTrackCount

Syntax `GetTrackCount`

Description This property returns the number of tracks in the movie.

Return A numeric value indicating the number of tracks in the movie.

Example

```
LocalVars trackCount
// Disable all tracks in the movie
For trackCount = 1 to GetTrackCount
TrackOfIndex(trackCount).SetEnabled(FALSE)
Next
```

QT Version 5.0 or later

GetVariable

Syntax `GetVariable(`*`Address`*`)`

`Address` A numeric expression specifying the address of a movie variable.

Description Gets the value of the variable that is located at the specified address.

Movie variables (those variables created using MovieVars) are stored within their own sprite track, nothing else is placed into this track. This track always has the name “Movie Variables” so that you can get and set variables in this track from another movie. This allows for data transfer from one movie to another.

The primary purpose of this command is to enable data transfer between movies. Variables that specify an address can be between the address range of 1 to 10,000. Non-addressed variables are stored from the address 10,001 and on.

To assign an address to a variable you would define it using the following form:

```
MovieVars variable_name : address
```

The variable name is any valid identifier for your variable, the address field will contain a number in the range of 1 - 10,000. This will ensure that variable is stored at that location. The following gives an example of an array of 50 items which is located at address 200, thus the full address range for the elements in the array would be 200 - 249.

```
MovieVars myArray[50]:200
```

Note that for more technical readers you may notice that you can have variables defined in such a way that you actually have different representations of your data. For example, you can define two variables that access the same address areas.

```
MovieVars firstArray[50]:200
MovieVars secondVar:220
```

The example above creates two variables, the first being an array containing 50 elements which are stored at address 200 on. The second variable points to address 220, thus referring to item 21 in the array.

Return A value representing the content of the movie variable specified.

Example `x = MoveNamed(“Remote”).GetVariable(200)`

QT Version 3.0 or later

See also SetVariable

GetWidth

Syntax `GetWidth`

Description This property returns the width of the movie in pixels.

Return A numeric value representing the width of the movie.

QT Version 5.0 or later

See also GetHeight

IsMovieActive

Syntax `IsMovieActive`

Description This property returns TRUE if the movie is active.

Return A boolean value indicating the active state of the movie.

QT Version 5.0 or later

MaxLoadedTimeInMovie

Syntax `MaxLoadedTimeInMovie`

Description This property returns an integer indicating the amount of the movie that has been downloaded so far. The value returned is in terms of the time scale of the movie (usually 600). To calculate the loaded time in seconds, divide the value returned by the time scale of the movie (usually 600).

Return Numeric value indicating the loaded time of the movie using the time scale of the movie.

Example

```
If (MaxTimeLoadedInMovie > SampleNamed(“My Sample”).StartTime)
// Script executes only when the
// MaxTimeLoadedInMovie is greater than the
// start time of the Sample named “My Sample”
EndIf
```

QT Version 4.0 or later

See also MovieTime, GotoTime

MovieIsLooping

Syntax `MovieIsLooping`

Description This property returns a boolean value of TRUE if the movie is set to loop when it reaches the end otherwise it returns FALSE.

Return Boolean value of TRUE if the movie is set to loop otherwise returns FALSE.

Example

```
If (MovieIsLooping = TRUE)
// Script executes only when the movie is looping
EndIf
```

QT Version 3.0 or later

See also Boolean Literal, SetLoopingFlag

MovieLoopIsPalindrome

Syntax `MovieLoopIsPalindrome`

Description This property returns a boolean value of TRUE if the movie is set to loop in a palindrome mode otherwise it returns FALSE. Palindrome looping mode plays the movie to the end then plays it backwards until the beginning then repeats the cycle.

Return Boolean value of TRUE if the loop mode of the movie is Palindrome otherwise FALSE is returned.

Example

```
If (MovieLoopIsPalindrome = TRUE)
// Script executes only when loop mode is
// Palindrome
EndIf
```

QT Version 3.0 or later

See also Boolean Literal, SetLoopingFlag

MovieRate

Syntax `MovieRate`

Description This property returns the current playback rate of a movie. The value returned is in terms of the normal playback speed of the movie. Negative values indicate that the movie is playing backward. The following lists the meaning of possible Movie Rate values.

MOVIE RATE	MEANING
1.0	Playing forward at normal speed.
0	Stopped.
-1.0	Playing backward at normal speed.
1.5	Playing forward at 1.5x normal speed.
-4.0	Playing backward at 4x normal speed.

Return Signed numeric value indicating the current playback rate of the movie.

Example

```
LocalVars theRate
theRate = MovieRate
```

QT Version 3.0 or later

See also SetRateTo, SetRateBy, StartPlaying, StopPlaying

MovieTime

Syntax `MovieTime`

Description This property returns an integer indicating the current time of the movie. The value returned is in terms of the time scale of the movie (usually 600). To calculate the movie time in seconds, divide the value returned by the time scale of the movie (usually 600).

Return Numeric value indicating the current time of the movie using the time scale of the movie.

Example

```
If (MovieTime > SampleNamed(“My Sample”).StartTime)
// Script executes only when MovieTime is later
// than the start time of the Sample
// named "My Sample"
EndIf
```

QT Version 3.0 or later

See also GotoTime

MovieVolume

Syntax `MovieVolume`

Description This property returns the current volume level of a movie. The value returned is between 0 and 255 with 0 indicating silence and 255 indicating maximum volume. The volume settings of the individual tracks are all proportional to this master volume setting.

Return Numeric value between 0 and 255.

Example

```
LocalVars theVolume
theVolume = MovieVolume
```

QT Version 3.0 or later

See also TrackVolume, SetMovieVolume

Movie Actions

GetMovieURL

Syntax `GetMovieURL(string_var)`

string_var The name of a variable to receive the string value for the URL.

Description This action retrieves the URL of the movie placing it in the variable specified by the parameter *string_var*.

Obsolete—use GetParentMovieURL instead.

Example

```
LocalVars theMovieURL
GetMovieURL(theMovieURL)
```

QT Version 4.0 or later

See also GotoURL, SetString, AppendString

GetParentMovieURL

Syntax `GetParentMovieURL(`*string_var*`)`

string_var This is the name of a string variable to receive the URL for the parent movie.

Description This action retrieves the URL of the parent movie that contains this child movie issuing the command. You can then use this string to build a new URL relative to the parent movie in order to load in other assets.

Example

```
LocalVars urlString
GetParentMovieURL(urlString)
```

QT Version 4.1 or later

See also GetRootMovieURL

GetRootMovieURL

Syntax `GetRootMovieURL(`*`string_var`*`)`

`string_var` This is the name of a string variable to receive the URL for the root movie.

Description This action retrieves the URL of the root movie. You can then use this string to build a new URL relative to the root movie in order to load in other assets.

Example

```
LocalVars urlString
GetRootMovieURL(urlString)
```

QT Version 4.1 or later

See also GetParentMovieURL

GoToBeginning

Syntax `GoToBeginning`

Description This action rewinds a movie to its beginning.

Example The following statement causes the current movie to be rewound to the beginning:

```
GoToBeginning
```

QT Version 3.0 or later

See also GoToEnd, GotoTime, MovieTime

GoToEnd

Syntax `GoToEnd`

Description This action goes to the end of the movie.

Example The following statement causes the current movie to go to the end:

```
GoToEnd
```

QT Version 3.0 or later

See also GoToBeginning, GotoTime, MovieTime

GoToTime

Syntax `GoToTime(`*`time`*`)`

`Time` A numeric expression indicating the time in 600ths of a second. You can enter times in your scripts as mm:ss.hhh, (minutes:seconds.fraction of a second in 600ths).

Description This action jumps to a new time in the movie. To specify a movie time of 2 seconds, you would have to specify 1200 (2 seconds x 600 units per second or 00:02.000).

Example The following statements cause the current movie to jump to the 10 second position:

```
GoToTime(6000)
GotoTime(0:10.0)
```

The following statements cause the current movie to jump to the 5.5 second position:

```
GoToTime(600 * 5.5)
GotoTime(0:5.300)
```

QT Version 3.0 or later

See also GotoBeginning, GotoEnd, MovieTime

GoToTimeByName

Syntax `GoToTimeByName(`*`name`*`)`

`Name` A string literal specifying the name of a chapter in the movie.

Description This action causes a movie to jump to the time of a chapter mark with the specified name.

In order to use this action the movie must have been created with chapter marks.

Example The following statement causes the current movie to jump to the "Introduction" chapter mark:

```
GoToTimeByName("Introduction")
```

QT Version 3.0 or later

See also GotoTime, GotoBeginning, GotoEnd, MovieTime

MovieChanged

Syntax `MovieChanged`

Description This action causes a notification to be sent to the application playing the movie indicating that the movie has been changed.

This notification would cause QuickTime Player to update its state.

QT Version 3.0 or later

PopAndGotoLabeledTime

Syntax `PopAndGotoLabeledTime(`*`label`*`)`

`Label` A string literal specifying the label of a saved time.

Description This action sets the current time of the movie to the time specified by the label. No action is performed if there is no time that matches the specified label.

Example PopAndGotoLabeledTime("Intro Sequence")

QT Version 3.0 or later

See also PushCurrentTime, PushCurrentTimeWithLabel, PopAndGotoTopTime

PopAndGotoTopTime

Syntax `PopAndGotoTopTime`

Description This action sets the current time of the movie to the last saved time.

Example `PopAndGotoTopTime`

QT Version 3.0 or later

See also PushCurrentTime, PushCurrentTimeWithLabel, PopAndGotoLabeledTime

PushCurrentTime

Syntax `PushCurrentTime`

Description This action saves the current time of the movie. The time may be retrieved later with the PopAndGotoTopTime and PopAndGotoLabeledTime commands.

Example `PushCurrentTime`

QT Version 3.0 or later

See also PushCurrentTimeWithLabel, PopAndGotoTopTime, PopAndGotoLabeledTime

PushCurrentTimeWithLabel

Syntax `PushCurrentTimeWithLabel(`*`label`*`)`

Label A string literal specifying the label for this saved time.

Description This action saves the current time of the movie and attaches a label to it so that it may be referred to by name at some later point.

Example `PushCurrentTimeWithLabel(“Intro Sequence”)`

QT Version 3.0 or later

See also PushCurrentTime, PopAndGotoTopTime, PopAndGotoLabeledTime

SetLanguage

Syntax `SetLanguage(`*`language`*`)`

`Language` A numeric expression indicating the language to use. Ranges from 0 (English) to 150 (Greenlandic).

Description This action sets the current language of the movie. This will cause tracks with the same groups to have the track with the matching language selected and enabled. If no track with the correct language can be found then the language is not changed.

Example `SetLanguage(1) // set language to french`

QT Version 3.0 or later

See also TrackEnabled

SetLoopingFlags

Syntax `SetLoopingFlags(`*`flags`*`)`

`Flags` A numeric constant specifying the new looping flags.

Description This action sets the loop mode of a movie. The following are possible loop mode settings and their associated define:

DEFINE	LOOP MODE DESCRIPTION
`kNoLoop`	Movie will play forward once and stop.
`kLoop`	Movie will play forward to the end and then rewind and repeat.
`kLoopPalindrome`	Movie will play forward to the end then play backward until the beginning is reached and then repeat (ping pong).

Example The following statement causes the current movie to play forever:

```
SetLoopingFlags(kLoop)
```

The following statement causes the current movie to play in a ping pong fashion indefinitely:

```
SetLoopingFlags(kLoopPalindrome)
```

QT Version 3.0 or later

See also Defines, MoveIsLooping, MovieLoopIsPalindrome

SetPlaySelection

Syntax `SetPlaySelection(`*`play`*`)`

`Play` A boolean literal of TRUE to indicate that only the selection should play.

Description This action sets or clears the Play Selection mode of a movie. The play parameter is either TRUE or FALSE.

A value of TRUE indicates that the movie will play only the selection (see *`SetSelection and SetSelectionByName`*). A value of FALSE will clear the Play Selection mode and allow the entire movie to be played.

Example The following statements will cause the selection from 2 seconds to 10 seconds in the current movie to be played:

```
SetSelection(0:2.0, 0:10.0)
SetPlaySelection(TRUE)
```

QT Version 3.0 or later

See also SetSelection, TogglePlaySelection

SetRateBy

Syntax `SetRateBy(`*`rate`*`)`<***MIN***(number) ***MAX***(number) ***wraparound***>

`Rate` A numeric expression indicating the change in rate for the movie.

Description This action causes the playback speed of a move to be altered by the specified amount relative to the current playback rate. The parameter rate can be positive or negative (see SetRateTo above).

Example The following statement causes the playback speed of the current movie to increase by 1 full normal speed forwards:
SetRateBy(1)

QT Version 3.0 or later

See also SetRateTo, StartPlaying, StopPlaying, MovieRate

SetRateTo

Syntax `SetRateTo(`*`rate`*`)`<***MIN***(number) ***MAX***(number)>

`Rate` A numeric expression indicating the new rate for the movie.

Description This action sets the playback speed of a movie. The parameter Rate supplies the desired playback rate of the movie. The following lists some possible rate values and their meaning:

RATE	MEANING
0	Movie is stopped
1	Normal speed forward
-1	Normal speed in reverse

Example The following statement causes the current movie track to play at normal speed forwards:

```
SetRateTo(1)
```

The following statement causes the current movie track to play at double speed in reverse:

```
SetRateTo(-2)
```

The following statement causes the current movie track to stop:

```
SetRateTo(0)
```

QT Version 3.0 or later

See also StartPlaying, StopPlaying, SetRateBy, MovieRate

SetSelection

Syntax `SetSelection(`*`start, end`*`)`

`Start` A numeric expression indicating the start time of the selection.

End A numeric expression indicating the end time of the selection.

Description This action specifies a selection within a movie. The two parameters specify the start and end times of the selection. The time values are in terms of the time scale of the movie (see *`GoToTime`* above).

Example The following statements cause a selection from 2 seconds to 10 seconds to be marked as the selection in the current i.e.:

```
SetSelection(2 * 600, 10 * 600)
SetSelection(0:2.0, 0:10.0)
```

QT Version 3.0 or later

See also SetSelectionByName, GoToTime, SetPlaySelection, TogglePlaySelection

SetSelectionByName

Syntax `SetSelectionByName(`*`start, end`*`)`

`Start` A string literal indicating the start time by specifying a chapter name.

`End` A string literal indicating the end time by specifying a chapter name.

Description This action specifies a selection within a movie. The two parameters specify the chapter names of the beginning and end of the selection.

In order to use this action, the movie must have been created with chapter marks.

Example The following statement causes a selection from the beginning of the "Introduction" to the beginning of the "Chapter 4" to be marked as the selection in the current movie:

```
SetSelectionByName("Introduction", "Chapter 4")
```

QT Version 3.0 or later

See also SetSelection, SetPlaySelection, TogglePlaySelection

SetVariable

Syntax `SetVariable(Address, Value)`

`Address` A numeric expression specifying the address of a movie variable.

`Value` A numeric expression specifying the value to store.

Description Sets the value of the variable that is located at the specified address.

For detailed information on how variable address work you should refer to the GetVariable documentation.

The address parameter specifies an address value between 1 and 10,000. These are all of the movie variables in the movie.

The Value parameter contains the value that you want to store at the specified address.

Example

```
MovieNamed("My Movie").SetVariable(8, 32)
Sets the variable at address 8 to the value of 32.
```

QT Version 3.0 or later

See also GetVariable

SetVolumeBy

Syntax `SetVolumeBy(volume)`<***MIN***(number) ***MAX***(number) ***wraparound***>

`Volume` A numeric expression from -256 to 256 indicating the change in the volume setting.

Description This action alters the volume level of the movie by the specified amount. The parameter Volume supplies the value with which the volume level is changed up (positive value) or down (negative value).

Example The following statement causes the volume level of the movie to increase by 25:

```
SetVolumeBy(25)
```

QT Version 3.0 or later

See also Numeric Expression, SetVolumeTo, MusicVolume

SetVolumeTo

Syntax `SetVolumeTo(volume)`<***MIN***(number) ***MAX***(number)>

`Volume` A numeric expression from -256 (off) to 256 (full volume) indicating the new volume setting.

Description This action sets the volume level of the movie. The volume level of each track can be adjusted separately. The volume level of the movie controls the overall volume setting. A setting of 0 or less will mute the sound. You can use negative volume settings to keep track of the previous volume setting.

Example The following statement causes the volume of the movie to be turned off:

```
SetVolumeTo(0)
```

QT Version 3.0 or later

See also Numeric Expression, SetVolumeBy, MusicVolume

StartPlaying

Syntax `StartPlaying`

Description This action causes the movie to start playing at normal speed. It is a shortcut for SetRateTo(1.0).

Example `StartPlaying`

QT Version 3.0 or later

See also SetRateTo, StopPlaying, SetRateBy, MovieRate

StepBackward

Syntax `StepBackward`

Description This action steps a movie backward a pre-determined period of time. The movie will always step backward to the start time of the previous visual sample.

Example The following statement causes the current movie to be stepped backward:

```
StepBackward
```

QT Version 3.0 or later

See also StepForward, MovieTime

StepForward

Syntax `StepForward`

Description This action steps a movie forward a pre-determined period of time. The movie will always step forward to the start time of the next visual sample.

Example The following statement causes the current movie to be stepped forward:

```
StepForward
```

QT Version 3.0 or later

See also StepBackward, MovieTime

StopPlaying

Syntax `StopPlaying`

Description This action causes the movie to stop playing. It is a shortcut for SetRateTo(0).

Example `StopPlaying`

QT Version 3.0 or later

See also SetRateTo, SetRateBy, StartPlaying, MovieRate

TogglePlaySelection

Syntax `TogglePlaySelection`

Description This action toggles the Play Selection mode of a movie. If the current Play Selection mode of a movie is TRUE, calling TogglePlaySelection will change this to FALSE and vice versa.

Example The following statements will cause the current movie to turn off the Play Selection mode:

```
SetPlaySelection(TRUE)
TogglePlaySelection
```

QT Version 3.0 or later

See also SetPlaySelection, SetSelection, SetSelectionByName

TRACK PROPERTIES AND ACTIONS

Track properties need to have a track and movie target specified. If no movie target is specified then the current movie is considered to be the target. You can specify a movie target by using MovieNamed(name) or MovieOfID(id) and pass in either the name of the movie or its ID. If no track target is specified then the track that contains the object that is currently executing the script is considered to be the target. You can specify a track target by using TrackNamed(name), TrackOfID(id), TrackOfIndex(index) or TrackOfType(type). See the section on targets for a complete description and some examples of how to specify a target.

Track Properties are available for all track types within a movie.

Track Properties

GetDuration

Syntax `GetDuration`

Description This property returns the duration of the movie in the time scale of the movie (typically 600).

The `ThisTrack` target must be used in conjunction with the GetDuration property in order to obtain the duration of the track. By default, if no target is specified, GetDuration returns the duration of the movie.

Return A numeric value representing the duration of the movie.

Example

```
LocalVars trackLen
trackLen = TrackOfIndex(1).GetDuration /
GetTimeScale
// Calculate the duration of track 1 in seconds
```

QT Version 5.0 or later

See also GetTimeScale

GetHeight

Syntax `GetHeight`

Description This property returns the height of the Track.

An explicit track target (i.e. TrackOfIndex(1)) must be used in conjunction with the GetHeight property in order obtain the height of the track. By default, if no target is specified, GetHeight returns the width of the movie.

Return A numeric value representing the height of the track.

Example

```
LocalVars trackHeight
// Get the height of track 1
trackHeight = TrackOfIndex(1).GetHeight
```

QT Version 5.0 or later

See also GetWidth

GetID

Syntax `GetID`

Description This property returns ID of the track. The ID of a track is assigned when the movie is loaded and therefore cannot be set explicitly.

Return A numeric value indicating the ID of the track.

Example

```
LocalVars trackID
trackID = TrackOfIndex(1).GetID
// Retrieve the ID of track 1
```

QT Version 5.0 or later

See also GetTrackName

GetName

Syntax `GetName`

Description This property returns name of the track. The name of a track can be set in the Property Window of the track in LiveStage Professional.

Return A string value indicating the name of the track.

Example

```
LocalVars trackName
trackName = TrackOfIndex(1).GetName
// Retrieve the name of track 1
```

QT Version 5.0 or later

See also GetID

GetWidth

Syntax `GetWidth`

Description This property returns the width of the Track.

An explicit track target (i.e. TrackOfIndex(1)) must be used in conjunction with the GetWidth property in order obtain the width of the track. By default, if no target is specified, GetWidth returns the width of the movie.

Return A numeric value representing the width of the track.

Example

```
LocalVars trackWidth// Get the width of the track
trackWidth = TrackOfIndex(1).GetWidth
```

QT Version 5.0 or later

See also GetHeight

TrackEnabled

Syntax `TrackEnabled`

Description This property returns a boolean value of TRUE if the track is enabled otherwise FALSE is returned. Only enabled tracks are processed by QuickTime. If a visual track (i.e. sprite track, video track, picture track, etc...) is enabled, it is displayed in the window of the movie. If an audible track (i.e. MIDI track, Instrument track, etc...) is enabled, it is played when the movie is played.

Return Boolean value indicating TRUE if the track is enabled otherwise FALSE is returned.

Example

```
If (TrackOfID(2).TrackEnabled)
```

QT Version 3.0 or later

See also Boolean Literal, SetEnabled, ToggleEnabled

Track Actions

SetEnabled

Syntax `SetEnabled(`*`enable`*`)`

`Enable` A boolean expression resulting in TRUE to enable the track.

Description This action enables or disables a track. A disabled track will not draw, make sounds or have any wired actions executed.

Example The following statement enables the track:

```
TrackOfID(2).SetEnabled(TRUE)
```

QT Version 3.0 or later

See also ToggleEnabled, TrackEnabled

ToggleEnabled

Syntax `ToggleEnabled`

Description This action disables the track if it enabled or enables it if it is disabled.

Example The following statement causes the state of the current track to be inverted:

```
ToggleEnabled
```

QT Version 3.0 or later

See also SetEnabled, TrackEnabled

SAMPLE PROPERTIES

The sample properties must have the sample target specified. The sample must exist in the current document and must be in one of the built-in tracks types.

Sample Properties

EndTime

Syntax `EndTime`

Description Returns the end time of the sample. This value is calculated only when the script is compiled. This means that you can not pass in a string value to get the end time of a sample while your script is running.

Example `GoToTime(SampleNamed(“My Sample”).EndTime)`

QT Version 3.0 or later

See also GotoTime, StartTime

StartTime

Syntax `StartTime`

Description Returns the end time of the sample. This value is calculated only when the script is compiled. This means that you can not pass in a string value to get the start time of a sample while your script is running.

Example `GoToTime(SampleNamed(“My Sample”).StartTime)`

QT Version 3.0 or later

See also GotoTime, EndTime

SPATIAL TRACK PROPERTIES AND ACTIONS

Spatial Track properties and actions need to have a track and movie target specified. If no movie target is specified then the current movie is considered to be the target. You can specify a movie target by using MovieNamed(name) or MovieOfID(id) and pass in either the name of the movie or its ID. If no track target is specified then the track that contains the object that is currently executing the script is considered to be the target. You can specify a track target by using TrackNamed(name), TrackOfID(id), TrackOfIndex(index) or TrackOfType(type). See the section on targets for a complete description and some examples of how to specify a target.

Spatial Tracks are tracks that have a visual representation in the movie. Spatial Track Properties and Actions are available for all spatial track targets. The built-in spatial track types are as follows:

- Color Track
- Effect Track
- Flash Track
- Movie Track
- Picture Track
- Sprite Track
- Text Track
- Video Track
- VR Track

Spatial Track Properties

CanBeFocus

Syntax `CanBeFocus`

Description This property returns a boolean value indicating True if the target track can accept keyboard focus.

New for QuickTime 5, a number of QuickTime Track types can process the KeyPressed event. These tracks are the Flash, Text and Sprite tracks.

By setting the “Accept Focus” property for these tracks, the KeyPressed event will be triggered when the user types on the keyboard.

The *`CanBeFocus`* property returns the “Accept Focus” state of the target track.

Returns Boolean value indicating *`True`* if the “Accept Focus” property is set for the target track.

Example

```
If (CanBeFocus = true)

// Do something if the track can receive focus
Endif
```

QT Version 5.0 or later

See also IsFocus, SetFocus, EatKeyPressedEvent

IsFocus

Syntax `IsFocus`

Description This property returns a boolean value indicating True if the target track is the current keyboard focus.

New for QuickTime 5, a number of QuickTime Track types can process the KeyPressed event. These tracks are the Flash, Text and Sprite tracks.

The *`IsFocus`* property returns *`True`* if the target track has keyboard focus.

Returns Boolean value indicating *`True`* if the target track has the keyboard focus.

Example

```
If (IsFocus = true)
// Do something if the track has keyboard focus
Endif
```

QT Version 5.0 or later

See also CanBeFocus, SetFocus, EatKeyPressedEvent

MouseHorizontal

Syntax `MouseHorizontal`

Description This property returns the current horizontal coordinate of the mouse pointer in the coordinate space of the track. The returned value is in pixels. Each time the MouseHorizontal property is accessed, the actual current horizontal coordinate of the mouse pointer is returned. This means that the value returned can change while the script is executing. In order to ensure that the same MouseHorizontal return value is used throughout the script, first assign the value to a local variable before using it in calculations.

***Return**s* Numeric value indicating the horizontal coordinate of the mouse pointer in pixels.

Example

```
LocalVars curMouseX
curMouseX = MouseHorizontal
```

QT Version 3.0 or later

See also MouseVertical

MouseVertical

Syntax `MouseVertical`

Description This property returns the vertical coordinate of the mouse pointer in the coordinate space of the track. The returned value is in pixels. Each time the MouseVertical property is accessed, the actual current vertical coordinate of the mouse pointer is returned. This means that the value returned can change while the script is executing. In order to ensure that the same MouseVertical return value is used throughout the script, first assign the value to a local variable before using it in calculations.

***Return**s* Numeric value indicating the vertical coordinate of the mouse pointer in pixels.

Example

```
LocalVars curMouseY
curMouseY = MouseVertical
```

QT Version 3.0 or later

See also MouseHorizontal

TrackHeight

Syntax `TrackHeight`

Description This property returns a numeric value indicating the height in pixels of the track. If the track is a non-visual track then 0 is returned.

***Return**s* Numeric value indicating the height of the track in pixels.

Example

```
LocalVars value
value = TrackHeight
```

QT Version 3.0 or later

See also TrackWidth, ResetMatrix, ScaleMatrixBy

TrackLayer

Syntax `TrackLayer`

Description This property returns a numeric value indicating the visual layer of the track. Tracks with a higher layer number are drawn first. This means that tracks with a higher number appear behind tracks with a lower layer number.

Returns Numeric value indicating the visual layer of a track.

Example

```
LocalVars value
// Get the track layer of the current track
value = TrackLayer
// Get the track layer of the specified track
value = TrackOfID(4).TrackLayer
```

QT Version 3.0 or later

See also SetLayerTo, SetLayerBy

TrackWidth

Syntax `TrackWidth`

Description This property returns a numeric value indicating the width in pixels of the track. If the track is a non-visual track then 0 is returned.

Returns Numeric value indicating the width of the track in pixels.

Example

```
LocalVars value
value = TrackWidth
```

QT Version 3.0 or later

See also TrackHeight, ResetMatrix, ScaleMatrixBy

Spatial Track Actions

EatKeyPressEvent

Syntax `EatKeyEvent`

Description This action prevents the KeyPress event from being sent back to the application playing the movie.

New for QuickTime 5, a number of QuickTime Track types can process the KeyPressed event. These tracks are the Flash, Text and Sprite tracks.

This action is used with KeyPressed event handlers.

Example

```
EatKeyEvent
// Indicate that the key was handled
```

QT Version 5.0 or later

See also CanBeFocus, IsFocus, SetFocus

MoveMatrixBy

Syntax `MoveMatrixBy(X,Y)`

X,Y An integer constant that specifies the amount to move the matrix by in the x and y plane.

Description This action moves the track by the amount specified by the X and Y parameters.

Example `MoveMatrixBy(10,10)`

QT Version 4.0 or later

See also MoveMatrixBy, MoveMatrixTo, ResetMatrix, ScaleMatrixBy, SetMatrixTo, SetMatrixBy

MoveMatrixTo

Syntax `MoveMatrixTo(X,Y)`

X,Y An integer constant that specifies where to move the matrix to in the x and y plane.

Description This action moves the track to the position specified by the X and Y parameters. It will also reset all other matrix settings to normal, thus losing all scaling and rotation that may have been applied to the matrix.

Example `MoveMatrixTo(10,10)`

QT Version 4.0 or later

See also MoveMatrixBy, MoveMatrixTo, ResetMatrix, RotateMatrixBy, ScaleMatrixBy, SetMatrixTo, SetMatrixBy

ResetMatrix

Syntax `ResetMatrix`

Description This action resets the display matrix of the track which removes any Spatial transformations that have been applied to it. This is a useful way to start the matrix in a known state before applying rotations or other transformations to it.

Example `ResetMatrix`

QT Version 4.0 or later

See also MoveMatrixBy, MoveMatrixTo, RotateMatrixBy, ScaleMatrixBy, SetMatrixTo, SetMatrixBy

RotateMatrixBy

Syntax `RotateMatrixBy(angle, xloc, yloc)`

Angle A numeric literal or expression that specifies the amount to rotate the matrix by.

xloc, yloc A numeric literal or expression that specifies the center of rotation

Description This action rotates the track by the amount specified around the point specified by xloc, yloc.

Example `RotateMatrixBy(5, 0, 0)`

QT Version 5.0 or later

See also MoveMatrixBy, MoveMatrixTo, ResetMatrix, ScaleMatrixBy, SetMatrixTo, SetMatrixBy

ScaleMatrixBy

Syntax `ScaleMatrixBy(`*`X,Y`*`)`

`X,Y` A numeric constant that specifies the amount to scale the matrix by in the x and y plane.

Description This action scales the track by the amount specified by the X and Y parameters.

Example `ScaleMatrixBy(2,0.5)`

QT Version 4.0 or later

See also MoveMatrixBy, MoveMatrixTo, ResetMatrix, RotateMatrixBy, SetMatrixTo, SetMatrixBy

SetClipRegionTo

Syntax SetClipRegionTo(region)

region A region specification provided by using the #RegionFromImageFile or #RegionFromRect preprocessor directives.

Description This action sets the clipping region of the spatial track. The region parameter is supplied by the preprocessor directives `#RegionFromImageFile` or `#RegionFromRect`. For more information on these preprocessor directives, please refer to the section on Preprocessor Directives.

Example

```
ThisTrack.SetClipRegionTo(
#RegionFromRect(0, 0, 100, 100))
// Sets the clipping region to the rectangle
// specified
```

QT Version 3.0 or later

See also #RegionFromImageFile, #RegionFromRect

SetFocus

Syntax `SetFocus`

Description This action sets the keyboard focus to the target track.

New for QuickTime 5, a number of QuickTime Track types can process the KeyPressed event. These tracks are the Flash, Text and Sprite tracks.

Example

```
SetFocus
// Direct the keyboard focus to the this track
```

QT Version 5.0 or later

See also CanBeFocus, IsFocus, EatKeyPressedEvent

SetGraphicsModeBy

Syntax `SetGraphicsModeBy(`*`Mode, Red_Color, Green_Color, Blue_Color`*`)` <***MIN***(*number*) ***MAX***(*number*) ***wraparound***>

`Mode` A constant indicating one of the graphic modes to use.

`Red_Color` A numeric constant indicating the brightness of the red component.

*`Green_Color`*A numeric constant indicating the brightness of the green component.

`Blue_Color` A numeric constant indicating the brightness of the blue component.

Description This action changes the graphic mode the track uses to render itself, by the specified amount. The parameter Mode is a predefined constant indicating the graphic mode to set (see SetGraphicsModeBy or see Appendix II for a list of the graphic mode constants).

The parameters Red_Color, Green_Color and Blue_Color are numeric constants specifying the amount each primary colors is to change by. You can not actually change the mode by a relative amount, only its red, green and blue components.

Example The following statement changes the red, green and blue components of the graphic mode of the current track by 1000:

```
SetGraphicsModeBy(srcCopy, 1000, 1000, 1000)
```

QT Version 3.0 or later

See also Numeric Expression, SetGraphicsModeTo

SetGraphicsModeTo

Syntax `SetGraphicsModeTo(`*`Mode, Red_Color, Green_Color, Blue_Color`*`)` <***MIN***(*number*) ***MAX***(*number*) >

Mode A constant indicating one of the graphic modes to use.

Red_Color A numeric constant indicating the brightness of the red component

Green_Color A numeric constant indicating the brightness of the green component.

Blue_Color A numeric constant indicating the brightness of the blue component.

Description This action sets the graphic mode the track uses to render itself. See the list below or Appendix II—Drawing Mode Reference.

AVAILABLE DRAWING MODES

```
srcCopy
srcOr
srcXor
srcBic
notSrcCopy
notSrcOr
notSrcXor
notSrcBic
blend
addPin
addOver
```

```
subPin
transparent
addMax
subOver
adMin
grayishTextOr
hilite
ditherCopy
graphicsModeStraightAlpha
graphicsModePreWhiteAlpha
graphicsModePreBlackAlpha
graphicsModeComposition
graphicsModeStraightAlphaBlend
graphicsModePreMulColorAlpha
```

Example The following statement causes the graphic mode to be set to transparent using black as the transparent color:

```
SetGraphicModeTo(transparent, 0, 0, 0)
```

QT Version 3.0 or later

See also Numeric Expression, SetGraphicsModeBy

SetLayerBy

Syntax `SetLayerBy(`*`layer`*`)` <***MIN***(number) ***MAX***(number) ***wraparound***>

`layer` A numeric expression indicating the amount to change the layer by.

Description This action changes the visual layer of a track by the amount specified. Tracks with a higher number layer will draw first. This means that they will appear behind tracks with a lower layer number.

Example `SetLayerBy(1) Min(10) Max(20)`

QT Version 3.0 or later

See also Numeric Expression, SetLayerTo, TrackLayer

SetLayerTo

Syntax `SetLayerTo(`*`layer`*`)`<***MIN***(number) ***MAX***(number) >

`layer` A numeric expression indicating the layer.

Description This action sets the visual layer of a track. Tracks with a higher number layer will draw first. This means that they will appear behind tracks with a lower layer number.

Example `SetLayerTo(1)`

QT Version 3.0 or later

See also Numeric Expression,. SetLayerBy, TrackLayer

SetMatrixBy

Syntax `SetMatrixBy(m1, m2, m3, m4, m5, m6, m7, m8, m9)`

m1... m9 Numeric literals specifying the values for the individual cells of a 3x3 matrix

Description This action increments the matrix of the Spatial Track using the 9 parameters supplied.

The 9 parameters are arranged in a 3x3 matrix as follows:

COL.	**1**	**2**	**3**
ROW			
1	m1	m2	m3
2	m4	m5	m6
3	m7	m8	m9

QT Version 3.0 or later

See also MoveMatrixBy, MoveMatrixTo, ResetMatrix, RotateMatrixBy, ScaleMatrixBy, SetMatrixTo

SetMatrixTo

Syntax `SetMatrixTo(m1, m2, m3, m4, m5, m6, m7, m8, m9)`

m1... m9 Numeric literals specifying the values for the individual cells of a 3x3 matrix

Description This action sets the matrix of the Spatial Track using the 9 parameters supplied.

The 9 parameters are arranged in a 3x3 matrix as follows:

COL.	**1**	**2**	**3**
ROW			
1	m1	m2	m3
2	m4	m5	m6
3	m7	m8	m9

Example

```
SetMatrixTo(1, 0, 0,
            0, 1, 0,
            0, 0, 1)
```

QT Version 3.0 or later

See also MoveMatrixBy, MoveMatrixTo, ResetMatrix, RotateMatrixBy, ScaleMatrixBy, SetMatrixBy

FLASH TRACK PROPERTIES AND ACTIONS

Flash Track properties and actions need to have a track and movie target specified. If no movie target is specified then the current movie is considered to be the target. You can specify a movie target by using MovieNamed(name) or MovieOfID(id) and pass in either the name of the movie or its ID. If no track target is specified then the track that contains the object that is currently executing the script is considered to be the target. You can specify a track target by using TrackNamed(name), TrackOfID(id), TrackOfIndex(index) or TrackOfType(type). See the section on targets for a complete description and some examples of how to specify a target.

Flash Tracks are also Spatial Tracks as they have a visual representation within the movie. Thus the Spatial Track Actions and Properties are also available for Flash Track Targets (see the section on Targets).

Flash Track Properties

GetFlashVariable

Syntax `GetFlashVariable(path, name)`

path A string specifying the path to the variable.

name A string specifying the name of the variable.

Description This property returns specified Flash Track variable.
Use the path string of “” to specify the root path in the Flash Track.

Return The value of the specified variable in the Flash Track.

Example

```
MovieVars flashVar
flashVar = GetFlashVariable(“”, “MyVar”)
// Get the Flash variable “MyVar”
```

QT Version 5.0 or later

See also SetFlashVariable

Flash Track Actions

GoToFrameNamed

Syntax `GoToFrameNamed(name)`

Name A string literal indicating the name of the frame to go to.

Description This action goes to a labeled frame with the specified name in the Flash Track. This is done by setting the current time in the movie to the time of the named frame in the Flash Track.

Example

```
GoToFrameNamed(“Intro”)
```

QT Version 4.0 or later

See also GoToFrameNumber, MovieTime

GoToFrameNumber

Syntax `GoToFrameNumber(frame)`

Frame An integer expression indicating the frame number to go to.

Description This action goes to the frame with the specified frame number in the Flash Track. This is done by setting the current time in the movie to the time of the indicated frame in the Flash Track.

Flash frames are numbered starting with 0 so to go to the 10th frame, you must specify the number 9.

Example `GoToFrameNumber(12)`
This example goes to the 13th frame in the Flash Track.

QT Version 4.0 or later

See also GoToFrameNamed, MovieTime

PerformFlashClick

Syntax `PerformFlashClick(`*`path, buttonID, transition`*`)`

`path` A string literal or variable specifying the path to the button

`buttonID` A numeric literal or variable specifying the ID of the button.

`transition` A numeric literal or variable specifying the mouse transition type.

Description This action simulates a mouse click within the Flash Track. A simulated mouse click is sent to the specified button.

`Transition` specifies the mouse transition type to send with the simulated mouse click:

MOUSE TRANSITION DEFINES	VALUE
kIdleToOverUp	0
kOverUpToIdle	1
kOverUpToOverDown	2
kOverDownToOverUp	3
kOverDownToOutDown	4
kOutDownToOverDown	5
kOutDownToIdle	6
kIdleToOverDown	7
kOverDownToIdle	8

Use the path string of “”to specify the root path in the Flash Track.

Example

```
PerformFlashClick(“”,100, 128)
// Simulate a button click on button ID 100
// in the Flash track
```

QT Version 5.0 or later

See also GetFlashVariable, SetFlashVariable

SetFlashVariable

Syntax `SetFlashVariable(`*`path, name, value, focus`*`)`

`path` A string literal or variable specifying the path to the variable.

`name` A string literal or variable specifying the name of the variable.

`value` A string literal or variable specifying the value to set the Flash Variable to.

`focus` A boolean literal or variable specifying if the focus is to be changed.

Description This action sets the specified Flash Track variable to the value in the parameters.

Use the path string of “” to specify the root path in the Flash Track.

Example

```
SetFlashVariable("", "MyVar", "1234", FALSE)
// Set the Flash variable "MyVar" to "1234"
```

QT Version 5.0 or later

See also GetFlashVariable

SetPan

Syntax `SetPan(X_Percent, Y_Percent)`

X_Percent An integer constant specifying the percent to pan the Flash Track along the horizontal plane.

Y_Percent An integer constant specifying the percent to pan the Flash Track along the vertical plane.

Description This action pans the Flash Track by the specified x and y percentages.

Example `SetPan(25, 45)`

QT Version 4.0 or later

See also SetZoom

SetZoom

Syntax `SetZoom(Zoom_Factor)`

Zoom_Factor An integer constant specifying the amount to zoom the Flash Track by.

Description This action zooms the Flash Track by the zoom factor.

Example `SetZoom(10)`

QT Version 4.0 or later

See also SetPan

SetZoomRect

Syntax `SetZoomRect(Left, Top, Right, Bottom)`

Left An integer constant specifying the left edge of the rectangle to zoom to.

Top An integer constant specifying the top edge of the rectangle to zoom to.

Right An integer constant specifying the right edge of the rectangle to zoom to.

Bottom An integer constant specifying the bottom edge of the rectangle to zoom to.

Description This action zooms the Flash Track to the specified rectangle.

Example `SetZoomRect(10, 10, 110, 100)`

This example will set a display area in the track that is offset 10 pixels in on the X and Y axis from the top left corner and is 100 pixels high and wide.

QT Version 4.0 or later

See also SetZoom

MOVIE TRACK ACTIONS

Movie Track properties and actions need to have a track and movie target specified. If no movie target is specified then the current movie is considered to be the target. You can specify a movie target by using MovieNamed(name) or MovieOfID(id) and pass in either the name of the movie or its ID. If no track target is specified then the track that contains the object that is currently executing the script is considered to be the target. You can specify a track target by using TrackNamed(name), TrackOfID(id), TrackOfIndex(index) or TrackOfType(type). See the section on targets for a complete description and some examples of how to specify a target.

Movie Tracks are also Spatial Tracks as they have a visual representation within the movie. Thus the Spatial Track Actions and Properties are also available for Movie Track Targets (see the section on Targets).

Movie Track Actions

AddChildMovie

Syntax `AddChildMovie(`*`id, URL`*`)`

`id` A numeric expression indicating the ID for this new child movie.

`URL` A string or string variable containing the URL for the child movie.

Description This action adds a new child movie to the child movie list for the movie track. This does not load the movie or start it downloading. You only need to do this once for any particular URL. If you are not creating the URL dynamically in your script, then you should add the URL to the movie list when you create the movie track.

Example `AddChildMovie(1000, “www.xcom.com/test.mov”)`

QT Version 4.1 or later

See also LoadChildMovie

LoadChildMovie

Syntax `LoadChildMovie(`*`id`*`)`

`id` The ID is a numeric expression and must match an ID of a previously loaded child movie, or one added to the movie list when the movie track was created.

Description This action causes the child movie that matches the ID to start loading. The movie will display and optionally start playing when enough of it has downloaded. You should not immediately start the movie playing since it will not have started downloading. You should use MaxLoadedTimeInMovie to determine when some of the movie has downloaded before starting it playing. Keep in mind that only one movie is loaded at a time. Once you load a movie, then the previously loaded movie is released and discarded.

Example `LoadChildMovie(1000)`

QT Version 4.1 or later

See also AddChildMovie

LoadChildMovieWithQTList

Syntax `LoadChildMovieWithQTList(id, XML)`

`id` A numeric expression indicating the ID of the child movie.

`XML` A string literal or variable containing an XML formated string specifying the new elements that are to be loaded.

Description This action loads the supplied *`XML`* string into the QTList of the specified child movie in the target movie track.

Example

```
LocalVars myXML
// Load myXML with XML formated text
TrackNamed(“Movies”).LoadChildMovieWithQTLIst(1, myXML)
```

QT Version 5.0 or later

See also QTList

RestartAtTime

Syntax `RestartAtTime(time, rate)`

`time` A numeric literal indicating the in the child movie to start playing from.

`rate` A numeric literal or expression for the playback rate.

Description This action plays the target child movie starting at the specified time and rate.

Example

```
// Play the child movie from the beginning
ChildMovieNamed(“MyChild”).RestartAtTime(0, 1.0)
```

QT Version 4.1 or later

MUSIC TRACK PROPERTIES AND ACTIONS

Music Track properties and actions need to have a track and movie target specified. If no movie target is specified then the current movie is considered to be the target. You can specify a movie target by using MovieNamed(name) or MovieOfID(id) and pass in either the name of the movie or its ID. If no track target is specified then the track that contains the object that is currently executing the script is considered to be the target. You can specify a track target by using TrackNamed(name), TrackOfID(id), TrackOfIndex(index) or TrackOfType(type). See the section on targets for a complete description and some examples of how to specify a target.

Track Actions and Properties are also available for Music Track targets (see the section on Targets).

Music Track Actions

PlayNote

Syntax `PlayNote(Instrument, Delay, Pitch, Velocity, Duration)`

`Instrument` The index of the instrument in the first sample of the target instrument track.

Delay The Delay parameter specifies a delay before the note starts playing. This value is measured in the movies Time Scale. A standard movie has a time scale of 600 so specifying 600 for your delay will create a 1 second delay.

Pitch The Pitch parameter specifies the note’s frequency. Values ranging from 0 to 127 are accepted, a value of 60 will produce a middle C, 59 is a middle B, etc.

Velocity The Velocity parameter specifies the volume the note is to be played at. Values for this parameter range from 0 (no sound) to 100 (full volume).

Duration The Duration parameter specifies the length of time the note will play for. This parameter operates like the Delay parameter. It is measured in the movies Time Scale. Note that instruments that naturally decay may not be obviously affected by this value.

Description This action plays a note using the specified Music Instrument. The parameters Instrument, Delay, Pitch, Velocity and Duration are all Numeric Expressions specifying how the note is to be played.

Musical instruments are added to a LiveStage Professional project through the Instrument Media Sample window. To add an instrument to the LiveStage Professional project, click on the “Add Built-in” button at the bottom of the Instrument Media Sample window. An instrument selection dialog will be displayed allowing you to selection a MIDI style instrument. You may also add digital audio clips as instruments by dragging them into the instruments list in the Instrument Media Sample window.

Example The following statement causes the note Middle C to be played using the first musical instrument in the track named “Instruments 1” at half volume.

```
TrackNamed(“Instruments 1”).PlayNote(1, 0, 60, 100, 1000)
```

QT Version 3.0 or later

See also Numeric Expression, SetController

SetController

Syntax `SetController(`*`Sample, Instrument, Delay, Controller, Value`*`)`

Sample The index of the sample of the target instrument track that contains the instrument to be altered.

Instrument The index of the instrument in the sample of the target instrument track that is to be altered.

Delay The Delay parameter specifies a delay before the SetController action takes place. This value is measured in the movies Time Scale. A standard movie has a time scale of 600 so specifying 600 for your delay will create a 1 second delay.

Controller A numeric constant indicating the controller to modify.

Value A numeric expression. This value has a different meaning for each controller.

Description Controllers are used to modify the playback of an instrument. They can adjust characteristics such as Pitch, Blend, Pan, Reverb, etc. The setting of a controller is performed by indicating the instrument to adjust, the controller you want to modify and the value to set that controller to.

Controller values control items such as pitch, blend and reverb. The controller numbers used by QuickTime are mostly identical to the standard MIDI controller numbers.   These are signed 8.8 values. The full range, therefore is -128.00 to 127 + 127 / 128 (or 0x8000 to 0x7FFF). All controls default to zero except for volume and pan. Pitch bend is specified in fractional semitones, which eliminates the restrictions of a pitch bend range. It can be bent as much as desired, and whenever it is needed. The last 16 controllers (113-128) are global controllers. Global controllers respond when the part number is given as 0, indicating the entire synthesizer.

LIST OF CONTROLLER DEFINITIONS

```
kControllerModulationWheel
kControllerBreath
kControllerFoot
kControllerPortamentoTime
kControllerVolume
kControllerBalance
kControllerPan
kControllerExpression
kControllerLever1
kControllerLever2
kControllerLever3
kControllerLever4
kControllerLever5
kControllerLever6
kControllerLever7
kControllerLever8
kControllerPitchBend
kControllerAfterTouch
kControllerPartTranspose
kControllerTuneTranspose
kControllerPartVolume
kControllerTuneVolume
kControllerSustain
kControllerPortamento
kControllerSostenuto
kControllerSoftPedal
kControllerReverb
kControllerTremolo
kControllerChorus
kControllerCeleste
kControllerPhaser
kControllerEditPart
kControllerMasterTune
kControllerMasterTranspose
kControllerMasterVolume
kControllerMasterCPULoad
kControllerMasterPolyphony
kControllerMasterFeatures
```

QT Version 3.0 or later

See also Numeric Expression, PlayNote

QD3D OBJECT ACTIONS

QD3D support in QuickTime is obsolete and may not be included in future releases of QuickTime subsequent to version 5.0. MacOS X does not support QD3D objects.

QD3D Object Actions

RotateTo

Syntax `RotateTo(X, Y, Z)`

X,Y,Z Numeric constants that specify the angles for each of the three dimensions.

Description This action will rotate the object to a position in 3D space.

Example `RotateTo(10.6, 0, 100)`

QT Version 4.0 or later

See also TranslateTo, ScaleTo

ScaleTo

Syntax `ScaleTo(X, Y, Z)`

X,Y,Z Numeric constants that specify the scale factors for each of the three dimensions.

Description This action scales the object in 3D space by the amount specified in X,Y,Z.

Example `ScaleTo(0.5, 1, 1)`
This example will scale the 3D object 50% on its width and maintain full size in the Y and Z axis.

QT Version 4.0 or later

See also TranslateTo, RotateTo

TranslateTo

Syntax `TranslateTo(X, Y, Z)`

X,Y,Z Numeric expression specifying the 3D position of the object.

Description This action moves the object to the specified location in 3D space.

Example `TranslateTo(100, 100, 5)`
This example will move the object to an X position of 100, Y position of 100 and the Z position of 5.

QT Version 4.0 or later

See also ScaleTo, RotateTo

SOUND TRACK PROPERTIES AND ACTIONS

Sound Track properties and actions need to have a track and movie target specified. If no movie target is specified then the current movie is considered to be the target. You can specify a movie target by using MovieNamed(name) or MovieOfID(id) and pass in either the name of the movie or its ID. If no track target is specified then the track that contains the object that is currently executing the script is considered to be the target. You can specify a track target by using TrackNamed(name), TrackOfID(id), TrackOfIndex(index) or TrackOfType(type). See the section on targets for a complete description and some examples of how to specify a target.

Track Actions are also available for Sound Track targets (see the section on Targets).

Sound Track Properties

GetBass

Syntax `GetBass`

Description This property returns the current Bass setting of the target sound track.

***Return*s** Numeric value indicating the Bass setting of the sound track.

Example

```
LocalVars theBass
theBass = GetBass
```

QT Version 5.0 or later

See also SetBassTreble

GetTreble

Syntax `GetTreble`

Description This property returns current Treble setting of the target sound track.

***Return*s** Numeric value indicating the Treble setting of the sound track.

Example

```
LocalVars theTreble
theTreble = GetTreble
```

QT Version 5.0 or later

See also SetBassTreble

TrackBalance

Syntax `TrackBalance`

Description This property returns the balance setting of the track. The value returned is between -128 and 128. The value of -128 indicates that the balance is set to the left channel only while a value of 128 indicates right channel only. A value of 0 indicates center balance.

***Return*s** Numeric value indicating the balance setting of the track.

Example

```
LocalVars theBalance
theBalance = TrackBalance
```

QT Version 3.0 or later

See also SetBalanceTo, SetBalanceBy

TrackVolume

Syntax `TrackVolume`

Description This property returns the volume level of the track. The value returned is between 0 (silence) and 255 (full volume). This is different from MovieVolume since TrackVolume returns only the volume of the specified track.

***Return**s* Numeric value indicating the volume level of the track.

Example

```
LocalVars theVolume
theVolume = TrackVolume
```

QT Version 3.0 or later

See also MovieVolume, SetVolumeTo, SetVolumeBy

Sound Track Actions

SetBalanceBy

Syntax `SetBalanceBy(`*`balance`*`)` <***MIN***(*number*) ***MAX***(*number*) ***wraparound***

`balance` A numeric expression from -256 (left) to 256 (right) indicating the change in sound balance from left to right for the track.

Description This action alters the left to right balance of a track by the specified amount. The parameter balance supplies the amount the balance is to be altered by. A negative value will shift the balance to the left while a positive value will shift the balance to the right.

Example The following statement causes the balance of the current track to shift to the left by 10%.

```
SetBalanceBy(-25)
```

QT Version 3.0 or later

See also Numeric Expression, SetBalanceTo, TrackBalance

SetBalanceTo

Syntax `SetBalanceTo(`*`balance`*`)`

`balance` A numeric expression from -128 (left) to 128 (right) indicating the sound balance from left to right for the track.

Description This action sets the sound balance for the track. The balance can change from the left speaker at full volume and the right speaker off all the way to the left speaker off and the right speaker at full volume.

Example The following statement sets the left to right balance of the current track to the center.

```
SetBalanceTo(0)
```

QT Version 3.0 or later

See also Numeric Expression, SetBalanceBy, TrackBalance

SetBassTrebleBy

Syntax `SetBassTrebleBy(`*`bass, treble`*`)` <***MIN***(*number*) ***MAX***(*number*) ***wraparound***>

`bass` A numeric expression from -128 to 128 indicating the amount to change the bass setting for the sound track by.

`treble` A numeric expression from -128 to 128 indicating the amount to change the treble setting for the sound track by.

Description This action sets the Bass and Treble for the sound track.

Example The following statement sets the Bass of the sound track sample by 10.

```
TrackNamed(“MySound”).SetBassTrebleBy(10, 0)
// Increase the bass setting by 10
```

QT Version 5.0 or later

See also GetBass, GetTreble, SetBassTrebleTo

SetBassTrebleTo

Syntax `SetBassTrebleTo(`*`bass, treble`*`)`

`bass` A numeric expression from -128 to 128 indicating the amount to change the bass setting for the sound track by.

`treble` A numeric expression from -128 to 128 indicating the amount to change the treble setting for the sound track by.

Description This action sets the Bass and Treble for the sound track.

Example The following statement sets the Bass and Treble of the sound track to neutral.

```
TrackNamed(“MySound”).SetBassTrebleTo(0, 0)
```

QT Version 5.0 or later

See also GetBass, GetTreble, SetBassTrebleBy

SetVolumeBy

Syntax `SetVolumeBy(`*`volume`*`)` <***MIN***(*number*) ***MAX***(*number*) ***wraparound***>

`Volume` A numeric expression from -256 to 256 indicating the change in the volume setting.

Description This action alters the volume level of the track by the specified amount. The parameter Volume supplies the value by which the volume level is changed up (positive value) or down (negative value).

Example The following statement causes the volume level of the current track to increase by 25:

```
SetVolumeBy(25)
```

QT Version 3.0 or later

See also Numeric Expression, SetVolumeTo, TrackVolume

SetVolumeTo

Syntax `SetVolumeTo(volume)` <***MIN***(*number*) ***MAX***(*number*)>

Volume A numeric expression from -256 (off) to 256 (full volume) indicating the new volume setting.

Description This action sets the volume level of the track. The volume level of each track can be adjusted separately. The volume level of the movie controls the overall volume setting. A setting of 0 or less will mute the sound from the track. You can use negative volume settings to keep track of the previous volume setting.

Example The following statement causes the volume of the current track to be turned off:

```
SetVolumeTo(0)
```

QT Version 3.0 or later

See also Numeric Expression, SetVolumeBy, TrackVolume

SPRITE TRACK PROPERTIES AND ACTIONS

Sprite Track properties and actions need to have a track and movie target specified. If no movie target is specified then the current movie is considered to be the target. You can specify a movie target by using MovieNamed(name) or MovieOfID(id) and pass in either the name of the movie or its ID. If no track target is specified then the track that contains the object that is currently executing the script is considered to be the target. You can specify a track target by using TrackNamed(name), TrackOfID(id), TrackOfIndex(index) or TrackOfType(type). See the section on targets for a complete description and some examples of how to specify a target.

Sprite Tracks are also Spatial Tracks as they have a visual representation within the movie. Thus the Spatial Track Actions and Properties are also available for Sprite Track Targets (see the section on Targets).

Sprite Track Properties

GetIdleFrequency

Syntax `GetIdleFrequency`

Description This property returns the idle frequency in movie time in 1/60 second increments.

Return Numeric value indicating the idle frequency.

Example

```
LocalVars idleFreq
idleFreq = GetIdleFrequency
// Get the current track's idle frequency
```

QT Version 5.0 or later

See also SetIdleFrequency

NumSprites

Syntax `NumSprites`

Description This property returns the number of sprites in a sprite track. If the track specified is not a sprite track then 0 is returned.

Return Numeric value indicating the number of sprites in the specified sprite track.

Example

```
IF (NumSprites > 100)
// Script executes only when the number of sprites is
// greater than 100
EndIf
```

QT Version 3.0 or later

See also MakeNewSprite, NewImages, SpriteOfIndex

NumImages

Syntax `NumImages`

Description This property returns the number of images in a sprite track. If the track specified is not a sprite track then 0 is returned. Images are used by sprites to render their visual appearance.

Return Numeric value indicating the number of sprites in the specified sprite track.

Example

```
// Set the image of the current sprite to the
// last image in the sprite track
SetImageIndexTo(NumImages)
```

QT Version 3.0 or later

See also SetImageIndexTo, SetImageIndexBy, NumSprites

SpriteAtPoint

Syntax `SpriteAtPoint(`*`x, y`*`)`

Description This property returns the ID of the sprite that is at the point specified in the Sprite Track. The ID of the first sprite that intersects this point (going from the topmost to the bottommost sprite) is returned.

Return Numeric value indicating the ID the sprite that corresponds to the supplied X,Y location.

QT Version 5.0 or later

Sprite Track Actions

MakeNewSprite

Syntax `MakeNewSprite(`*`Sprite_ID, Handler_ID, Image_Index, Visible, Layer`*`)`

Sprite_ID An integer expression specifying the ID of the new sprite.

Handler_ID An integer expression specifying the ID of the sprite that will be used to handle events for the new sprite.

Image_Index An integer expression specifying the initial image index for the new sprite.

Visible A boolean expression specifying the visible state of the new sprite.

Layer An integer expression specifying the drawing layer for the new sprite.

Description This action creates a new sprite with the given Sprite_ID.

The Handler_ID parameter indicates the sprite id of the sprite that will contain the handlers for this sprite. Note that you cannot use a sprite that was created with MakeNewSprite as the handler sprite. When the new sprite needs to handle an event, such as a Mouse Click event, the sprite with the ID of Handler_ID will have its handler called to handle the event. You can use the property CustomHandlerID to get the ID of the new sprite that was actually clicked on.

Example

```
MakeNewSprite(100, 1, 3, TRUE, 1)
```

The above example will create a single sprite with the ID of 100 which uses sprite at ID 1 as its handler. The sprite will use the image at index 3, is created visible and will be placed on layer 1.

QT Version 4.0 or later

See also DisposeSprite, CustomHandlerID

DisposeSprite

Syntax `DisposeSprite(Sprite_ID)`

Sprite_ID An integer expression specifying the ID of the sprite to dispose of.

Description This action will dispose of the sprite that has the specified ID. The sprite must have been previously created with MakeNewSprite.

Example

```
DisposeSprite(100)
```

The above example will dispose of the sprite that has ID 100.

QT Version 4.0 or later

See also MakeNewSprite

SetIdleFrequency

Syntax `SetIdleFrequency(frequency)`

frequency An integer expression indicating the frequency of the idle events in 1/60 second increments.

Description This action will change the idle frequency of the sprite track. Idle Frequency determines how often the Idle Event handlers are called by QuickTime.

Example

```
SetIdleFrequency(1 * 60)
// Set the idle frequency of the track to once
// every second
```

QT Version 5.0 or later

QTLIST PROPERTIES AND ACTIONS

A QTList is a flexible data storage mechanism that is attached to QuickTime Movies and Tracks.

QTList Actions and Properties must be explicitly targeted when working with QTLists attached to Tracks. The following targeting must be used when targeting Tracks:

```
TrackNamed
TrackOfIndex
TrackOfType
```

QTList Actions and Properties without an explicit target will default to refer to the QTList attached to the movie.

Please note that the target specifier ThisTrack does nothing when used with QTList Actions and Properties and will always target the QTList of the movie.

QTLists can be used to store a wide variety of hierarchical information with a List Enabled QuickTime Object. The data storage metaphor is similar to that of an XML document. In fact, you can load all or part of a List using XML documents.

Elements in a QTList are addressed using a path string. The format of the path string is as follows: *path_parent.path_child1.path_child2.path_child...*

Each part of the path string is separated by a "." character (similar to the Property and Action access syntax). Each part of the path string specifies a branch in the hierarchical structure of the list.

For example, to access a particular date in a calendar, you would use the following path:

```
year.2001.month.february.day.16
```

The above path string would be used to address a QTList Element for February 16, 2001. The List hierarchy for such a calendar list would look like this in XML format:

```
<year>
        <2001>
                <month>
                        <january>
                        </january>
                        <february>
                          <day>
                                <16>Some Data</16>
                        </day>
                        </february>
                ...
                </month>
        </2001>
        ...
</year>
```

The following is an example of how to create a List Element for February 14, 2001. In the following examples, we will assume that the track we are using has the index of 1:

```
TrackOfIndex(1).AddListElement("year.2001.month.february.day", 1, "14")
```

In the above example, the path string of "year.2001.month.february.day" specified the branch within the hierarchy of the QTList where the new element is to be added. The 1 specified as the index indicates that the new element will be the first element in this part of the hierarchy of the QTList. In the example hierarchy, the index for the 16th would have been 1 prior to the new element being added. This index changes to 2 after the element insertion.

Now that the new element has been added, you can now assign it a value:

```
TrackOfIndex(1).SetListElement("year.2001.month.february.day.14",
            "Valentine's Day")
```

Executing the above line of script will set the element corresponding to February 14, 2001 to "Valentine's Day".

To retrieve the value of this element, use the following script:

```
LocalVars elementValue
elementValue = TrackOfIndex(1).GetListElementValue(
            "year.2001.month.february.day.14")
```

When the element is no longer needed, you can delete it using the following script:

```
elementValue = TrackOfIndex(1).RemoveListElements(
            "year.2001.month.february.day", 1, 1)
```

QTList Elements can also be accessed via their index number. For instance, in the above example, To access the "February" element you can use the following path:

```
            "year.2001.month[2]"
```

The "[2]" in the path designates the second element in the container element "month". You can thus iterate through the elements of the container element with a "FOR" loop utilizing GetListElementCount to get the number of elements in the container path (in this case "year.2001.month"). By assembling a path string that is appended with "[x]" (where x is the index), you can step through each element.

QTLists can only be attached to List Enabled QuickTime Objects such as Movies and Tracks. QTList properties and actions must be implicitly targeted as they do not make use of default targets.

QTLists can be preloaded into the Movie and Tracks by utilizing the UI provided in the Info Tab of the Document Window and in the Advanced Tab of the Track Properties Dialogs.

QTList Properties

GetListAsXML

Syntax `GetListAsXML(`*`path, start, end`*`)`

`path` A string literal or variable specifying the list path to the element

`start` A numeric literal or variable specifying the first index

`end` A numeric literal or variable specifying the last index

Description This property returns an XML formatted string containing the List elements specified by the *`path`* starting at index *`start`* and ending with index *`end`*.

The start and end index numbers must be within the valid range of elements. Specifying an invalid index will cause an error.

Return An XML formatted string containing the list elements specified.

Example

```
LocalVars xmlString
LocalVars numElmt
numElmt = TrackOfIndex(1).GetListElementCount(
          “year.2001.month.february.day”)
xmlString = TrackOfIndex(1).GetListAsXML(
          “year.2001. month.february.day”, 1, numElemt)
// Get the XML string for all days in February
```

QT Version 5.0 or later

GetListElementCount

Syntax `GetListElementCount(`*`path`*`)`

`path` A string literal or variable specifying the list path to the element.

Description This property returns the number of elements in the List Element specified by the *`path`*.

Return A numeric value indicating the number of elements in the specified List Element.

Example

```
LocalVars numElmt
numElmt = TrackOfIndex(1).GetListElementCount(
          “year.2001.month.february.day”)
// Get the number of days defined in February
```

QT Version 5.0 or later

GetListElementName

Syntax `GetListElementName(`*`path, index`*`)`

`path` A string literal or variable specifying the list path to the element.

`index` A numeric literal or variable specifying the index within the specified element.

Description This property returns the name of element specified by the *`path`* and *`index`* parameters.

Return A string value containing the name of the specified list element.

Example

```
LocalVars elmtName
SetString(elmtName, TrackOfIndex(1).GetListElementName(
          “year.2001.month.february.day”, 1)
// Get the name of the first day in February
```

QT Version 5.0 or later

GetListElementValue

Syntax `GetListElementValue(`*`path`*`)`

`path` A string literal or variable specifying the list path to the element.

Description This property returns the value of the QTList Element specified by the *`path`*.

Return A string value indicating the value stored in the specified QTList Element.

Example

```
LocalVars elmtValue
```

```
SetString(elmtValue,
TrackOfIndex(1).GetListElementValue(
         “year.2001.      month.february.day.14”))
// Get the value stored in February 14, 2001
```

QT Version 5.0 or later

QTList Actions

AddListElement

Syntax `AddListElement(path, index, name)`

path A string literal or variable indicating the list path to the parent element for the new element.

index A numeric literal or variable specifying the index of the new element to be added.

name A string literal or variable containing the name of the new element.

Description This action adds a new list element at the specified location. If the element addition is an insertion, all subsequent elements with the same parent element are renumbered.

QT Version 5.0 or later

ExchangeList

Syntax `ExchangeList(URL, path)`

URL A string literal or variable indicating the URL to send the list to.

path A string literal or variable containing the list path to the parent element.

Description This action calls the specified *URL* and sends the list elements referred to by the parameter *path* as an XML formatted parameter string. For example:

```
ExchangeList(“www.myurl.com/somecgi.cgi”, “”)
```

The above line of script will send the entire list converted to XML format to the specified URL as a parameter:

```
www.myurl.com/somecgi.cgi
?qtlist=<qtlist> ... XML list content ... </qtlist>
```

Upon return, a List Received event is triggered for the track which sent the ExchangeList action so that the received list can be processed.

When the List Received event is triggered by a retur*n*ned list being sent by the server, the returned list is saved in the temporary list of the event handler. To target the event’s local list, you must use the `ThisEvent` target specifier.

The format or the local list of the List Received event is as follows:

```
<event>
         <listName>Name of the list</listName>
         <list>
                  Contents of the returned list
                           ...
         </list>
</event>
```

ExchangeList is an asynchronous action. This means that the result of the ExchangeList action will not be available immediately following the execution of the action. The completion of the ExchangeList action is signaled by the triggering of the List Received event in the target track.

Example

```
// List Received Event Handler
MovieVars myList
SetString(myList, ThisEvent.GetListAsXML(“event.list”, 1,
ThisEvent.GetListElementCount(“event.list”))
// Retrieve the returned list as an XML string
```

QT Version 5.0 or later

See also QueryListServer

LoadListFromXML

Syntax `LoadListFromXML(path, start, XML)`

path A string literal or variable indicating the list path to the parent element where the new elements from the XML string is to be loaded.

start A numeric literal or variable specifying the index where the new elements from the XML string is to be loaded.

XML A string literal or variable containing an XML formatted string specifying the new elements that are to be loaded.

Description This action pastes the elements contained in the specified XML formatted string into the list. The *path and start* parameters specify the location where the new elements are to be inserted.

QT Version 5.0 or later

See also SetListFromURL

QueryListServer

Syntax `QueryListServer(URL, keyValuePairs, flags, path)`

URL A string literal or variable indicating the URL to send the list to.

keyValuePairs A string literal or variable containing the key/value pairs to be sent as parameters to the *URL*.

flags A numeric expression containing the flags used to control the action taken.

path A string literal or variable containing the list path to the parent element that will be sent as an XML parameter to the *URL*.

Description This action calls the specified *URL* and sends the list elements referred to by the parameter *path* as an XML formatted parameter string. For example:

```
QueryListServer(“www.myurl.com/somecgi.cgi”,
        “command=some_command”,
        kListQuerySendListAsXML +
        kListQuerySendListAsNameValuePairs, “”)
```

The above line of script will send the *keyValuePairs* string as well as the entire list converted to XML format to the specified URL as parameters. Here is what the above action would generate as a complete URL from the supplied parameters:

```
www.myurl.com/somecgi.cgi?command=some_command
&qtlist=<qtlist> ... XML list content ... </qtlist>
```

The available flags are as follows:

FLAG	MEANING
`kListQuerySendListAsXML`	Send the list specified by *path* as a parameter
`kListQuerySendList` *AsKeyValuePairs*	Send the string specified by *keyValue Pairs* as parameters
`kListQueryWantCallback`	Specify that a List Received event is to be triggered by returning data
`kListQueryDebugTrace`	Send the complete URL generated as a DebugStr to the application running the movie.

These flags can be used together simply by adding them together (e.g. `kListQuerySendListAsXML + kListQueryWantCallback` will cause the supplied list to be sent along with the URL. In addition, a List Received event will also be received).

Also note neither keyValuePairs nor path are required. Omitting the flags `kListQuerySendListAsXML` and `kListQuerySendListAsKeyValuePairs` will simply call the specified URL with no parameters.

Upon return, a List Received event is triggered for the track which sent the QueryListServer action so that the received list can be processed.

When the List Received event is triggered by a returned list being sent by the server, the returned list is saved in the temporary list of the event handler. To target the event's local list, you must use the `ThisEvent` target specifier.

The format of the local list of the List Received event is as follows:

```
<event>
        <listName>Name of the list</listName>
        <list>
                Contents of the returned list
                        ...
        </list>
</event>
```

QueryListServer is an asynchronous action. This means that the result of the QueryListServer action will not be available immediately following the execution of the action. The completion of the QueryListServer action is signaled by the triggering of the List Received event in the target track (if one is requested by using the kListQueryWantCallback flag).

QT Version 5.0 or later

See also ExchangeList

RemoveListElement

Syntax `RemoveListElement(path, start, end)`

`path` A string literal or variable indicating the list path to the parent element where the target elements are located

`start` A numeric literal or variable specifying the index of the first element to be removed.

`end` A numeric literal or variable specifying the index of the last element to be removed.

Description This action removes the specified elements from the specified list path.

QT Version 5.0 or later

ReplaceListFromXML

Syntax `ReplaceListFromXML(path, XML)`

`path` A string literal or variable indicating the list path to the parent element where the replacement operation is to take place.

`XML` A string literal or variable containing an XML formatted string containing element replacements.

Description This action replaces the elements within the specified parent element (specified by path) with element names that match those in the specified XML formatted string.

QT Version 5.0 or later

SetListElement

Syntax `SetListElement(path, value)`

`path` A string literal or variable indicating the list path to the element.

`value` A string literal or variable containing the new value of the element.

Description This action sets the value of the specified QTList element.

QT Version 5.0 or later

SetListFromURL

Syntax `SetListFromURL(URL, path)`

`URL` A string literal or variable containing the URL where the XML file is located.

`path` A string literal or variable indicating the list path to the parent element where the new elements from the XML file are to be loaded.

Description This action loads the elements contained in the specified XML file into the list. The path parameters specify the location where the new elements are to be inserted.

SetListFromURL is a synchronous action. This means that the action will complete its operation before proceeding to the next command in the script (unlike asynchronous actions like ExchangeList and QueryListServer).

Example

```
SetListFromURL("www.totallyhip.com/my.xml", "")
// Load the XML file "my.xml" to the root
// of the movie's QTList
```

QT Version 5.0 or later

See also LoadListFromXML

SPRITE PROPERTIES AND ACTIONS

Sprite properties and actions need to have a sprite, track and movie target specified. If no movie target is specified then the current movie is considered to be the target. You can specify a movie target by using MovieNamed(name) or MovieOfID(id) and pass in either the name of the movie or its ID. If no track target is specified then the track that contains the object that is currently executing the script is considered to be the target. You can specify a track target by using TrackNamed(name), TrackOfID(id), TrackOfIndex(index) or TrackOfType(type). If no sprite target is specified then the sprite executing the script is considered to be the target. You can specify a sprite target by using SpriteNamed(name), SpriteOfID(ID) or SpriteOfIndex(index). See the section on targets for a complete description and some examples of how to specify a target.

Track and Spatial Track Properties and Actions are also available for Sprite Track targets (see the section on Targets).

Sprite Properties

BoundsBottom

Syntax `BoundsBottom`

Description As a sprite is moved, rotated, stretched and skewed, the bounding rectangle of the sprite changes. BoundsBottom returns the bottom of that bounding rectangle.

Return Numeric value indicating the bottom edge of the sprite.

Example `If (BoundsBottom > 100)`

QT Version 3.0 or later

See also MoveTo, MoveBy, BoundsLeft, BoundsRight, BoundsTop

BoundsLeft

Syntax `BoundsLeft`

Description As a sprite is moved, rotated, stretched and skewed, the bounding rectangle of the sprite changes. BoundsLeft returns the left edge of that bounding rectangle.

Return Numeric value indicating the left edge of the sprite.

Example `If (BoundsLeft > 100)`

QT Version 3.0 or later

See also MoveTo, MoveBy, BoundsRight, BoundsTop, BoundsBottom

BoundsRight

Syntax `BoundsRight`

Description As a sprite is moved, rotated, stretched and skewed, the bounding rectangle of the sprite changes. BoundsRight returns the right edge of that bounding rectangle.

Return Numeric value indicating the right edge of the sprite.

Example `If (BoundsRight > 100)`

QT Version 3.0 or later

See also MoveTo, MoveBy, BoundsLeft, BoundsTop, BoundsBottom

BoundsTop

Syntax `BoundsTop`

Description As a sprite is moved, rotated, stretched and skewed, the bounding rectangle of the sprite changes. BoundsTop returns the top edge of that bounding rectangle.

Return Numeric value indicating the top edge of the sprite.

Example `If (BoundsTop > 100)`

QT Version 3.0 or later

See also MoveTo, MoveBy, BoundsLeft, BoundsRight, BoundsBottom

FirstCornerX

Syntax `FirstCornerX`

Description The corners of a sprite that is in its natural drawing state are the top left corner (FirstCorner), top right corner (SecondCorner), bottom right corner (ThirdCorner), and bottom left corner (FourthCorner). Once the sprite has been rotated and distorted, these corners will move to new positions. FirstCornerX returns the current X coordinate of the first corner.

Return Numeric value indicating the X coordinate of the first corner.

Example `If (FirstCornerX < 10)`

QT Version 3.0 or later

See also ResetMatrix, Stretch

FirstCornerY

Syntax `FirstCornerY`

Description The corners of a sprite that is in its natural drawing state are the top left corner (FirstCorner), top right corner (SecondCorner), bottom right corner (ThirdCorner), and bottom left corner (FourthCorner). Once the sprite has been rotated and distorted, these corners will move to new positions. FirstCornerY returns the current Y coordinate of the first corner.

Return Numeric value indicating the Y coordinate of the first corner.

Example `If (FirstCornerY < 10)`

QT Version 3.0 or later

See also ResetMatrix, Stretch

FourthCornerX

Syntax `FourthCornerX`

Description The corners of a sprite that is in its natural drawing state are the top left corner (FirstCorner), top right corner (SecondCorner), bottom right corner (ThirdCorner), and bottom left corner (FourthCorner). Once the sprite has been rotated and distorted, these corners will move to new positions. FourthCornerX returns the current X coordinate of the fourth corner.

Return Numeric value indicating the X coordinate of the fourth corner.

Example

```
If (FourthCornerX < 10)
```

QT Version 3.0 or later

See also ResetMatrix, Stretch

FourthCornerY

Syntax `FourthCornerY`

Description The corners of a sprite that is in its natural drawing state are the top left corner (FirstCorner), top right corner (SecondCorner), bottom right corner (ThirdCorner), and bottom left corner (FourthCorner). Once the sprite has been rotated and distorted, these corners will move to new positions. FourthCornerY returns the current Y coordinate of the fourth corner.

Return Numeric value indicating the Y coordinate of the fourth corner.

Example

```
If (FourthCornerY < 10)
```

QT Version 3.0 or later

See also ResetMatrix, Stretch

GetSpriteName

Syntax `GetSpriteName`

Description This property returns name of the sprite. The name of a sprite can be set in the Sprite Track Sample Editor Window in LiveStage Professional.

Return A string value indicating the name of the sprite.

Example

```
LocalVars spriteName
spriteName = ThisSprite.GetSpriteName
// Retrieve the name of the sprite
```

QT Version 5.0 or later

See also ID

ID

Syntax `ID`

Description This property returns a numeric value that uniquely identifies the sprite. This is the value that would be supplied to the Target specifier SpriteOfID. You can give your sprites any ids that you desire and can use the id to do special processing, like using the id to specify the pitch in the PlayNote action.

Return Numeric value indicating the ID of the sprite.

Example `SetImageIndexTo(ID)`

QT Version 3.0 or later

See also Targets, SpriteOfID

ImageIndex

Syntax `ImageIndex`

Description This property returns the index of the image being used by the sprite. Sprites use images to render their visual appearance. Setting the image index to refer to another image in the sprite track changes the appearance of the sprite.

Return Numeric value indicating the image index being used by the sprite.

Example `If (ImageIndex = 3)`

QT Version 3.0 or later

See also SetImageIndexTo, SetImageIndexBy, NumImages

ImageRegistrationPointX

Syntax `ImageRegistrationPointX`

Description This property returns the X coordinate of the registration point of the image used by the sprite. Each image in the sprite track has a registration point. This value is the value entered in the image property portion of the Images Tab in the Sprite Sample editor window.

Return Numeric value indicating the X coordinate of the registration point of the image being used by the sprite.

Example `If (ImageRegistrationPointX = 0)`

QT Version 3.0 or later

See also MoveTo, MoveBy

ImageRegistrationPointY

Syntax `ImageRegistrationPointY`

Description This property returns the Y coordinate of the registration point of the image used by the sprite. Each image in the sprite track has a registration point. This value is the value entered in the image property portion of the Images Tab in the Sprite Sample editor window.

Return Numeric value indicating the Y coordinate of the registration point of the image being used by the sprite.

Example `If (ImageRegistrationPointY = 0)`

QT Version 3.0 or later

See also MoveTo, MoveBy

Index

Syntax `Index`

Description This property returns the index of the sprite. The index is the order number of the sprite in the list of sprites in the sprite track. This is the value that would be supplied to the Target specifier SpriteOfIndex.

Return Numeric value indicating the index of the sprite in the sprite list of the sprite track.

Example `If (Index > 10)`

QT Version 3.0 or later

See also Targets, SpriteOfIndex, NumSprites

IsVisible

Syntax `IsVisible`

Description This property returns a boolean value of *TRUE* if the sprite is visible otherwise *FALSE* is returned. Sprites that are not visible still receive mouse events; they are just not drawn when they are not visible.

Return Boolean value of *TRUE* if the sprite is visible otherwise *FALSE*.

Example `If (IsVisible = TRUE)`

QT Version 3.0 or later

See also SetVisible, ToggleVisible

Layer

Syntax `Layer`

Description This property returns a numeric value indicating the visual layer of the sprite within the sprite track. Sprites with a higher layer number are drawn first. This means that sprites with a higher number appear behind sprites with a lower layer number.

Return Numeric value indicating the layer of the sprite.

Example `If (Layer < SpriteOfID(1).Layer)`

QT Version 3.0 or later

See also SetLayerTo, SetLayerBy

SecondCornerX

Syntax `SecondCornerX`

Description The corners of a sprite that is in its natural drawing state are the top left corner (FirstCorner), top right corner (SecondCorner), bottom right corner (ThirdCorner), and bottom left corner (FourthCorner). Once the sprite has been rotated and distorted, these corners will move to new positions. SecondCornerX returns the transformed X coordinate of the second corner.

Return Numeric value indicating the X coordinate of the second corner.

Example `If (SecondCornerX < 10)`

QT Version 3.0 or later

See also ResetMatrix, Stretch

SecondCornerY

Syntax `SecondCornerY`

Description The corners of a sprite that is in its natural drawing state are the top left corner (FirstCorner), top right corner (SecondCorner), bottom right corner (ThirdCorner), and bottom left corner (FourthCorner). Once the sprite has been rotated and distorted, these corners will move to new positions. SecondCornerY returns the current Y coordinate of the second corner.

Return Numeric value indicating the Y coordinate of the second corner.

Example `If (SecondCornerY < 10)`

QT Version 3.0 or later

See also ResetMatrix, Stretch

ThirdCornerX

Syntax `ThirdCornerX`

Description The corners of a sprite that is in its natural drawing state are the top left corner (FirstCorner), top right corner (SecondCorner), bottom right corner (ThirdCorner), and bottom left corner (FourthCorner). Once the sprite has been rotated and distorted, these corners will move to new positions. ThirdCornerX returns the current X coordinate of the third corner.

Return Numeric value indicating the X coordinate of the third corner.

Example `If (ThirdCornerX < 10)`

QT Version 3.0 or later

See also ResetMatrix, Stretch

ThirdCornerY

Syntax `ThirdCornerY`

Description The corners of a sprite that is in its natural drawing state are the top left corner (FirstCorner), top right corner (SecondCorner), bottom right corner (ThirdCorner), and bottom left corner (FourthCorner). Once the sprite has been rotated and distorted, these corners will move to new positions. ThirdCornerY returns the current Y coordinate of the third corner.

Return Numeric value indicating the Y coordinate of the third corner.

Example `If (ThirdCornerY < 10)`

QT Version 3.0 or later

See also ResetMatrix, Stretch

Sprite Actions

ClickOnCodec

Syntax `ClickOnCodec(`*`X, Y`*`)`

X An integer expression indicating the X coordinate of the location to simulate a mouse click.

Y An integer expression indicating the Y coordinate of the location to simulate a mouse click.

Description This action causes the ripple codec to ripple as if the user clicked at the point specified by the parameters X and Y.

Example The following statement passes the coordinate (10, 10) to the ripple codec:

```
ClickOnCodec(10, 10)
```

QT Version 3.0 or later

See also Numeric_Constant, PassMouseToCodec

ExecuteEvent

Syntax `ExecuteEvent(`*`Handler_ID`*`)`

Handler_ID An integer constant specifying the ID of the handler to call.

Description This action causes the specified event handler of a sprite to be executed. There are constants defined for the built-in event handlers so that you can call them too. You can also use the pre-defined define instead of the handler ID like so:

```
ExecuteEvent($”My Handler”)
```

This approach may also be used to refer to a custom event by name instead of its ID:

```
ExecuteEvent($”custom event name”)
```

Example The following statement causes the current sprite to execute its custom event 1000:

```
ExecuteEvent(1000)
```

QT Version 3.0 or later

See also Numeric Expression, Defines

MoveBy

Syntax `MoveBy(X, Y)`

X An integer expression indicating the X distance to move by.

Y An integer expression indicating the Y distance to move by.

Description This action causes a sprite to be moved by the specified amount. You can use this in the idle event handler to cause the sprite to steadily move around on the stage.

Example The following statement causes the current sprite to be moved towards the top left by 1 pixel:

```
MoveBy(-1, -1)
```

QT Version 3.0 or later

See also Numeric Expression, MoveTo

MoveTo

Syntax `MoveTo(X, Y)`

X An integer expression indicating the X coordinate of the location to move the sprite to.

Y An integer expression indicating the Y coordinate of the location to move the sprite to.

Description This action causes the registration point of a sprite to be moved to the specified coordinates.

Example The following statement causes the current registration point of sprite to be moved to the coordinate (0, 0):

```
MoveTo(0, 0)
```

QT Version 3.0 or later

See also Numeric Expression, MoveBy

PassMouseToCodec

Syntax PassMouseToCodec

Description When the mouse is clicked on a sprite, you can call this action to pass the mouse click on to the codec that is rendering the image for the sprite. If this is the ripple codec, this will cause a ripple effect to appear where the mouse was clicked.

Example The following statement when placed in the Mouse Click handler of a sprite will cause the ripple codec to ripple where the mouse was clicked: PassMouseToCodec

QT Version 3.0 or later

See also ClickOnCodec

ResetMatrix

Syntax `ResetMatrix`

Description This action causes the display matrix of the sprite to be reset, thus undoing any Spatial transformations that had been applied to it.

Example `ResetMatrix`

QT Version 3.0 or later

See also Rotate, Scale

Rotate

Syntax `Rotate(Degrees)`

Degrees An integer expression indicating the number of degrees to rotate the sprite by.

Description This action causes a sprite to be rotated by the specified number of degrees. You can only rotate the sprite to a specific angle by first resetting its matrix by calling ResetMatrix and then calling Rotate.

Example The following statement causes the current sprite to rotate by 45 degrees counter clockwise:

```
Rotate(-45)
```

QT Version 3.0 or later

See also Numeric Expression, ResetMatrix

Scale

Syntax `Scale(`*`X`*`, `*`Y`*`)`

X An integer expression indicating the amount to scale the sprite in the horizontal direction.

Y An integer expression indicating the amount to scale the sprite in the vertical direction.

Description This action causes a sprite to be scaled by the specified amounts. Scale values smaller than 1 specify that the sprite is to be shrunk while values greater than 1 enlarge the sprite.

Example The following statement causes the current sprite to be expanded by 200%:

```
Scale(2, 2)
```

QT Version 3.0 or later

See also Numeric Expression, ResetMatrix

SetGraphicsModeBy

Syntax `SetGraphicsModeBy(`*`Mode, Red_Color, Green_Color, Blue_Color`*`)`
<***MIN***(*number*) ***MAX***(*number*) ***wraparound***>

Mode A constant indicating one of the graphic modes to use.

Red_Color A numeric constant indicating the brightness of the red component.

`Green_Color`A numeric constant indicating the brightness of the green component.

`Blue_Color` A numeric constant indicating the brightness of the blue component.

Description This action changes the graphic mode a sprite uses to render itself by the specified amount. The parameter Mode is a predefined constant indicating the graphic mode to set (see SetGraphicsModeBy see Appendix II for a list of the graphic mode constants).

The parameters Red_Color, Green_Color and Blue_Color are Numeric Expressions specifying the amount each primary colors is to change by.

Example The following statement changes the red, green and blue components of the graphic mode of the current sprite by 1000:

```
SetGraphicsModeBy(srcCopy, 1000, 1000, 1000)
```

QT Version 3.0 or later

See also Numeric Expression, SetGraphicsModeTo

SetGraphicsModeTo

Syntax `SetGraphicsModeTo(Mode, Red_Color, Green_Color, Blue_Color)` <***MIN***(*number*) ***MAX***(*number*)>

`Mode` A constant indicating one of the graphic modes to use.

`Red_Color` A numeric constant indicating the brightness of the red component

`Green_Color`A numeric constant indicating the brightness of the green component.

`Blue_Color` A numeric constant indicating the brightness of the blue component.

Description This action sets the graphic mode a sprite uses to render itself. The parameter Mode is a predefined constant indicating the graphic mode to set. See the list below or Appendix II—Drawing Mode Reference.

AVAILABLE DRAWING MODES

```
srcCopy
srcOr
srcXor
srcBic
notSrcCopy
notSrcOr
notSrcXor
notSrcBic
blend
addPin
addOver
subPin
transparent
addMax
subOver
adMin
grayishTextOr
hilite
ditherCopy
```

```
graphicsModeStraightAlpha
graphicsModePreWhiteAlpha
graphicsModePreBlackAlpha
graphicsModeComposition
graphicsModeStraightAlphaBlend
graphicsModePreMulColorAlpha
```

Example The following statement graphic mode to be set to transparent using black as the transparent color:

```
SetGraphicModeTo(transparent, 0, 0, 0)
```

QT Version 3.0 or later

See also Numeric Expression, SetGraphicsModeBy

SetImageIndexBy

Syntax `SetImageIndexBy(`*`index`*`)` <***MIN***(*number*) ***MAX***(*number*) ***wraparound***>

`Index` A numeric expression indicating the amount to change the image index by for the sprite.

Description This action alters the image index of a Sprite by the specified amount. The parameter Index indicates the amount to modify the image index of a sprite by.

Images in a LiveStage project can be referenced by their Index numbers. Changing the image index of the sprite has the effect of changing its appearance since setting a different image index would cause a different image to be used in rendering the Sprite. `SetImageIndexBy can be used to animate a Sprite using a series of images.`

Example The following code fragment sets the current image index of the sprite to the next image in a series of 5 images. This example assumes that there is a Sprite and a series of 5 images in the LiveStage project with the index of 1 through 5.

In the Idle event handler of the Sprite enter the following QScript statements:

```
SetImageIndexBy(1) MIN(1) MAX(5) wraparound
```

Export and run the movie. The result is a movie that animates the 5 images that you have in the project in a continuous loop.

QT Version 3.0 or later

See also Numeric Expression, SetImageIndexTo, ImageIndex

SetImageIndexTo

Syntax `SetImageIndexTo(`*`Index`*`)` <***MIN***(*number*) ***MAX***(*number*)>

`Index` A numeric expression indicating the new image index to use for the sprite.

Description This action sets the image index of a Sprite. The parameter Index is indicates the new image index to set for the Sprite.

Images in a LiveStage project can be referenced by their Index numbers or by using their name, which is pre-defined to be its index. Changing the image index of a sprite has the effect of changing its appearance since setting a different image index would cause a different image to be used in rendering the Sprite. SetImageIndexTo can be used to animate a Sprite using a series of images.

Example The following code fragment sets the image index of the current sprite to the next image in a series of 5 images. This example assumes that there is a Sprite and a series of 5 images in the LiveStage project with the index of 1 through 5.

In the Frame Loaded event handler of the Sprite enter the following QScript statements:

```
GlobalVars CurrentCel
CurrentCel = 0
```

In the Idle event handler of the Sprite enter the following QScript statements:

```
GlobalVars CurrentCel
CurrentCel = CurrentCel + 1
SetImageIndexTo(CurrentCel) MIN(1) MAX(5)
```

Export and run the movie. The result is a movie that animates the 5 images that you have in the project and halts at the last image.

QT Version 3.0 or later

See also Numeric Expression, SetImageIndexBy, ImageIndex

SetLayerBy

Syntax `SetLayerBy(`*`layer`*`)` <***MIN***(*number*) ***MAX***(*number*) ***wraparound***>

`layer` A numeric expression indicating the amount to change the layer by.

Description This action changes the visual layer of a sprite by the amount specified. Sprites with a higher number layer will draw first. This means that they will appear behind sprites with a lower layer number.

Example `SetLayerBy(1) Min(10) Max(20)`

QT Version 3.0 or later

See also Numeric Expression, layer, SetLayerTo

SetLayerTo

Syntax `SetLayerTo(`*`layer`*`)` <***MIN***(*number*) ***MAX***(*number*)>

`layer` A numeric expression indicating the layer.

Description This action sets the visual layer of a sprite. Sprites with a higher number layer will draw first. This means that they will appear behind sprites with a lower layer number.

Example `SetLayerTo(1)`

QT Version 3.0 or later

See also Numeric Expression, Layer

SetVisible

Syntax `SetVisible(`*`Visible`*`)`

`Visible` A boolean expression that makes the sprite visible if *`TRUE`*.

Description This action shows or hides a Sprite. The Visible parameter is a Boolean_Expression indicating that the sprite should be shown (*TRUE*) or hidden (*FALSE*).

Example The following statement causes the current sprite to be hidden:

```
SetVisible(FALSE)
```

QT Version 3.0 or later

See also Boolean_Expression, IsVisible, ToggleVisible

Stretch

Syntax `Stretch(X1, Y1, X2, Y2, X3, Y3, X4, Y4)`

X1-X4 The x coordinate of each of the four corners of the sprite.

Y1-Y4 The y coordinate of each of the four corners of the sprite.

Description The Stretch action is a very powerful function that allows you to manipulate the shape of a sprite. The parameters you specify in the Stretch action represent the four corners of the sprite. These start with the top left, top right, bottom right and finally the bottom left.

This action causes a sprite to be skewed. The parameters X1, Y1 through X4, Y4 specifying Numeric Expressions containing a series of 4 points corresponding to the four corners of the sprite specifying the new location of the corners.

Although Stretch gives you a great deal of control of a sprites shape there are a few quirks in QuickTime which can make the stretch action operate in unexpected ways.

No Warped Corners Allowed.

QT Version 3.0 or later

See also Numeric Expression, ResetMatrix

ToggleVisible

Syntax `ToggleVisible`

Description This action toggles the visible state of a Sprite.

Example The following statements cause the current sprite to be shown and then hidden:

```
SetVisible(TRUE)
ToggleVisible
```

QT Version 3.0 or later

See also SetVisible, IsVisible

TEXT TRACK PROPERTIES AND ACTIONS

Text Track properties and actions need to have a track and movie target specified. If no movie target is specified then the current movie is considered to be the target. You can specify a movie target by using MovieNamed(name) or MovieOfID(id) and pass in either the name of the movie or its ID. If no track target is specified then the track that contains the object that is currently executing the script is considered to be the target. You can specify a track target by using TrackNamed(name), TrackOfID(id), TrackOfIndex(index) or TrackOfType(type). See the section on targets for a complete description and some examples of how to specify a target.

Track and Spatial Track Properties and Actions are available for Text Track targets.

The target Text Track's Edit State must be set to `kScriptEditing` using the SetTextEditState action before you can alter the appearance or the data contained in the text track (see SetTextEditState).

Text Track Properties

GetEditState

Syntax `GetEditState`

Description This property returns current edit state of the target text track.

The available edit states are:

```
kNoEditing
kDirectEditing
kScriptEditing
```

Return A numeric value indicating the current edit state of the target text track.

QT Version 5.0 or later

GetIdleFrequency

Syntax `GetIdleFrequency`

Description This property returns the idle frequency in movie time in 1/60 second increments.

Return Numeric value indicating the idle frequency.

Example

```
LocalVars idleFreq
idleFreq = GetIdleFrequency
// Get the idle frequency of the current track
```

QT Version 5.0 or later

See also SetIdleFrequency

GetSelectionStart

Syntax `GetSelectionStart`

Description This property returns numeric start position of the current selection in the target text track.

Return A numeric value indicating the current selection start position in the target text track.

QT Version 5.0 or later

See also ReplaceText, GetSelectionEnd, GetText

GetSelectionEnd

Syntax `GetSelectionEnd`

Description This property returns numeric end position of the current selection in the target text track.

Return A numeric value indicating the current selection end position in the target text track.

QT Version 5.0 or later

See also ReplaceText, GetSelectionStart, GetText

GetText

Syntax `GetText(`*`start, end`*`)`

`start` A numeric literal or variable specifying the location of the first character position to retrieve.

`end` A numeric literal or variable specifying the locationof the last character position to retrieve.

Description This action retrieves the specified selection of characters (as defined by the parameters *`start`* and *`end`*).

The *`start`* and *`end`* specifies a selection of characters. In order to specify the starting character, you must specify the character position (1 based) of the character preceding the start character of your selection (i.e. Specify 0 to select from the first character). In order to specify the ending character, you must specify the character position (1 based) of the character following the last character in your selection.

Example

```
LocalVars theString, theText
SetString(theString, “This is the first string”)
SetString(theText, GetText(0, 5))
// The resulting string now contains “this”
```

QT Version 5.0 or later

See also ReplaceText

GetTextBoxBottom

Syntax `GetTextBoxBottom`

Description This property returns the location of the bottom of the text box in the coordinates of the target text track.

Return A numeric value indicating location of the bottom edge of the text box of the target text track.

QT Version 5.0 or later

See also GetTextBoxLeft, GetTextBoxRight, GetTextBoxTop, SetTextBox

GetTextBoxLeft

Syntax `GetTextBoxLeft`

Description This property returns the location of the left side of the text box in the coordinates of the target text track.

Return A numeric value indicating location of the left edge of the text box of the target text track.

QT Version 5.0 or later

See also GetTextBoxBottom, GetTextBoxRight, GetTextBoxTop, SetTextBox

GetTextBoxRight

Syntax `GetTextBoxRight`

Description This property returns the location of the right side of the text box in the coordinates of the target text track.

Return A numeric value indicating location of the right edge of the text box of the target text track.

QT Version 5.0 or later

See also GetTextBoxBottom, GetTextBoxLeft, GetTextBoxTop, SetTextBox

GetTextBoxTop

Syntax `GetTextBoxTop`

Description This property returns the location of the top of the text box in the coordinates of the target text track.

Return A numeric value indicating location of the top edge of the text box of the target text track.

QT Version 5.0 or later

See also GetTextBoxBottom, GetTextBoxLeft, GetTextBoxRight, SetTextBox

GetTextLength

Syntax `GetTextLength`

Description This property returns the number of characters in the target text track.

Return A numeric value indicating the number of characters in the target text track. The text length includes spaces.

QT Version 5.0 or later

Text Track Actions

EatKeyEvent

Syntax `EatKeyEvent`

Description This action prevents the text track from processing the Key Pressed event. This action must be called from within the Key Pressed Event handler.

QT Version 5.0 or later

EnterText

Syntax `EnterText(`*`character`*`)`

`character` A string literal or variable specifying the character to use.

Description This action replaces the current selection in the target text track with the specified character.

This action is useful for simulating a text entry field using a text track sample. As the characters are typed, the KeyPressed event is triggered so that each character can be sent to the text sample to simulate text entry.

In order to use this action, you must first set the edit mode of the Text Track to `kScriptEditing` (See SetTextEditState).

QT Version 5.0 or later

See also SetSelection, GetSelectionStart, GetSelectionEnd

FindText

Syntax `FindText(`*`string, red, green, blue, flags`*`)`

`string` A string literal or variable containing the text to search for.

`red` A numeric literal specifying the red component of the found text color.

`green` A numeric literal specifying the green component of the found text color.

`blue` A numeric literal specifying the blue component of the found text color.

`flags` A numeric literal or variable specifying the search settings.

Description This action searches and marks the specified text in the target text track. *`string`* supplies the text to search for while *`red, green, blue`* parameters supply the found text color. The movie time will advance to the sample where the text is found if the `kSearchCurrentSample` flag is not used.

The available search flags are:

```
kSearchCurrentSample
kSearchCaseSensitive
kSearchReverse
kSearchWraparound
kSearchAgain
```

The search flags may be used together by adding the constants together (i.e. `kSearchCaseSensitive + kSearchReverse`).

The selection will be set if the specified text was found so that you can retrieve the start and end of the text in the target text track.

In order to use this action, you must first set the edit mode of the Text Track to `kScriptEditing` (See SetTextEditState).

Example

```
SetTextEditState(kScriptEditing)
FindText(“LiveStage”, 10000, 10000, 10000,
kSearchCaseSensitive + kReverse)
// Search backwards to find the word “LiveStage”
// and mark the text with a gray color
```

QT Version 5.0 or later

See also GetSelectionStart, GetSelectionEnd

HandleMouseClick

Syntax `HandleMouseClick`

Description This action passes the mouse events to the text track allowing it to make text selections and insertion point changes.

This action is effective when the kScriptEditing flag is set using the SetTextEditState action.

QT Version 5.0 or later

See also SetTextEditState, GetEditState

ReplaceText

Syntax `ReplaceText(`*`string, start, end`*`)`

`string` A string literal or variable containing the text used to replace the specified selection.

`start` A numeric literal or variable specifying the location of the first character to replace.

`end` A numeric literal or variable specifying the location of the last character to replace.

Description This action replaces the specified selection of characters (as defined by the parameters *`start`* and *`end`*) with the characters in the supplied *`string`*.

The character positions begin with 0. The end character index you specify must be 1 greater than the character position. Thus in the string “Hello World!”, you must specify 0 for start and 5 for end in order to replace the word “Hello”.

In order to use this action, you must first set the edit mode of the Text Track to `kScriptEditing` (See SetTextEditState).

Example

```
LocalVars theText
SetTextEditState(kScriptEditing)
SetString(theText, “This is the first string”)
ReplaceText(theText, 0, 1000)
// Replace the first 1000 characters in the
// Target text track with the value in theText
```

QT Version 5.0 or later

See also GetSelectionStart, GetSelectionEnd, GetText

ScrollText

Syntax `ScrollText(xdelta, ydelta)`

`xdelta` A numeric literal or variable specifying the amount and direction to scroll the text horizontally.

`ydelta` A numeric literal or variable specifying the amount and direction to scroll the text vertically.

Description This action scrolls the text in the target text track. To scroll the text in the reverse direction (i.e. scrolling from left to right or scrolling top to bottom), specify a negative scroll value.

In order to use this action, you must first set the edit mode of the Text Track to `kScriptEditing` (See SetTextEditState).

Example

```
SetTextEditState(kScriptEditing)
ScrollText(-10, 0)
// Scroll the text from left to right by 10 pixels
```

QT Version 5.0 or later

See also SetTextScrollDelay

SetBackgroundColor

Syntax `SetBackgroundColor(red, green, blue)`

`red` A numeric literal specifying the red component of the background color.

`green` A numeric literal specifying the green component of the background color.

`blue` A numeric literal specifying the blue component of the background color.

Description This action sets the background color of the target text track.

In order to use this action, you must first set the edit mode of the Text Track to `kScriptEditing` (See SetTextEditState).

QT Version 5.0 or later

See also SetTextColor

SetIdleFrequency

Syntax `SetIdleFrequency(frequency)`

`frequency` An integer expression of the frequency of the idle events in 1/60 second increments.

Description This action will change the idle frequency of the text track. Idle Frequency determines how often the Idle Event handlers are called by QuickTime.

Example

```
SetIdleFrequency(60)
// Set the idle frequency of the track to once
// every second
```

QT Version 5.0 or later

SetSelection

Syntax `SetSelection(start, end)`

start A numeric literal or variable specifying the location of the first character in the selection.

end A numeric literal or variable specifying the location of the last character in the selection.

Description This action selects the specified characters (as indicated by the start and end parameters) in the target text track.

The character positions begin with 0. The end character index you specify must be 1 greater than the character position. Thus in the string “Hello World!”, you must specify 0 for start and 5 for end in order to select the word “Hello”.

In order to use this action, you must first set the edit mode of the Text Track to `kScriptEditing` (See SetTextEditState).

QT Version 5.0 or later

See also GetSelectionStart, GetSelectionEnd, ReplaceText

SetTextAlignment

Syntax `SetTextAlignment(alignment)`

alignment A numeric literal or variable indicating text alignment.

Description This action sets the text alignment for the current text selection in the target text track.

The available text alignments are:

```
kCenterJustify
kLeftJustify
kRightJustify
```

In order to use this action, you must first set the edit mode of the Text Track to `kScriptEditing` (See SetTextEditState).

QT Version 5.0 or later

See also SetTextFont, SetTextStyle, SetTextSize

SetTextBox

Syntax `SetTextBox(left, top, right, bottom)`

left A numeric literal or variable specifying the location of the bottom of the left side of the text box.

top A numeric literal or variable specifying the location of the bottom of the top of the text box.

right A numeric literal or variable specifying the location of the bottom of the right side of the text box.

bottom A numeric literal or variable specifying the location of the bottom of the text box.

Description This action sets the text box coordinates of the target text track.

In order to use this action, you must first set the edit mode of the Text Track to `kScriptEditing` (See SetTextEditState).

QT Version 5.0 or later

See also GetTextBoxLeft, GetTextBoxTop, GetTextBoxRight, GetTextBoxBottom

SetTextColor

Syntax `SetTextColor(red, green, blue)`

`red` A numeric literal specifying the red component of the foreground color.

`green` A numeric literal specifying the green component of the foreground color.

`blue` A numeric literal specifying the blue component of the foreground color.

Description This action sets the foreground color of the current text selection.

In order to use this action, you must first set the edit mode of the Text Track to `kScriptEditing` (See SetTextEditState).

QT Version 5.0 or later

See also SetBackgroundColor

SetTextDisplayFlags

Syntax `SetTextDisplayFlags(flags)`

`flags` A numeric literal or variable indicating the display flags setting.

Description This action sets the display flags of the target text track.

The available display flags are:

```
kDontDisplay
kDontAutoScale
kClipToTextBox
kUseMovieBGColor
kShrinkTextBoxToFIt
kScrollIn
kScrollOut
kHorizScroll
kReverseScroll
kContinuousScroll
kFlowHoriz
kContinuousKaraoke
kDropShadow
kAntiAlias
kKeyedText
kInverseHilite
kTextColorHilite
```

The above flags can be used together to set multiple flags at the same time.

In order to use this action, you must first set the edit mode of the Text Track to `kScriptEditing` (See SetTextEditState).

Example

```
SetTextEditState(kScriptEditing)
SetTextDisplayFlags(kScrollIn + kScrollOut + kHorizScroll)
// Sets the flags to scroll in and out horizontally
```

QT Version 5.0 or later

SetTextDropShadow

Syntax `SetTextDropShadow(xOffset, yOffset, transparency)`

xOffset A numeric literal or variable indicating the horizontal offset of the dropshadow.

end A numeric literal or variable indicating the vertical offset of the dropshadow.

transparency A numeric literal or variable indicating the degree of transparency of the dropshadow.

Description This action sets the dropshadow parameters used for the target text track.

This action is valid only if the kDropShadow flag is set for the target text track. (see SetTextDisplayFlags)

In order to use this action, you must first set the edit mode of the Text Track to `kScriptEditing` (See SetTextEditState).

QT Version 5.0 or later

See also SetTextHilite, SetTextDisplayFlags

SetTextEditState

Syntax `SetTextEditState(state)`

state A numeric literal or variable indicating the text edit state.

Description This action sets the text edit state of the target text track.

The available edit states are:

EDIT STATE	MEANING
`kNoEditing`	The target text track is not editable by either a script or the user
`kDirectEditing`	The target text track is editable by the user but NOT by a script
`kScriptEditing`	The target text track is editable by a script but NOT the user

QT Version 5.0 or later

See also GetEditState, HandleMouseClick

SetTextFont

Syntax `SetTextFont(fontID)`

fontID A numeric literal or variable indicating the Font ID.

Description This action sets the font for the current selection in the target text track.

The available font IDs are:

```
kAthensFont
kCairoFont
kCourierFont
kGenevaFont
kHelveticaFont
kLondonFont
kLosAngelesFont
kMobileFont
kMonacoFont
kNewYorkFont
kSanFranciscoFont
kSymbolFont
kTimesFont
kTorontoFont
kVeniceFont
```

In order to use this action, you must first set the edit mode of the Text Track to `kScriptEditing` (See SetTextEditState).

QT Version 5.0 or later

See also SetTextStyle, SetTextSize

SetTextHilite

Syntax `SetTextHilite(`*`start, end, red, green blue`*`)`

`start` A numeric literal or variable indicating the start position of the selection.

`end` A numeric literal or variable indicating the end position of the selection.

`red` A numeric literal indicating the red component of the hilite color.

`green` A numeric literal indicating the green component of the hilite color.

`blue` A numeric literal indicating the blue component of the hilite color.

Description This action hilites the specified selection of characters (as defined by the start and end parameters) in the target text track using the supplied color.

In order to use this action, you must first set the edit mode of the Text Track to `kScriptEditing` (See SetTextEditState).

QT Version 5.0 or later

See also SetTextDropShadow

SetTextLinkColor

Syntax `SetTextLinkColor(`*`index, red, green, blue`*`)`

`index` A numeric literal or variable indicating the index number of the hyper link.

`red` A numeric literal specifying the red component of the text link color.

`green` A numeric literal specifying the green component of the text link color.

`blue` A numeric literal specifying the blue component of the text link color.

Description This action sets the specified hyper link color in the target text track.

In order to use this action, you must first set the edit mode of the Text Track to `kScriptEditing` (See SetTextEditState).

QT Version 5.0 or later

See also SetTextLinkStyle

SetTextLinkStyle

Syntax `SetTextLinkStyle(`*`index, style`*`)`

`index` A numeric literal or variable indicating the index number of the hyperlink.

`style` A numeric literal or variable indicating the Font Style.

Description This action sets the font style for the specified (*`index`*) hyper link text in the target text track.

The available text styles are:

```
kNormalFace
kBoldFace
kItalicFace
kUnderLineFace
kOutlineFace
kShadowFace
kCondenseFace
kExtendFace
```

In order to use this action, you must first set the edit mode of the Text Track to `kScriptEditing` (See SetTextEditState).

QT Version 5.0 or later

See also SetTextLinkColor

SetTextScrollDelay

Syntax `SetTextScrollDelay(`*`delay`*`)`

`delay` A numeric literal or variable indicating amount of time between scrolling in and scrolling out.

Description This action sets the amount of time that is spent between scrolling in and scrolling out in the target text track. This action works with the `kScrollIn` and `kScrollOut` flags used with the *`SetTextDisplayFlags`* action.

When used with the `kScrollIn` or the `kScrollOut` display flags, this action sets the amount of time the text is displayed between the Scroll-in and Scroll-out transitions.

In order to use this action, you must first set the edit mode of the Text Track to `kScriptEditing` (See SetTextEditState).

QT Version 5.0 or later

See also ScrollText

SetTextSize

Syntax `SetTextSize(`*`size`*`)`

`size` A numeric literal or variable indicating the Font Point Size.

Description This action sets the font size in points for the current selection in the target text track.

In order to use this action, you must first set the edit mode of the Text Track to `kScriptEditing` (See SetTextEditState).

QT Version 5.0 or later

See also SetTextFont, SetTextStyle

SetTextStyle

Syntax `SetTextStyle(`*`style`*`)`

`style` A numeric literal or variable indicating the Font Style.

Description This action sets the font style for the current selection in the target text track.

The available text styles are:

```
kNormalFace
kBoldFace
kItalicFace
kUnderLineFace
kOutlineFace
kShadowFace
kCondenseFace
kExtendFace
```

In order to use this action, you must first set the edit mode of the Text Track to `kScriptEditing` (See SetTextEditState).

QT Version 5.0 or later

See also SetTextFont

VR TRACK PROPERTIES AND ACTIONS

VR Track properties and actions need to have a track and movie target specified. If no movie target is specified then the current movie is considered to be the target. You can specify a movie target by using MovieNamed(name) or MovieOfID(id) and pass in either the name of the movie or its ID. If no track target is specified then the track that contains the object that is currently executing the script is considered to be the target. You can specify a track target by using TrackNamed(name), TrackOfID(id), TrackOfIndex(index) or TrackOfType(type). See the section on targets for a complete description and some examples of how to specify a target.

VR Tracks are also Spatial Tracks as they have a visual representation within the movie. Thus the Spatial Track Actions and Properties are also available for VR Track Targets (see the section on Targets).

VR Track Properties

GetIdleFrequency

Syntax `GetIdleFrequency`

Description This property returns the idle frequency in movie time in 1/60 second increments.

Return Numeric value indicating the idle frequency.

Example

```
LocalVars idleFreq
idleFreq = GetIdleFrequency
// Get the current track's idle frequency
```

QT Version 5.0 or later

See also SetIdleFrequency

ViewHorizCenter

Syntax `ViewHorizCenter`

Description This property returns the horizontal view center.

Return A numeric value indicating the horizontal view center.

QT Version 5.0 or later

See also ViewVertCenter

ViewVertCenter

Syntax `ViewVertCenter`

Description This property returns the vertical view center.

Return A numeric value indicating the vertical view center.

QT Version 5.0 or later

See also ViewHorizCenter

AreHotSpotsVisible

Syntax `AreHotSpotsVisible`

Description This property returns TRUE if the HotSpots are shown.

Return A boolean value indicating TRUE if the HotSpots are visible.

Example

```
if (TrackNamed(“MyPano”).AreHotSpotsVisible = TRUE)
// Script executes only when hotspots are visible
EndIf
```

QT Version 5.0 or later

See also ShowHotSpots, HideHotSpots

PanAngle

Syntax `PanAngle`

Description This property returns the side to side (Pan) angle of the VR track. The value returned is between 0 and 360 degrees. The pan angle of a VR track indicates the direction the track is pointing.

Return Numeric value between 0 and 360 degrees indicating the pan angle.

Example

```
If (TrackOfID(3).PanAngle = 0)
// Script executes only when PanAngle is 0
EndIf
```

QT Version 3.0 or later

See also SetPanAngleTo, SetPanAngleBy

TiltAngle

Syntax `TiltAngle`

Description This property returns the up and down (Tilt) angle of the VR track. The value returned is -90 to 90 degrees. A value of 90 indicates that the VR track is tilted vertically upwards while a value of -90 indicates that the track is tilted vertically downwards. A value of 0 indicates the tilt angle is set to the horizontal (no tilt).

Return Numeric value between -90 and 90 degrees.

Example

```
If (TrackOfID(3).TiltAngle = 0)
// Script executes only when TiltAngle is 0
EndIf
```

QT Version 3.0 or later

See also SetTileAngleTo, SetTiltAngleBy

FieldOfView

Syntax `FieldOfView`

Description This property returns the Field of View setting of the VR track. Changing the Field of View changes the zoom factor of the VR track.

Return Numeric value indicating the Field of View setting of the VR track.

Example The following example zooms in by x2:

```
TrackNamed(“VR Movie”).
        SetFieldOfViewTo(
        TrackNamed(“VR Movie”).FieldOfView / 2)
```

The following example zooms out by x2:

```
TrackNamed(“VR Movie”).
        SetFieldOfViewTo(
        TrackNamed(“VR Movie”).FieldOfView * 2)
```

QT Version 3.0 or later

See also SetFieldOfViewTo, SetFieldOfViewBy

NodeID

Syntax `NodeID`

Description This property returns the ID of the current Node in the VR track. A node in a VR track contains the necessary data for displaying a VR panorama. A unique ID identifies each node in a VR track.

Return Numeric value indicating the ID of the current Node in the VR track.

Example `If (TrackOfID(3).NodeID = 21)      ...`

QT Version 3.0 or later

See also GotoNodeID

VR Track Actions

EnableHotSpot

Syntax `EnableHotSpot(`*`HotSpot_ID, enable`*`)`

`HotSpot_ID` A numeric literal, expression or variable indicating the ID of the VR hotspot to enable/disable.

`enable` A boolean literal, expression or variable indicating that the hotspot is to be enabled or disabled

Description This action enables or disables the specified VR HotSpot.

Example

```
ThisTrack.EnableHotSpot(101, FALSE)
// Disable the hotspot with the ID of 101
```

QT Version 5.0 or later

See also DisableHotspot

GoToNodeID

Syntax `GoToNodeID(`*`Node_ID`*`)`

`Node_ID` An integer expression indicating the ID of the VR node to go to.

Description This action sends a VR track to the specified Node. The node ID of 2147483648 (hex 80000000) will go to the previous node and the node ID of 2147483649 (hex 80000001) will go to the default node.

Example `TrackNamed(“Room”). GoToNodeID(100)`

QT Version 3.0 or later

See also Numeric Expression, NodeID

HideHotSpots

Syntax `HideHotSpots`

Description This action hides the VR HotSpots for the current VR node.

Example

```
ThisTrack.HideHotSpots
// Hide the hotspot
```

QT Version 5.0 or later

See also AreHotSpotsVisible, ShowHotSpots

SetFieldOfViewBy

Syntax `SetFieldOfViewBy(`*`Angle`*`)` <***MIN***(*number*) ***MAX***(*number*) ***wraparound***>

`Angle` A numeric expression indicating the change in the field of view angle.

Description This action changes the Field of View of a VR track by the specified amount. See the description in SetFieldOfViewTo for an explanation of what the field of view is.

Example TrackNamed(“Room”).SetFieldOfViewBy(5)

QT Version 3.0 or later

See also Numeric Expression SetFieldOfViewTo, FieldOfView

SetFieldOfViewTo

Syntax `SetFieldOfViewTo(`*`Angle`*`)` <***MIN***(*number*) ***MAX***(*number*)>

`Angle` A numeric expression indicating the field of view angle.

Description This action sets the Field of View of a VR track. The field of view is the angle formed by a line connecting the bottom of the image, to the view point, to the top of the image. A larger field of view will distort the image but will allow you to see more of it at once. The pan and tilt angles may be adjusted to keep the image in view.

Example `TrackNamed(“Room”).SetFieldOfViewTo(90)`

QT Version 3.0 or later

See also Numeric Expression, FieldOfView, SetFieldOfViewBy

SetIdleFrequency

Syntax `SetIdleFrequency(`*`frequency`*`)`

`frequency` An integer expression setting the new frequency of the idle events in 1/60 second increments.

Description This action will change the idle frequency of the sprite track. Idle Frequency determines how often the Idle Event handlers are called by QuickTime.

Example

```
SetIdleFrequency(1 * 60)
// Set the idle frequency of the track to once
// every second
```

QT Version 5.0 or later

SetPanAngleBy

Syntax `SetPanAngleBy(`*`Angle`*`)` <***MIN***(*number*) ***MAX***(*number*) ***wraparound***>

`Angle` A numeric expression indicating the change in the pan angle.

Description This action modifies the pan angle of a VR track. The parameter Angle specifies the number of degrees to modify the side to side angle of the VR track being displayed.

Example The following statement causes the VR track named "Room" to pan continuously (if called from the Idle Handler) to the right by 5 degrees wrapping around when the angle reaches 360:

```
SetPanAngleBy(5) MIN(0) MAX(359) wraparound
```

QT Version 3.0 or later

See also Numeric Expression, SetPanAngleTo, PanAngle

SetPanAngleTo

Syntax `SetPanAngleTo(`*`Angle`*`)` <***MIN***(*number*) ***MAX***(*number*)>

`Angle` A numeric expression indicating the new pan angle.

Description This action sets the pan angle of a VR track. The parameter Angle specifies the side to side angle of the VR track to display.

Example The following statement sets the pan angle of the VR track named "Room" to 0:

```
TrackNamed("Room").SetPanAngleTo(0)
```

QT Version 3.0 or later

See also Numeric Expression, SetPanAngleBy, PanAngle

SetTiltAngleBy

Syntax `SetTiltAngleBy(`*`Angle`*`)` <***MIN***(*number*) ***MAX***(*number*) ***wraparound***>

`Angle` A numeric expression indicating the change in the tilt angle.

Description This action modifies the tilt angle of a VR track. The parameter Angle specifies the number of degrees to modify the up and down angle of the VR track being displayed.

Example The following statement causes the VR track named "Room" to tilt upwards continuously (if added to the Idle Handler) by 5 degrees wrapping around when the angle reaches 90:

```
SetTiltAngleBy(5) MIN(-90) MAX(90) wraparound
```

QT Version 3.0 or later

See also Numeric Expression, TiltAngle, SetTiltAngleTo

SetTiltAngleTo

Syntax `SetTiltAngleTo(`*`Angle`*`)` <***MIN***(*number*) ***MAX***(*number*)>

`Angle` A numeric expression indicating the new tilt angle.

Description This action sets the tilt angle of a VR track. The parameter Angle specifies the up and down angle of the VR track to display.

Example The following statement causes the VR track named “Room” to reset the tilt angle to 0 (horizontal):

```
TrackNamed(“Room”).SetTiltAngleTo(0)
```

QT Version 3.0 or later

See also Numeric Expression, SetTiltAngleBy, TiltAngle

ShowDefaultView

Syntax `ShowDefaultView`

Description This action returns a VR track to its default view.

Example The following statement causes the VR track named “Room” to return to its default view:

```
TrackNamed(“Room”).ShowDefaultView
```

QT Version 3.0 or later

See also SetFieldOfViewTo SetPanAngleTo SetTiltTo

ShowHotSpots

Syntax `ShowHotSpots`

Description This action shows the VR HotSpots for the current VR node.

Example

```
ThisTrack.ShowHotSpots
// Show the hotspots
```

QT Version 5.0 or later

See also AreHotSpotsVisible, HideHotSpots

NEW QSCRIPTS FOR QUICKTIME 6.0

New Text Track Actions

GotoNextChapter

Syntax `GotoNextChapter`

Description Sends play head to the start of the next chapter

Example `ThisMovie.GotoNextChapter`

QT Version 6.0 or later

GotoPreviousChapter

Syntax `GotoPreviousChapter`

Description Sends play head to the start of the previous chapter

Example `ThisMovie.GotoPreviousChapter`

QT Version 6.0 or later

GotoFirstChapter

Syntax `GotoFirstChapter`

Description Sends play head to the start of the first chapter

Example `ThisMovie.GotoFirstChapter`

QT Version 6.0 or later

GotoLastChapter

Syntax `GotoLastChapter`

Description Sends play head to the start of the last chapter

Example `ThisMovie.GotoLastChapter`

QT Version 6.0 or later

GotoChapterByIndex

Syntax `GotoChapterByIndex(`*`Index`*`)`

`Index` The index of the chapter

Description Sends play head to the start of the nth chapter

Example `ThisMovie.GotoLastChapter`

QT Version 6.0 or later

New Text Track Properties

GetChapterCount

Syntax `GetChapterCount`

Description Returns the number of chapter in the movie.

Return Numeric value indicating the number of chapters.

Example

```
LocalVars MyNumberOfChapters
MyNumberOfChapters = ThisMovie.GetChapterCount
```

QT Version 6.0 or later

GetCurrentChapterIndex

Syntax `GetCurrentChapterIndex`

Description Returns the current chapter index of the movie.

Return Numeric value indicating the current chapter index.

Example

```
LocalVars MyChapterIndex
MyChapterIndex = ThisMovie.GetCurrentChapterIndex
```

QT Version 6.0 or later

GetCurrentChapterName

Syntax `GetCurrentChapterName`

Description Returns the current chapter name of the movie.

Return A string value indicating the name of the current chapter.

Example

```
LocalVars MyChapterName
SteString(MyChapterName, ThisMovie.GetCurrentChapterName)
```

QT Version 6.0 or later

GetChapterNameAtIndex

Syntax `GetChapterNameAtIndex(`*`Index`*`)`

`Index` A numeric literal or variable specifying the index of the chapter.

Description Returns the name of the nth chapter

Return A string value indicating the name of the chapter at the index specified.

Example

```
LocalVars MyChapter
SetTring(MyChapter, ThisMovie.GetChapterNameAtIndex(5))
```

QT Version 6.0 or later

GetChapterIndexNamed

Syntax `GetChapterIndexNamed(“`*`name`*`”)`

`Name` The name of the chapter

Description Returns the index of the chapter with passed in name

Return Numeric value indicating the index of the chapter at the chapter name specified.

Example

```
LocalVars MyChapterNamed
MyChapterNamed = GetChapterIndexNamed(“name”)
```

QT Version 6.0 or later

New Movie Actions

SetMovieScale

Syntax `SetMovieScale(`*`xScale, yScale`*`)`

`XScale` A numeric literal or variable specifying the scale ratio of the width

`YScale` A numeric literal or variable specifying the scale ratio of the height

Description Sets the target movie’s dimensions. This operates similarily to QTPlayer’s menu commands for setting the movie’s size. This action does not support variables.

Example: Half Size : 0.5, 0.5
Normal Size : 1.0, 1.0
Double Size : 2.0, 2.0

Example `ThisMovie.SetMovieScale(2, 2)`
`// double the size of the movie.`

QT Version 6.0 or later

New Math Function

SetRandomSeed

Syntax `SetRandomSeed(`*`Seed`*`)`

`Seed` A numeric literal or variable specifying the seed sequence.

Description Sets the QuickDraw seed value, which is the starting point for any subsequent Random calls.

Example

```
SetRandomSeed(TickCount)
// Set the Ramdon to never return the same value, as
TickCount will always be different.
```

QT Version 6.0 or later

New QTVR Action

NOTE: A QTVR object has 3 states depending on the users interaction, Default, MouseDown, and Current. Using the new action, you can set the state without user interaction. This mean that you can decide to have the Object always behave like if it was on MouseDown even if the user does not click on it.

SetViewState

Syntax `SetViewState(`*`ViewStateType, State`*`)`
`ViewStateType` An integer expression indicating the View state to change
`State` An integer expression indicating the View state to change to.

Description QTVR has the concept of view states for object movies depending on the mouse button. These are alternate images that are displayed depending on the state of the mouse button. SetViewState sets the object node's state type to the new state value.

The 3 available states are: `kDefaultViewState, kCurrentViewState, kMouseDownViewState`

Example

```
TrackNamed("QTVR").SetViewState(kDefaultViewState,
kMouseDownViewState)
// Set the default view state to behave like the mouse down state.
```

QT Version 6.0 or later

New QTVR Properties

GetCurrentViewState

Syntax `GetCurrentViewState(`*`ViewStateType`*`)`

`ViewStateType` An integer expression indicating which View state to get.

Description QTVR has the concept of view states for object movies depending on the mouse button. These are alternate images that are displayed depending on the state of the mouse button. *`GetCurrentViewState`* gets the value of a view state.

The 3 available states are: `kDefaultViewState, kCurrentViewState, kMouseDownViewState`

Return Numeric value indicating the value of the view state.

Example

```
TrackNamed(“QTVR”).GetCurrentViewState(kDefaultViewState)
// Get the view state of the state defined.
```

QT Version 6.0 or later

GetViewStateCount

Syntax `GetViewStateCount`

Description QTVR has the concept of view states for object movies depending on the mouse button. These are alternate images that are displayed depending on the state of the mouse button. GetViewStateCount gets the count of view states for an object node.

Return Numeric value indicating the number of view states.

Example

```
LocalVars NumberOfView
NumberOfView  = TrackNamed(“QTVR”).GetViewStateCount
// Get the number of view state in the track defined.
```

QT Version 6.0 or later

LoadNewImage

Syntax `LoadNewImage(`*`URL, ID`*`)`

`URL` Absolute URL of an image to be loaded..

`ID` Integer value of the ID..

Description LoadNewImage akes as a parameter the URL of the image to be requested and an ID with which you can reference that image. Passing an ID of 0 will prompt this action to assign the next available (unique) ID greater than the current image count. In comparison, the index assigned will always be the integer one greater than the current image count.

Example

```
TrackNamed(“SpriteTrack”).LoadNewImage(“http://www.myserver.com/
myimage.png”, 0)
// Load the image defined into the Sprite track.
```

QT Version 6.0 or later

DisposeImage

Syntax `DisposeImage(Index)`

`Index` Integer value of the Index of the image available in the Spirte sample...

Description Removes the Images referred by the index from QT memory.

Example `TrackNamed(“SpriteTrack”).DisposeImage(3)`
`// Dispose from memory the image of index 3`

QT Version 6.0 or later

GetMovieAnnotation

Syntax `GetMovieAnnotation(`*`“String requested”, Flag`*`)`

`Flag` An integer expression indicating formatting of the data returned
`KAnnotationAsString`: return data as a string
`KAnnotationAsXML`: return data as xml formatted string

`String requested` As string: a single annotation type, that is, “qt-userdata-cpy”
As XML: an empty string means return all appropriate user data as an XML-formatted string. Otherwise, it is a comma-deliminated list of the specific user data that the scriptor wants returned as an XML-formatted string.

Description This property allows you to query the predefined annotations available in a movie. Each annotation is predefined by the prefix “qt-userdata-”. For example if you wish to request the Name, you should define “qt-userdata-nam”.

Other Example: `qt-userdata-cpy = Copyright`
`qt-userdata-cmt = Comment`
`qt-userdata-inf = Info`

Example

```
ThisMovie.getMovieAnnotation(“”,KAnnotationAsXML)
// Return all the annotation available in the current movie
formatted as XML.
```

QT Version 6.0 or later

Appendix

2

Drawing Mode Reference

INTRODUCTION

Drawing modes are used in all visual objects within LiveStage Professional. A drawing mode tells QuickTime how the particular visual object is to be drawn in the movie. To view an example of the different drawing modes run the “Graphics Modes.mov” demo movie in the “Featured Demos” folder on the LiveStage Professional 4.0 CD-ROM. This movie allows you to view the various drawing modes and how they work with multiple graphics.

Some drawing modes are designed to perform basic operations that will always operate the same way; others utilize an OpColor. An OpColor is an RGB value that acts as a modifier for the drawing mode. The blend operation is a good example of this. It uses the OpColor to specify how much of a blend will occur between the source image and the destination.

DRAWING MODE DESCRIPTIONS

The drawing modes included in LiveStage Professional are described below, however you may wish to try out the Graphics Modes sample movie included. Playing with this sample movie will provide you with a better understanding of how each of the drawing modes operates.

COPY | SRCCOPY

Performs a straight copy of the source onto the destination.

INVERSE COPY | NOTSRCCOPY

This operates the same as the Copy mode except that color information from the source is inverted. This means that black becomes white and white becomes black.

OR | SRCOR

Operates similar to a copy except any image information set in the source or destination images is kept. For example, if you have two images, one is an A in black and the other is a B in black and they overlap using Or, both will show.

INVERSE OR | NOTSRCOR

Inverse Or inverts the source and then performs a standard Or operation on the source and destination.

EXCLUSIVE OR | SRCXOR

An Exclusive Or drawing mode works like the Or drawing mode except where pixels set in the source and destination images occupy the same location. In this case, the pixels are turned off. Referring to the example in the Or drawing mode, wherever A and B overlap each other, the result would be white pixels.

INVERSE EXCLUSIVE OR | NOTSRCXOR

This drawing mode inverts the source and then performs the Exclusive Or drawing mode. This will cause areas in the source and destination that are black to become white. Areas where the source is black and the destination is white will be drawn in black. If the source is white and the destination is white, the resulting pixels will be white. If the source is white and the destination is black, the result will be black.

BIT CLEAR | SRCBIC

Any pixels that are black in the source are set to white in areas where they overlap black pixels in the destination.

INVERSE BIT CLEAR | NOTSRCBIC

Inverse Bit Clear will clear all pixels that are black in destination when a white pixel in the source overlaps it. Any black pixels in the source will allow black pixels in the destination to show through.

BLEND | BLEND

The Blend graphics mode will produce a weighted average of the source and destination pixels. This graphics mode uses the OpColor to indicate the weighting applied to the blending. If the OpColor is white, the source image data is copied over the destination. Using an OpColor of 50% gray will take half of the source color and half of the destination color to produce the result. Finally, an OpColor of black will use only the destination pixels; no source pixels will be shown.

TRANSPARENT | TRANSPARENT

The color you specify in the OpColor will be rendered transparently in the source image. Note that only the Animation Codec supports this mode properly. Using other codecs with this mode will produce undesirable results.

ADD MAX | ADDMAX

Color values set in the source image data will add to the colors in the destination. Any white pixels in the source image will force white in the destination; black pixels add no color information to the destination thus letting the source pixels to show through.

ADD MIN | ADDMIN

This mode operates opposite to the Add Max graphics mode. With this mode, any white pixels do not affect the destination pixels thus letting the destination pixels to show through. Any black pixels in the source image will force black on the destination.

ADD OVER | ADDOVER

This mode takes the sum of the source and destination image data to produce its result. If the value exceeds the maximum color value, the new value will be the sum minus the maximum color value. The maximum color value is 65535.

ADD PIN | ADDPIN

This mode assigns to the destination pixel the color closest to the sum of the destination pixel values. This value is constrained to a maximum value indicated by the OpColor setting.

SUB OVER | SUBOVER

The result of this graphics mode is the color closest to the difference of the source and destination pixels. If the result of this operation is less than zero the result wraps around to the maximum color value (65535) minus the result.

SUB PIN | SUBPIN

The result of this graphics mode is the color closest to the difference of the sum of the source and difference pixel values. The maximum value of the result is limited by the value of the OpColor.

DITHER | DITHERCOPY

Same as the Copy drawing mode except that dithering is performed if the destination bit depth is less than that of the source.

GRAYISH TEXT OR | GRAYEDTEXTOR

Similar to Or, but makes use of the gray text color to draw text that appears dim.

HILITE | HILITE

Replaces areas of the image with the highlight color defined by the user.

ALPHA CHANNEL

Straight alpha combines the source pixel value with **graphicsModeStraightAlpha** the background pixel based on the value contained in the alpha channel. For example, if the alpha value is zero, only the background pixel will appear. If the alpha value is 255, only the foreground pixel will appear. If the alpha value is 127, then (127/255) of the foreground pixel will be blended with (128/255) of the background pixel to create the resulting pixel, and so on.

PRE-WHITE ALPHA

Pre-multiplied with white means that the color **graphicsModePreWhiteAlpha** components of each pixel have already been blended with white, based on their alpha channel value. Effectively, this means that the image has already been combined with a white background. To combine the image with a different background color, QuickTime must first remove the white from each pixel and then blend the image with the actual background pixels. Images are often pre-multiplied with white as this reduces the appearance of jagged edges around objects.

PRE-BLACK ALPHA

Pre-multiplied with black is the same as **graphicsModePreBlackAlpha** pre-multiplied with white, except the background color that the image has been blended with is black instead of white.

COMPOSITION

Supports rendering of image data like that of **graphicsModeComposition** animated GIFs where differencing can occur between frames, or special rendering operations are performed.

ALPHA BLEND

This graphics mode causes QuickTime to interpret **graphicsModeStraightAlphaBlend** the alpha channel as a straight alpha channel, but when it draws it combines the pixels and applies the color specified in the OpColor to the alpha channel.

PRE-MULTIPLY ALPHA

Pre-Multiply Alpha is the same as pre-multiplied **graphicsModePreMulColorAlpha** with white, except the background color that the image has been blended with is the OpColor instead of white.

Appendix

3

Codec Reference

INTRODUCTION

QuickTime Codecs are used for compressing visual media within LiveStage Professional. The codec tells QuickTime how the particular visual object is to be compressed in the movie.

CODEC DESCRIPTIONS

The codecs included in LiveStage Professional are described below, however you may wish to try out the “Codec Demo” sample movie included in the “Features Demos” folder. Playing with this sample movie will provide you with a better understanding of how each of the codecs operates.

DON’T RECOMPRESS

No additional compression is done on the image data. It is used exactly as provided and incorporated into the movie. Use this if your image is already compressed well or if it is a vector graphic.

GRAPHICS

Best for digital and other related images. This compressor provides very good compression with no loss in image quality. It is almost half the data rate of the Animation codec.

ANIMATION

Specifically designed to handle animated content like cartoon graphics, still screen captures and graphic images with solid color. This can provide very high levels of compression for this kind of content, but is not appropriate for photo type graphics like JPEG images, analog video, or 3D animation. This codec is lossless at the best quality setting, suitable for CD-ROM delivery at lower quality settings and has good temporal compression

ANIMATION + ALPHA

Similar to Animation but this codec is also used when you wish to use a transparent color for the image, as the other codecs do not support transparency.

JPEG

Standard JPEG style compression that is good for photographs or similar images. Can compress to very small images but there is a sacrifice in image quality.

SORENSON

High quality compressor similar to JPEG but creates smaller files. The Sorenson compressor is known to create unusual visual side effects when used for sprites whose matrices are being manipulated extensively.

CINEPAK

High quality compressor for video which can do 10:1 compression. Similar to Sorenson but does not look nearly as good.

VIDEO

Fast encoder/decoder for video compression.

MOTION JPEG A & B

Used for compressing photos. These compressors have Photo JPEG plus field support, are common in many non-linear editing systems, can be broadcast quality at higher data rates and are good for archiving slide shows. These compressors do not have temporal compression, the grayscale is only 8-bit, and every frame is a key frame.

H251

Used for compression of video. This compressor has a faster decoder than H263, has good temporal compression and works best with low motion. This compressor is weaker than H263, not good on a modem connection and is CPU intensive.

H263

Used for compression of video. This compressor is good for video conferencing, is better with high motion at modem speeds, has a fast encoder, and has better image quality than H261. This compressor has a slow decoder, works best at multiples of 16, and is CPU intensive.

PNG

Perfect compressor for images with an Alpha Channel. There are other codecs that you may also specify but they are not for image compression. These codecs allow you to create certain effects like fire within the image.

RIPPLE

Generates a ripple effect like that of a rock being thrown into a pool of water. It should be noted that sprites that utilize the ripple codec only affect sprites in the Sprite Track. You must also ensure that the sprite using the ripple codec is on a layer above the sprites that it is to ripple. Changing the OpColor used with the Ripple Codec will affect the amount of ripple used for each color component.

FIRE

A fire effect which you can use with alpha blending to create some spectacular effects.

CLOUD

Produces cloud effects for use in your movie.

Appendix 4

LiveStage Professional and Applescript

INTRODUCTION

LiveStage Professional has always had one of the most extensive AppleScript dictionaries of any Macintosh application. During the development of LiveStage Professional 4.0, considerable pains have been taken to improve upon previous versions, making LiveStage Professional 4.0 the premier application for AppleScripting. This appendix and accompanying pdf on the CD-ROM is drawn directly from Brennan Young's extensive work and documentation surrounding AppleScript for LiveStage Professional. Totally Hip Software owes Brennan a great deal of gratitude for the work and effort he has put into improving LiveStage Professional's AppleScript dictionary. This appendix is intended to give a head start to those who want to go, as Brennan says, "adventuring" with AppleScript in LiveStage Professional. The meat of the appendix is so extensive that we were forced to include it in digital .pdf format only. The file LSP4ASDictionary.pdf is located on your CD-ROM in the documentation folder.

LiveStage Professional's AppleScript dictionary is constructed according to the best principles - with an emphasis on classes and containment rather than on unique commands - and almost everything that can be done manually can also be done with a script. In fact, there are a few things that can only be accomplished with AppleScript.

NEW Recordability

New to version 4 is recordability. Most actions are now recordable, which means you can set your script editing application to record, and then work normally inside the LiveStage Professional Application. The AppleScript recording your action will be automatically generated for you, but be forewarned that recorded scripts have a tendency to be verbose. They often involve a lot of unnecessary interface activity such as moving windows, so manual editing will always play some role in AppleScripting LiveStage Professional (or any other application for that matter).

WHY IS APPLESCRIPT IMPORTANT FOR LIVESTAGE PROFESSIONAL DEVELOPERS?

Automation

In any production process, there are always areas where automation will increase efficiency. Increased efficiency leads to cost savings and higher profitability or, in today's age, less time in front of your computer and more time for other activities. As Internet audiences increase and the demand for rich media content increases, the ability to automate rich media production becomes a tremendous if not vital advantage for media professionals.

Designing automated work processes takes time. You need to take a step back from your work and find broad similarities and differences in workflow among various projects. This is where you will find opportunity to implement AppleScript to its greatest use. AppleScript is a rare and beautiful example of a technology, which enables ordinary users to create automated processes that go well beyond the capabilities of the individual tools.

With that, we invite you to discover and take advantage of LiveStage Professional's extensive AppleScript dictionary.

Locate the LiveStage Professional AppleScript dictionary (LSP4ASDictionary.pdf) on the CD.

Appendix

5

Glossary

8-BIT

This is the color depth that permits 256 colours to be displayed simultaneously. The colors that are used are shown in the “Palette”. Many older computers only have 8-bit displays.8-bit is also called “256 Colors” on the Macs.

16-BIT

This is the color depth that permits thousands of colours to be displayed simultaneously. Also known as “Thousands of Colors” on Macs,and “High Color “on Windows.

24-BIT

This is the color depth that permits millions of colours to be displayed simultaneously, producing photographic quality images.24-bit mages are also called “Millions of Colors” on Macs, and “True Color” on Windows.

ABSOLUTE PATH

Absolute Paths are the exact (not relative)location of a folder or file, such as http://www.totallyhip.com/lsdn .You generally use Absolute Paths to provide the location of files on different servers, such as RTSP streaming files housed on streaming servers.

ALPHA CHANNEL

This is an additional image channel that is generally used for transparency or compositing. Only certain formats support alpha channels. These include PICT, PNG and the QuickTime Animation CODEC.

ALTERNATE MOVIES

Alternate Movies enable you to create multiple versions of a movie and set criteria for when the QuickTime Plug-in should display the different versions.

BANDWIDTH

Bandwidth is the amount of data that can be processed or moved over a connection in a given amount of time. Bandwidth refers to how fast a computer can receive or send information, generally referred to in kilobits per second (Kbps).

BEHAVIOR

Pre-defined actions that can be applied to sprites. These can be used instead of writing a script to do common actions.

BITMAP IMAGE

Bitmap Images are graphics composed of a 2D array of pixels. Bitmap images are also known as Raster images.

CODEC

Compressor-Decompressor. Provides compression and decompression services for media samples and other data.

COLOR DEPTH

Color Depth is the range of colours that can be used in a movie or image. Color Depth is measured in bits,with higher values such as 24-bit providing higher quality images, though at a higher file size.

CPU-INTENSIVE

Processes that demand a lot of computer processor power are considered to be CPU-Intensive, generally loading the computer down while they are running and may not even function on slower machines.

DATA RATE

This is the amount of information per second a movie produces, generally expressed in KBps (KiloBytes per second)

DITHERING

Dithering improves picture quality when you display an image that exists at a higher bit-depth on a lower bit-depth device. For example, you may want to dither a 24-bit image if your user will display it on an 8-bit screen.

DRAWING MODE

See graphics mode.

EFFECT

An effect is a type of visual media that takes 0, 1 or 2 sources of visual data and uses it to calculate the image it will display. Effects are typically used to do visual transitions such as wipes and dissolves.

EFFECT TRACK

A track in a movie that contains effect samples.

EVENT HANDLER

A script within a sprite that will be run in response to a user action. For example, the mouse click event handler will be run when the user clicks the mouse on the sprite. Event handlers can also be run by the scripts themselves.

EXTERNAL SPRITE TRACK

A track in a movie that contains sprite media samples. Those sprite media sample are imported from an external Movie file. The sprite media sample cannot be modified.

FLASH SAMPLE WINDOW

This window is shown when a Flash Sample is double clicked. In this window, scripts similar to the ones used for sprites can be written and attached to any of the frames in the Flash Sample.

FLASH TRACK

A track in a movie that contains Macromedia Flash Media samples. Scripts can be attached to specific frames of the Flash Sample by double clicking on the Flash Sample. The Flash Track can be modified by double clicking on the Track Header.

FLATTENING

The last stage in the creation of a QuickTime movie, flattening lays out movie data in a completely linear fashion, and removes all references to external media, creating an independent presentation.

GRAPHICS MODE

The property of a sprite or visual track that determines how it will draw. One useful mode is transparent. Used along with an OpColor it will cause areas in the image with that color to be see-through.

IMAGE

An image is a graphic employed in a LiveStage Professional project. Images can be bitmap or vector based.

IMAGE INDEX

A property of a sprite that specifies the image to display. The index of an image is simply its ordinal position within the Image List of a given sprite sample, with 1 being the first image.

IMAGES LIST

This list is visible in the sprite sample window when the images tab is selected. This list shows all the images stored in the sprite sample.

IMAGES TAB

This tab is visible in the sprite sample window. Selecting this tab shows a list of images contained in the sprite sample.

INSTRUMENT SAMPLE

A list of MIDI instruments. These instruments are used by sprites when the “PlayNote” QScript action is run. The index of the instrument in this sample is passed as a parameter in the “PlayNote” action. These MIDI instruments can be built-in MIDI instruments, or you can create your own instruments by using a sound file for the instrument.

INSTRUMENT TRACK

A track in a movie that contains MIDI instrument samples.

INTERPOLATE

To calculate an intermediate value between two values. For example, starting with the end values of 0 and 5, the interpolated values would be 1, 2, 3, and 4.

INTERPOLATOR

An interpolator is a special type of tween that is used to interpolate time values. A tween normally interpolates its data in a linear way as time goes by. A time interpolator will cause the time to go by in a non-linear way. It is most common to use an X-Y Path to do time interpolation. One would create a path that looks like a wave form. Time would be plotted along the X-axis and the result would be read from the Y-axis. This result would be the time actually used in the tween that has this X-Y path as its time interpolator. You can use this to create an ADSR (Attack Decay Sustain Release) type volume envelope for sounds, or to make sprites move in more realistic ways by making them start slowly, accelerate, and then stop slowly.

LAYER

A mechanism for determining drawing order for visual tracks and sprites in a movie. When QuickTime plays a movie, it displays the movie’s tracks according to their layer. Tracks with lower layer numbers are shown closer to the front; tracks with higher layer numbers are shown behind those tracks.

LINEAR

In order or sequence. To linearly change from 1 to 5 means to change like this: 1 to 2 to 3 to 4 to 5. A non-linear change from 1 to 5 would be like this: 1 to 2 to 5. The non-linear change is not smooth. Each change is not by the same amount.

MASK

A black and white image used to remove areas from the original image. The mask must be the same size as the original image and use only black and white. The black indicates that the image in this area is to be drawn and the white indicates that this area should be left alone.

MATRIX

See transformation matrix or the sprite’s matrix (using a path tween).

MATTE

The matte defines which of the Track's pixels are displayed in a movie. A Matte can be 8 bits deep and can specify various degrees of transparency.

MEDIA

Media actually contains the data used in the track for drawing, playing sound or other things. Media usually refers to the media samples in a track.

MEDIA SAMPLE

One indivisible piece of data stored in a track. Sound typically has 22,000 samples per second, while video has 24 samples per second. In the case of video, each sample is one still image.

MODIFIER TRACK

A track in a movie that contains data samples. The data samples are usually simple data types like integers or real numbers. Modifier tracks are typically used to modify some attribute in another track. One example would be to have a modifier track that contains a sequence of integers and to use that modifier track to control the image displayed by a sprite.

MOVIE

A collection of tracks. Each track contains media that is used to display pictures or play sound for example. A movie is typically used to display video and play sound.

MOVIE CONTROLLER

A user interface element displayed at the bottom of the movie that provides a set of controls to control the movie's playback.

MUSIC TRACK

A track in a movie that contains sound samples. The Music Track is imported from an External Video file. The Music data cannot be modified.

OPCOLOR

This is a color value used by some drawing modes to provide special effects. For transparent drawing it indicates the color that will be transparent. For blend drawing it indicates the blend percentage for each color channel (red, green, blue).

PALINDROME LOOPING

When Looping is enabled and Palindrome Looping is selected, the movie runs forward to the end and then backwards back to the beginning repeatedly.

PICTURE TRACK

A track that contains picture samples. Each sample is a single still image. A Picture Track is typically used to provide a static backdrop for a Sprite Track.

PLAYHEAD

A visual indicator in the tracks tab that indicates the current time in the movie.

POSTER FRAME

A single frame of your movie that displays in open file dialog windows and other applications that provide thumbnail previews.

PREVIEW MOVIE

A short version of a movie which displays when the user selects the movie file in an “Open File” dialog window or other application that provides thumbnail previews.

PROGRESSIVE DOWNLOAD

This is media that viewers may watch as it downloads through QuickTime ‘ Fast Start feature. Progressive download is also called HTTP Streaming.

PROPERTY

Information about an object. Sprites have properties that specify what image to draw, where to draw it and how to draw it.

PROPERTY CHIP

These items can be seen in the sprite timeline. Each one represents the properties for the sprite at that point in time. Placing more than one of these in the sprite timeline will allow the properties for the sprite to be changed. The changes applied by subsequent property chips are persistent until the next property chip that changes that property.

QSCRIPT

The scripting language used within LiveStage Professional to add interactivity to a movie. This language is compiled into a form that QuickTime understands and is thus limited by the capabilities of QuickTime.

QTLIST

A flexible data storage mechanism that is attached to QuickTime Movies and Tracks.

RATE

Specifies how fast the movie plays in multiples of its normal speed. The rate also indicates which way the movie plays. A negative rate will make the movie play backwards. A rate of -1 will make the movie play backwards at normal speed. A rate of 2.0 will make the movie play twice as fast, 0.5 half as fast.

REGISTRATION POINT

The location in an image that will be used to locate the image spatially in the sprite track. When a sprite uses an image, the location of the sprite coincides with the registration point of the image. This is used to align animating images, like a walking figure. If the registration point is placed on the foot of the figure that is on the ground, then the figure will appear to walk as the images are cycled through.

RELATIVE PATH

A Relative Path is made up of directions to a file that are based on the current file ‘ location. For example,”../media/Presentation.mov”is the “Presentation.mov”file located in a folder called “media” in the next folder above the current location. Relative paths are often used when files are on the same server, such as progressive download movies that are housed on the same server as the web pages that refer to them.

RTP

RealTime Transfer Protocol This transport protocol delivers live events to one or more viewers simultaneously. RTP is the transfer protocol for RTSP streaming.

RTSP

RealTime Streaming Protocol This protocol transmits true streaming media to one or more viewers simultaneously. RTSP also allows viewers to randomly access the stream.

SAMPLE

See media sample.

SAMPLE DURATION BAR

This is a blue bar that is shown in both the tracks tab and the sprites tab. It visually indicates the time that the sample is active. In the tracks tab it shows this for the selected sample only.

SCRIPT CHIP

These items can be seen in the sprite timeline. Each one represents the event handler scripts for the sprite at that point in time. Placing more than one of these in the sprite timeline will allow the event handler scripts for the sprite to be changed at that point in time. The changes applied by subsequent script chips are persistent until the next script chip that changes that event handler.

SCRIPT / PROPERTY WELL

This is an area to the left of the sprite time line. You can drag new script or property chips from here onto the time line.

SELECTION PROPERTIES PANEL

Displays information in the tracks tab about the currently selected track or sample.

SOURCE

Most properties also have a source popup menu in which an alternative method of determining the property can be chosen. When the popup indicates “none”, then the property can be manually altered by the user. Tracks and tweens that can be used to supply data for the property will show up in the popup menu and can be selected as the source of information for that property. For example, if you have a modifier track with integer data in it then you can use it as the source for the image index property of a sprite.

SPRITE

An object that can display an image and contain actions to be performed in response to user interaction. Sprites can move and change their image so as to produce animated sequences.

SPRITE BEHAVIORS TAB

This tab is visible in the sprite sample window when the sprites tab is selected and a sprite is selected in the sprites list. The behaviors of the sprite are shown and edited here.

SPRITES LIST

This list is visible in the sprite sample window when the sprites tab is selected. This list shows all the sprites stored in the sprite sample. Double clicking a sprite in this list will open a window allowing you to edit the properties of the sprite and to add QScripts and Behaviors to it.

SPRITE PROPERTIES TAB

This tab is visible in the sprite sample window when the sprites tab is selected and a sprite is selected in the sprites list. The properties of the sprite are shown and changed here.

SPRITE SAMPLE WINDOW

This window is shown when a sprite sample is double clicked. In this window you can add images and sprites to your sprite sample.

SPRITE SCRIPTS TAB

This tab is visible in the sprite sample window when the sprites tab is selected and a sprite is selected in the sprites list. The scripts for the various event handlers are shown and edited here.

SPRITES TAB

This tab is visible in the sprite sample window. Selecting this tab shows a list of sprites contained in the sprite sample.

SPRITE TIMELINE

The timeline for a sprite is visible in the sprites tab when a sprite is selected. It shows the time line for the sprite sample in which this sprite is contained. The property and script chips are shown in the timeline. A repository of blank chips is to the left of the timeline where new chips can be dragged out and placed in the time line.

SPRITE TRACK

A track in a movie that contains sprite media samples.

STREAMING

A generic term that is generally applied both to technologies that match the bandwidth of a media signal to the viewer 's connection, so that the media is always seen in realtime ("True Streaming"),as well as media which may be viewed over a network prior to being fully downloaded ("HTTP Streaming" and "Progressive Download").

TIME SCALE

The number of time units that pass per second in a time coordinate system. A time coordinate system that measures time in sixtieths of a second, for example, has a time scale of 60. Movies typically have a time scale of 600 since 60 frames per second and 24 frames per second can both be expressed as whole numbers in this time scale.

Example:	10 =10 minutes
	10: = 10 minutes
	10:1 = 10 minutes, 1 second
	:1 = 1 second
	10:1.100 = 10 minutes, 1 second, 100/600ths of a second

TRACK

A track is a container in a movie. Each track is independent of other tracks but all tracks are linked in time. When the movie is at a particular time, then all tracks are at that time. A track contains one or more samples of a single kind of media. All media samples in a track MUST be of the same type. The media does not have to actually be stored in the movie.

TRACK OFFSET

This is the time between the beginning of a movie and the beginning of a track 's data. In an audio track, the blank space would equate silence; in a video track, the blank space would mean no visible image.

TRANSFORMATION MATRIX

A 3 by 3 matrix that defines how to map points from one coordinate space into another. This can be used to scale, skew, rotate, distort or translate sprites or visual tracks. The matrix tells the sprite or track where to draw. Use this to position and distort a sprite.

TRUE STREAMING

Also known as RTSP streaming, this refers to the matching of the media signal ‘bandwidth to that of the viewer ‘s connection, so that the event is seen in real time.

TWEEN

Performs linear interpolation between values of various data types based on an algorithm. For instance, if a tween has a starting value of 1.0, an ending value of 2.0, and a duration of 1 second, then at 1/2 a second, the value returned from the tween would be 1.5.

TWEEN TRACK

A track in a movie that contains tween samples. A tween track can be used as the source for various object properties. They are typically used to smoothly alter a property, like the blend percentage of a sprite, or the sprite’s matrix (using a path tween).

VECTOR IMAGE

A mathematical description of an image that is more compact than a bitmap. Vectors also scale in size much better than bitmaps since each pixel is derived mathematically.

WIRED MOVIE

Wired movies have tracks that contain scripts that run in response to a user’s actions.

XML

XML stands for Extensible Markup Language, XML is a method to describe data and to focus on what the data is.

ZOOM IN/OUT BUTTONS

These buttons allow the user to change the scale at which the timeline is drawn. Pressing zoom in will show more detail, allowing precise placement of samples.

Index