Improvisational Theater for the Web!

totally hip™
Software Inc.

MACINTOSH

Thanks to Joachim Gola for input on using QuickTime movies with Macromedia Director

CONTENTS

Before You Begin

About This Guide

Please be sure to read the chapter on installation and other important information before you begin using LiveStage. This manual is organized by the menus and palettes, followed by the QScript.

This manual assumes a familiarity with the Windows Operating System and terminology. For complete instructions on the Windows Operating System see your Windows documentation.

System Requirements

Before installing any of the software, make sure the computer meets the minimum system requirements:

Power Macintosh

Mac OS 7.1 or later

QuickTime 3 or later, or QuickTime 3 Pro (included on this CD ROM)

5 MB of application RAM

15 MB of hard disk space for basic installation of all three applications

For the latest breaking news on LiveStage and WebPainter, please see the Read Me document on the LiveStage CD-ROM.

Installing LiveStage

1. Disable virus protection software and restart the computer.
2. Double-click the LiveStage icon.
3. Follow the installation instructions as they appear in the dialog box.
4. Accept the license agreement.
5. Enter your registration information.
6. Select the destination directory.
7. Launch LiveStage from the application icon.

The first time LiveStage is launched, enter the name, company's name and the product serial number. The serial number must be entered exactly as shown on the inside of the CD cover, with all capital letters and no spaces.

Installing WebPainter

1. Disable virus protection extensions and restart the computer.
2. Double-click the 'Install WebPainter 3' icon to launch the installer.
3. Follow the installation instructions as they appear on the screen.
4. Accept the license agreement.
5. After completing all the installations, restart the computer.
6. Double-click the WebPainter 3 application icon to launch WebPainter.

The first time WebPainter is launched, enter the name, company's name and the product serial number. The serial number must be entered exactly as shown on the inside of the CD cover, with all capital letters and no spaces.

Installing QuickTime and the QuickTime browser plug-in

1. Disable virus protection extensions and restart the computer.
2. Double-click the 'Install QuickTime 3' to launch the installer.
3. Follow the installation instructions as they appear on the screen.
4. After completing all the installations, restart the computer.
5. To receive a free upgrade to QuickTime Pro simply register LiveStage online at http://www.totallyhip.com/Link/ProductsRegister.html.
6. Once registered you will receive the QuickTime Pro unlock code and installation instructions.

Support

There is a list server available for all registered users of LiveStage. This list is monitored by the developers, and is designed as a forum to share solutions to problems, ideas, scripts and development suggestions. Please go to `http://livestage.listbot.com` and register to begin using this valuable resource.

For technical support regarding WebPainter, please contact `techsupport@totallyhip.com`.

Totally Hip Software on the Web

Totally Hip Software has been making award-winning Internet multimedia software since 1995. Our Web site is located at `http://www.totallyhip.com/`. Be sure to check our Web site on a regular basis for information on new products, updates to products, and special offers.

Registering LiveStage Online

In order for you to receive your free QuickTime Pro unlock code, you must register your software with us online. Registration also allows us to continue to provide you with the highest quality software, and keep you informed about our new software products, please register LiveStage online at:
`http://www.totallyhip.com/Link/ProductsRegister.html`

Chapter 1
LiveStage Basics

INTRODUCTION

Please note that this documentation for Version 1 (the Developer's Release) is still under revision. Further documentation will be forthcoming. Please join the LiveStage list at http://livestage.listbot.com for technical support, and visit http://www.totallyhip.com/ for more information as it becomes available.

LiveStage, from Totally Hip Software, is a powerful and extensive multimedia development tool designed for creating interactive Rich Media content for the Web and CD-ROM. It provides designers with tools to develop compelling content of the highest quality, and lowest-bandwidth. LiveStage exports interactive QuickTime Movies which are viewable by a large audience because QuickTime is the most popular web browser multimedia player technology for both Windows and Macintosh computers.

LiveStage creates engaging interactive presentations that encourage viewers to explore the designer's site. Adding interactivity with LiveStage is much more powerful and user friendly than creating JavaScript rollover effects or navigational image maps.

This User Guide teaches the fundamentals of using LiveStage. This includes how to create and assemble sprites, how to animate them on the Stage, and how to build interactive movies using QScript; the powerful multimedia scripting language built into LiveStage.

What is LiveStage?

LiveStage is based on QuickTime to bring cross-browser, cross-platform interactive multimedia to the Web. LiveStage enables designers to import artwork from their favorite tools, such as Adobe Photoshop and Totally Hip's WebPainter 3, to create the image content. Interactivity can be added using LiveStage's QScript multimedia scripting language. Soundtracks, MIDI files, and instrument sounds from QuickTime may be added, as well as interactively triggered sound effects.

The steps involved in creating a LiveStage project mirror the process of creating a Hollywood movie. Start by creating or gathering up multimedia elements, or objects. Objects in LiveStage include sprites, images, paths and sounds.

A project is made by creating sprites and assigning them in the Stage window. The Stage window allows the sprites within a project to be previewed.

A project plays out on the Stage. This is the background upon which sprites are animated.

What is WebPainter 3?

WebPainter 3 has all the tools needed to create professional Web animation and graphics. Its ease-of-use and unique cel animation interface creates the most compact, high quality GIF animations, QuickTime Movies, JPEGs and GIF files in the fewest possible steps. By combining bitmap and vector graphic tools, sophisticated image layering features, SMPTE special effects, auto-transitions, image filters, and powerful export features, users can quickly create professional animation and graphics for corporate or personal Web sites.

WebPainter 3 also comes with over 1,000 professionally designed, royalty free GIF animations.

The WebPainter interface and tools are extremely easy to use to create images and animated sequences.

1. Cel Paint Document Window

   This is where graphics are painted and drawn using WebPainter's paint and vector drawing tools.

2. Paint Tools Palette

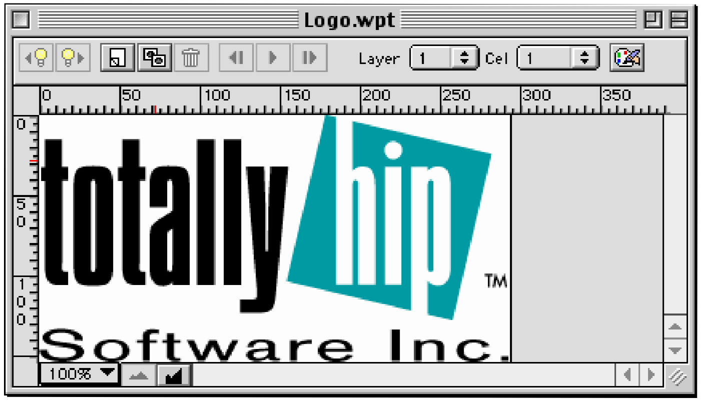
   This includes a number of bitmap painting tools to create your bitmap or pixel based images. To switch to the vector drawing tools click on the top right icon next to the Cel number pop-up in the top left corner of the Cel Paint Document Window.

3. Layers Palette

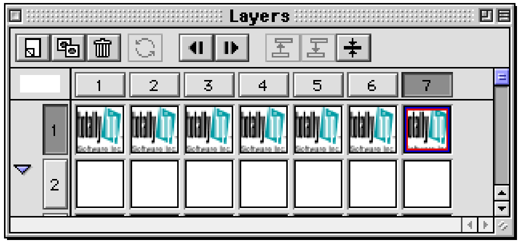
   Different layers that can be painted and drawn on in your animation. The Layers palette allows for easy navigation between the layers. WebPainter is a frame based animation program, and each frame has 32 layers. Layer 1 is the bottommost image. Each time a new layer is used, that image will be placed in front of the last one.

4. Colors

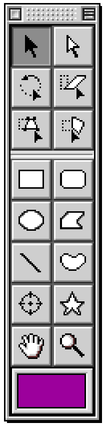
   Colors to paint on the cel document layers in bitmap or vector mode are selected from this palette.

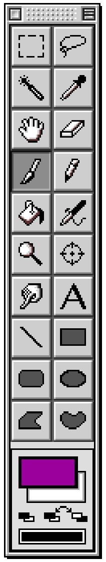


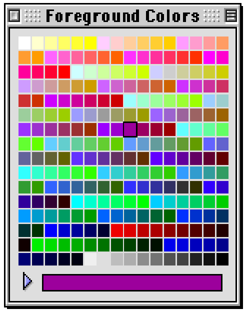
WebPainter 3 Cel Paint Document Window



Layers Palette


Vector Tools


Bitmap Tools



Color Palette

There are a number of other palettes and features in WebPainter. Please refer to the WebPainter Getting Started Guide and Manual to learn more about WebPainter. These are automatically installed with WebPainter and will be found in the WebPainter 3 folder from the Windows Start menu.

What is QuickTime?

With more than 11,500 CD-ROM titles using QuickTime, and over 100 new QuickTime-enhanced CD-ROM titles introduced each month, QuickTime is the industry standard multimedia format. It is the leading technology for digital video on the Internet today using both the QuickTime browser plug-in and the QuickTime MoviePlayer.

QuickTime is an enabling technology, and is comprised of pieces of software that extend the ability of the Macintosh or Windows operating system to handle dynamic media. Totally Hip designed and developed LiveStage to allow end-users to harness the power of QuickTime technology, and turn it into an interactive application development environment. The combination of LiveStage and QuickTime version 3 or greater, allows the user to build interactive content for CDs or the Internet such as games, interactive ad banners, or training materials.

QuickTime is much more than just video and sound. It is a true multimedia architecture that allows the integration of text, still graphics, video, animation, 3D, VR and sound in a cohesive platform. LiveStage and QuickTime make it easy to bring these media types together in an interactive, compelling experience on the Web using the QuickTime browser plug-in.

Totally Hip will be continually evolving the capabilities of LiveStage, allowing the creation of more applications and rich media content for Web sites or CD-ROMs.

What is the QuickTime Plug-In?

Once interactive movies are created with LiveStage, they can be played back on the Web through the QuickTime Plug-in.

Apple's QuickTime Plug-in for Netscape Navigator and Microsoft Internet Explorer lets the user experience QuickTime content including QuickTime VR panoramas directly in the browser window.

QuickTime with the web browser plug-in for Mac OS, and Windows 95/98/NT can be downloaded for free from:

```
http://www.apple.com/quicktime/download
```

What is QuickTime Pro and MoviePlayer Pro?

QuickTime Pro gives the user full access to the power of QuickTime by expanding the powerful editing capabilities of QuickTime. MoviePlayer is the application that comes with QuickTime, which allows opening and play back of dozens of different QuickTime content types. The QuickTime Pro set of tools includes the fully featured MoviePlayer, a more robust version of the PictureViewer tool, and a more powerful QuickTime Plug-In.

Totally Hip is providing a free upgrade to QuickTime Pro and MoviePlayer Pro when LiveStage is registered. This is a direct saving of $29.95! Once registered, we will send the upgrade key for QuickTime Pro and MoviePlayer with installation instructions.

What is a Sprite?

A sprite is like an actor performing on a stage. It can move, change appearance and interact with other characters in the play. A sprite is similar to a castmember in Director™, the authoring tool from Macromedia™. For many kinds of animations, a QuickTime sprite track provides a much more compact representation of the animation than a traditional video track.

In LiveStage, sprites can either be a bitmap image importable by QuickTime, such as a GIF or Photoshop file, or they can be a vector image in the QuickTime Image File format, which is exportable by WebPainter 3. Vector images are preferable to bitmap images because they are commonly several orders of magnitude smaller, as well as resize much more smoothly.

A sprite has properties that describe its location and appearance at a given point in time. During the course of an animation, a sprite's properties can be modified to change its appearance and to move it around the stage.

Each sprite has a corresponding image. During the animation, a sprite's image can be changed. To perform cel-based animation, for example, a series of images can be assigned to a sprite in succession.

A sprite may have handlers written in QScript which tell the sprite how to react to user input, time passing, or other events.

Editing of properties, images, and handlers for a sprite is easily accomplished in LiveStage from the Actions tab of the Objects window.

What is Sprite Animation?

Sprite animation differs from traditional video animation. With traditional video animation, a frame is described by specifying the color of each pixel. By contrast, with sprite animation, a frame is described by specifying which sprites appear at various locations. At a given moment a sprite displays a single image selected from a pool of images shared by all the sprites.

What is a Track?

QuickTime movies have the data within them oranized into tracks. In a conventional movie, there is one video track and one audio track. However, QuickTime is capable of displaying a large and continually growing variety of track types. A single movie can contain multiple tracks of any particular type.

Quicktime stores the information about each sprite in a sprite track. As the movie plays, QuickTime uses the information in the sprite track to draw the sprites. Other track types may be in the movie as well. LiveStage allows sound-tracks and music (MIDI) tracks to be added directly, while sound files or QuickTime instruments can be added to an Instruments track to be used as sound effects triggered by sprites.

LiveStage also allows QScripts to be written which control other track types, such as 3D, VR, and text. An example of how to use MoviePlayer to add these other track types into a movie exported by LiveStage is in the 'X2K' project, which can be found in the Samples folder installed with LiveStage.

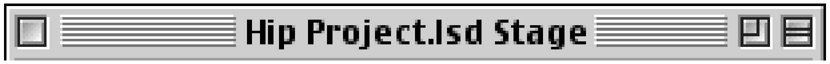
Chapter 2
The LiveStage Interface

The LiveStage interface is comprised of floating palettes and a standard Macintosh menu bar. These palettes can be moved around the desktop area, or opened and closed as desired to allow the user to build the layout they prefer.

To show or hide any of the palettes check the corresponding palette name from the Window menu. Alternately, use the shortcut commands to open or close the the following items.

Object Window: Command + B
Media Library: Command + L
QScript Reference: Command + T
Last Active Window: Command + 1
2nd Last Active Window: Command + 2
3rd Last Active Window: Command + 3

All windows and palettes have the standard Mac OS Close, Zoom, and Collapse options within the Title Bar.

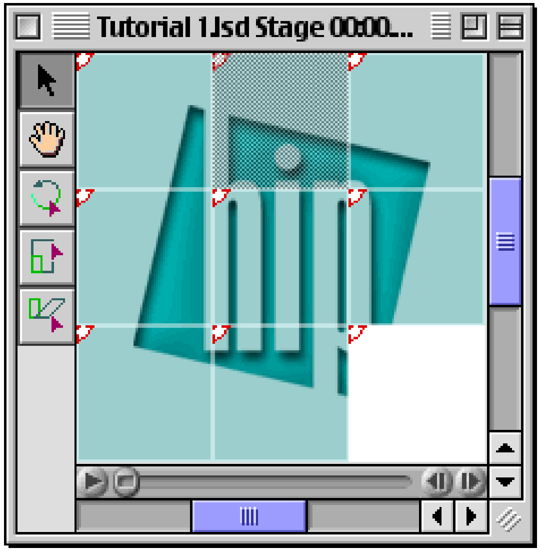


Close *Title* *Zoom* *Collapse*

MENU BAR

There are six standard menu choices available in the LiveStage menu bar: File, Edit, Path, Time, Window, Help. (See Menu Overview later in this chapter)

STAGE WINDOW



The Stage Window shows an approximate overview of the movie at a particular time. That time is shown in the Stage title bar in minutes, seconds, and hundredths of seconds if the animation is set to be timed.

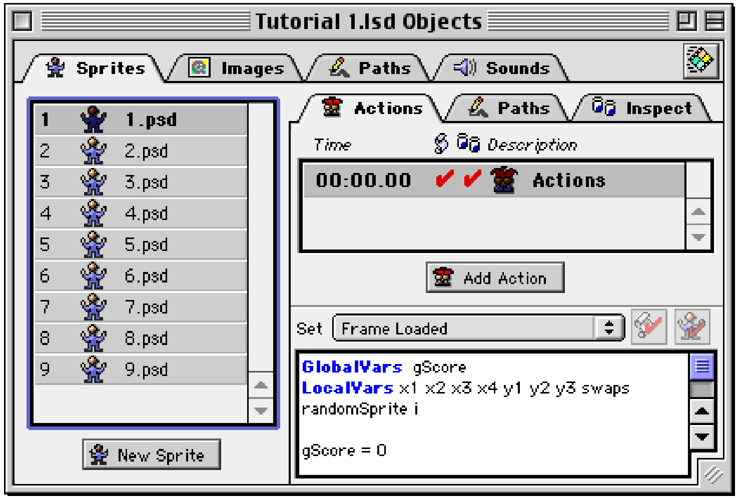
It is also the place where images are placed form the Media Library to create sprites. Once they have been dragged here from the Library, they will appear in the Images and Sprites tabs within the Object Window for further manipulation.

The Time Controller is the horizontal bar at the bottom of the Stage, just above the horizontal scroll bar. It has a Play/Pause button on the left to preview the motion, as well as step forward and backward buttons on the right. The bar contains a floating icon which indicates the progress of the project along the determined time line. The movie controller allows the user to control the editing time within the interactive movie.

By default, the Time Controller is not visible. To add it to the stage window, select it from the Window menu.

The Stage shows the movie's sprites at their current location with their registration points. The sprites can also be arranged, rotated, scaled, or skewed on the Stage.

OBJECT WINDOW

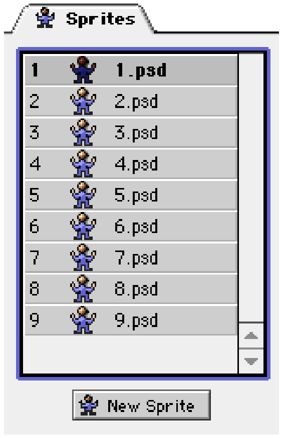


```
GlobalVars gScore
LocalVars x1 x2 x3 x4 y1 y2 y3 swaps
randomSprite i

gScore = 0
```

The Object Window is the palette from which images, paths and sounds can be incorporated into sprites. It is also where Paths, and Sounds are created, and Sprites can be inspected. The palette consists of four primary tabs along the top: Sprites, Images, Paths and Sounds, and 3 secondary tabs below them when the Sprites tab is selected. When clicked on, each primary tab activates a corresponding editor or inspector pane below it, where attributes for each item may be added, or modified.

The following section is broken down by the primary tabs with the corresponding panes that they activate.

Sprites Tab



Sprites can be referred to within a script in three ways: by Sprite Index, Sprite Name, and Sprite ID. LiveStage uses these references to know which variables to assign to which sprites.

The Sprite Index is a numerical value that is automatically assigned when a sprite is created. It is useful when creating script that involve randomness (see Tutorial 1). Using this format contains a potential drawback when creating other types of scripts. If, for example, the Object Window contained sprites numbered 1-4, and sprite number 1 was deleted, the three remaining sprites would drop one number in value. The result would show the sprites now being numbered from 1-3. This can result in broken or malfunctioning scripts.

Sprite Name and Sprite ID differ from Sprite Index by the fact that they do not change automatically, even if the order of the sprites in the project is modified.

The Sprite Name and Sprite ID can be modified in the Inspect tab.

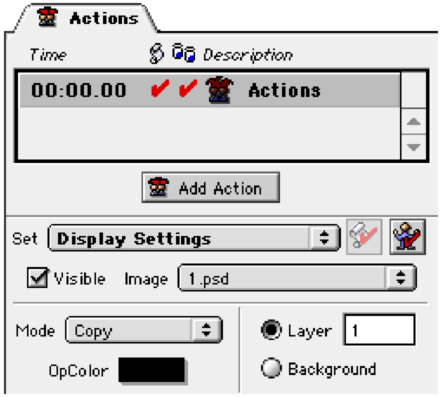
To add a sprite to a project, drag an image from the Media Library to the Stage. A new Sprite with the original image file name will appear in the Sprite tab of the Object window.

Alternately, click New Sprite to add a sprite to the LiveStage document. Ensure that there is at least one image in the Images tab of the Object window. Select the sprite, then select Display Settings from the Set pop-up list in the Actions tab of the Sprites edit pane. From there, select the image to assign to the selected sprite from the Image pop-up menu. (For other options see Actions tab).

To remove a sprite from the list, select it and click Delete on the keyboard. Selecting the Sprites tab activates the sprites editing pane.

SPRITE EDITING PANE

Actions Tab



00:00.00 ✓ ✓ Actions

Set: Display Settings

✓ Visible Image: 1.psd

Mode: Copy

● Layer: 1

Time: 00:00.00 indicates the time of the action to 1/100th of a second.

Script icon: A red check will indicate if there is a script associated with the action.

Visible icon: A red check will indicate if the sprite is visible (default). This is changed in the Display Settings.

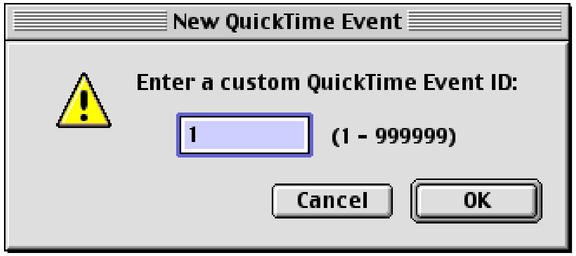
Description: The Action # is assigned automatically to each new action that is created. They can be deleted by slecting the desired action, then clicking Delete Action.

Add Action: Click this button to create a new Action for the selected sprite. An Action denotes a fixed moment in time that selected Events will occur at. Once this is clicked, select the Event to occur after the Action from the Set drop-down list.

This will prompt an error message if the animation is untimed, or if the last action in the list is at the end of the project. If this is the case, either set the project to be timed, or add more time to the movie as needed from the Document Settings dialog from the Edit menu.

Set: This drop-down list carries standard Events, Custom Events, Image Transforms and Display Settings.

If an Action is selected, enter the desired script to be associated with it into the pane below. The script can be dragged from the QScript Reference or typed.



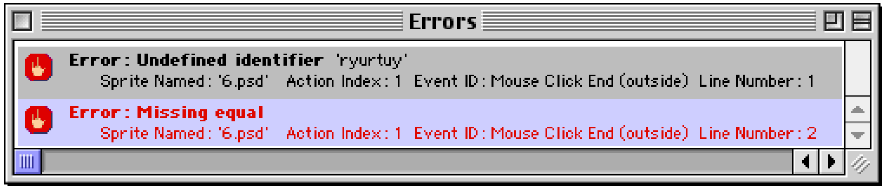
Enter a custom QuickTime Event ID:

1 (1 - 999999)

Add Custom Event: Select this to add an event that is not in the list. A dialog will appear prompting input for a number for the new QuickTime event. This value can be from 1-65535.

CHECK SYNTAX

Once an Event is completed, click the Check Syntax button. If there are errors in the syntax of the script, a dialog will appear indicating which portions of script are incorrect.



Errors

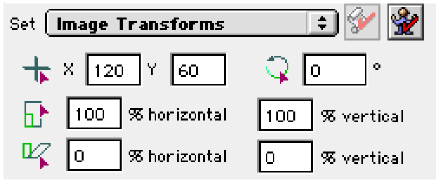
Error : Undefined identifier 'ryurtuy'
Sprite Named : '6.psd' Action Index : 1 Event ID : Mouse Click End (outside) Line Number : 1

Error : Missing equal
Sprite Named : '6.psd' Action Index : 1 Event ID : Mouse Click End (outside) Line Number : 2

ERROR DIALOG

Image Transform: This allows the following options to be specified for the current sprite:

- X,Y position in pixels
- rotation in degrees
- horizontal and vertical scale in percentages
- horizontal and vertical slant in percentages



Set Image Transforms

X 120 Y 60 ; 0 °

100 % horizontal ; 100 % vertical

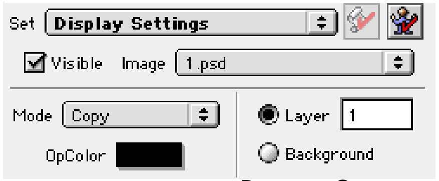
0 % horizontal ; 0 % vertical

IMAGE TRANSFORMS

SYNCHRONIZE STAGE

After using an Image Transformation, click the Synchronize Sage button to update the changes to the Stage.

Display Settings: This allows the following options to be specified for the current sprite:



Set Display Settings

☑ Visible Image 1.psd

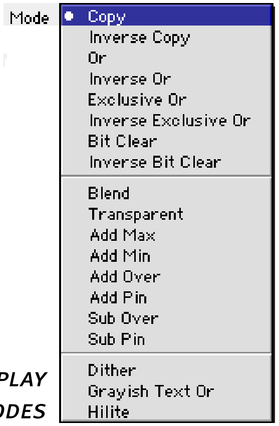
Mode Copy ; ● Layer 1

OpColor ■ ; ○ Background

DISPLAY SETTINGS

Visible: Sets the image to be visible when checked.

Image drop-down list: Select the image to assign to the current sprite. To add an image to this list, it must first be imported from the Media Library to the Images tab of the Object window (see Images tab in the Object window).



DISPLAY MODES

Mode: Boolean, Arithmetic, and Miscellaneous modes can be found in the pull-down menu. (For descriptions of each mode, see Modes in Adding QScripts to a Sprite Object)

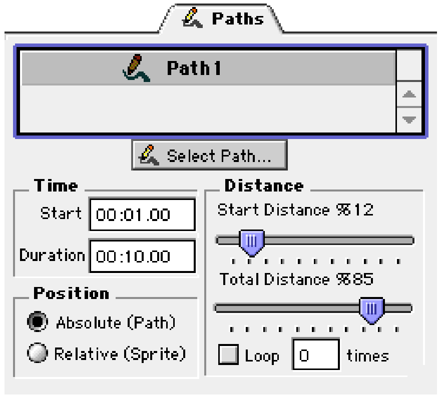
OpColor: This is the color in the sprite image that the Mode processes.

Layer: This indicates the layer the sprite is drawn in.

Background: This radio button will ensure the current sprite is on the bottom layer to appear as the background.

After changing the Display Settings, click the Synchronize Sage button to update the changes to the Stage.

Paths Tab



Start 00:01.00
Duration 00:10.00
Start Distance %12
Total Distance %85
Position: Absolute (Path)
Loop 0 times

Once a path has been created (see Paths tab in the Object window), associate it with a sprite by selecting the sprite, then clicking the Select Path button. Once it has been selected, input the values for the following options:

Time

Start: 00:00.00 up to 1/100th of a second. This is when the sprite will begin to follow the path.
Duration: This is the total amount of time it will take the selected sprite to move along the selected path.

Position

Absolute (Path): Indicates the sprite is to follow the points in the path directly.

Relative (Sprite): Indicates the sprite is to follow the points in the path as offsets to its position.

Distance

Start Distance: This amount indicates where on the path (as a percentage) the current sprite begins.

Total Distance: This is the total amount of the path (as a percentage) the current sprite travels.

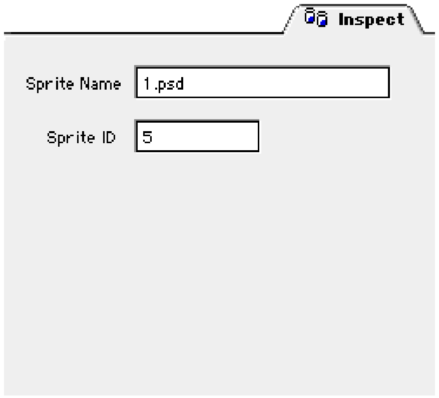
Loop: Check this to have the current sprite repeat its movement along the path. The default is 0. To specify the number of times the sprite will loop along the path, enter the number in the input field.

Inspect Tab

Sprite Name: Indicates the name associated with the current sprite.

Sprite ID: Indicates the ID associated with the current sprite.

Each of these sprite identifiers can be changed by typing new values into their corresponding fields. These will be updated globally in the project once the mouse is clicked outside of these fields.



Sprite Name: 1.psd

Sprite ID: 5

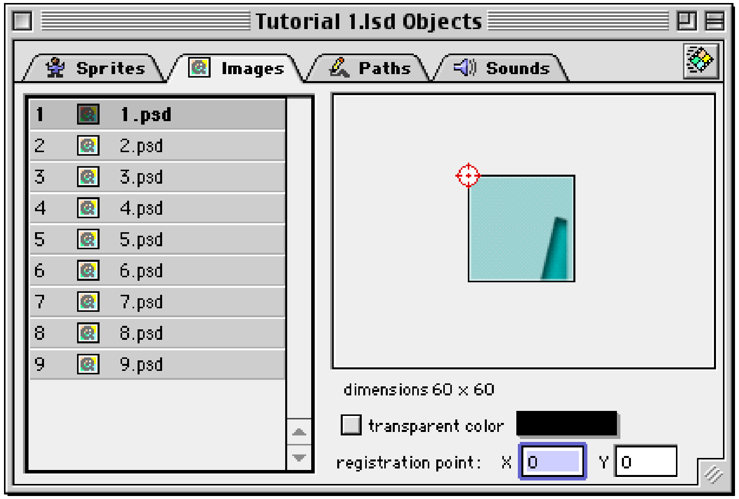
Images Tab

This is where images are added from the Media Library to the current project where they can then be assigned to a sprite (see Sprites tab in the Object window). This can be done by dragging the image icon from the Media Library, and dropping it within the list in the Images tab of the Object window. Images are automatically assigned an ID number as they are added.

Image inspector pane

The Image inspector pane displays the image centered, with a registration point. The registration point is a red crosshair within a circle designed as an alignment tool. It can be manually moved into place by click-dragging, or by entering the exact coordinates in the X,Y coordinates boxes below the image pane.

The Image Inspector also displays the image dimensions in pixels, and allows for the option and selection of a transparent color.



1	1.psd
2	2.psd
3	3.psd
4	4.psd
5	5.psd
6	6.psd
7	7.psd
8	8.psd
9	9.psd

dimensions 60 x 60

☐ transparent color

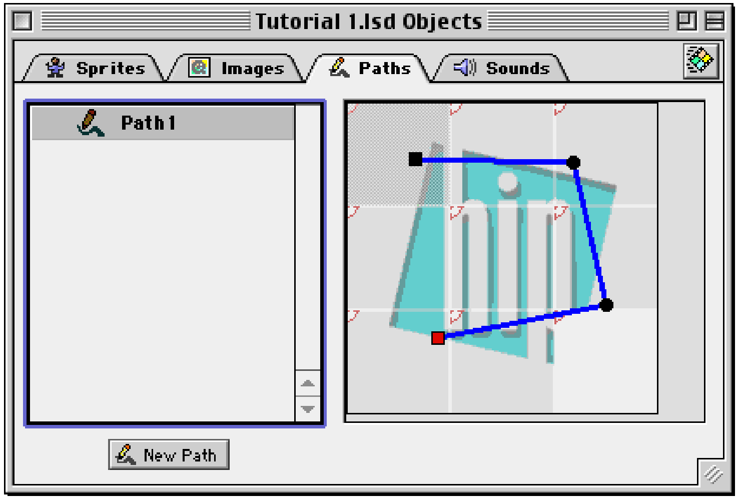
registration point: X 0 Y 0

Paths Tab

A path is a line made up of multiple points that a sprite can follow around the Stage. To create a path for the current project click the New button.

Path edit pane

Selecting the Paths tab activates the Paths Editor.

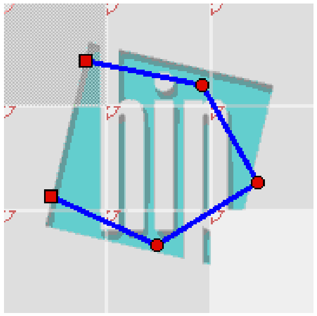


This window shows a preview image of the Stage window as a reference.

To create a path: Click the start point of the path on the Stage preview. A point will appear where the click was made. Click the next position on the path and another point will appear. A line will also appear, connecting the two points. Repeat this until the path is completed as desired. To adjust the position of a point, simply select it and drag it to the new location.

The following options may be chosen from the Path menu or from key strokes.

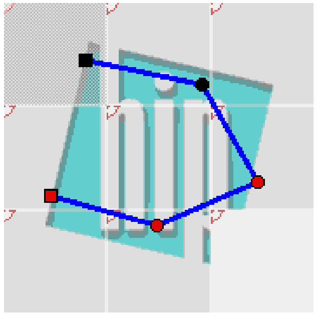
Distribute Points: Select multiple contiguous points by Shift-clicking on them. As they are selected, they will appear red. Distributes the selected points evenly within the selected area of the path.



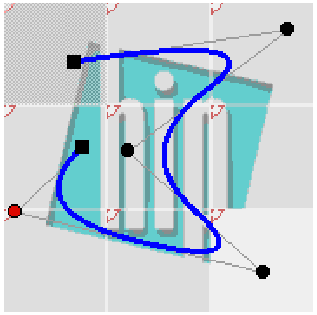
Distributing points evenly along a path

Smooth Points: Select multiple contiguous points by Shift-clicking on them. As they are selected, they will appear red. Smooth Points will reduce angles at the points.

Closed Path: Closes an open path or opens a closed path.

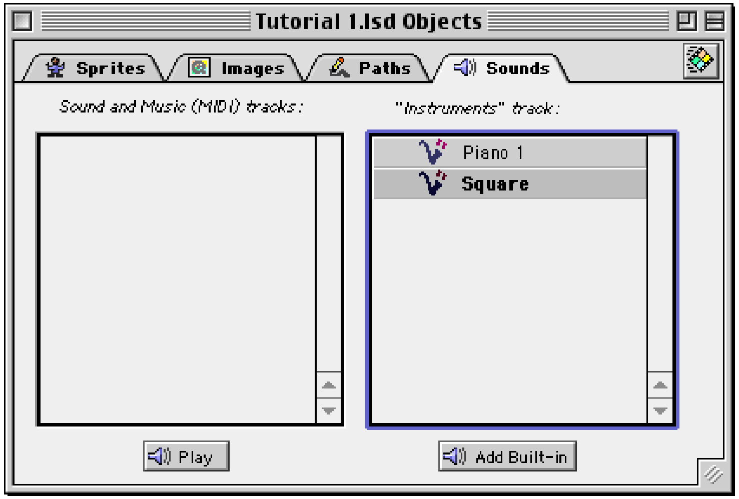

Smoothing points along a path

Point on Curve: On multiple selected contiguous points, this will place the path off the actual point, and curve it. The point will still control the path, but more in the way a spline does in a vector-based graphics program.
Shortcut: OPTION+ SHIFT


Point on Curve produces a control point

Sounds Tab

This is where sounds are added to the current project. To add a sound to the Object window, either drag and drop it from the Media Library, or click the Add Built-in button. If you select Add Built-in, the Instrument Picker dialog will appear.



INSTRUMENT PICKER DIALOG

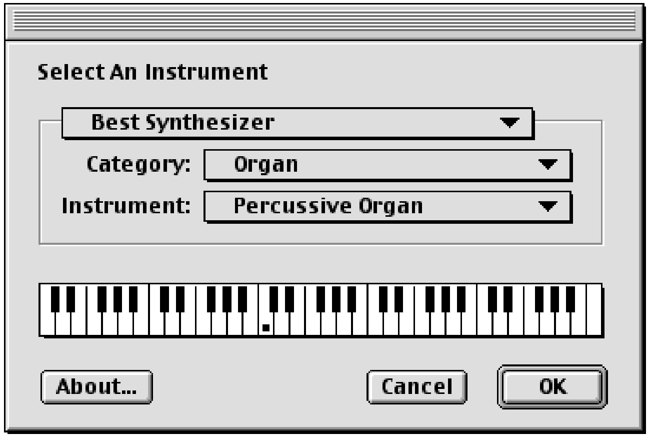
If Best Synthesizer is chosen:

Choose: Category

Choose: Instrument

Once a selection has been made, it may be tested by 'playing' the keyboard image.

Click OK


Select An Instrument

Best Synthesizer

Category: Organ

Instrument: Percussive Organ

If QuickTime Music Synthesizer is chosen:

Choose: Category

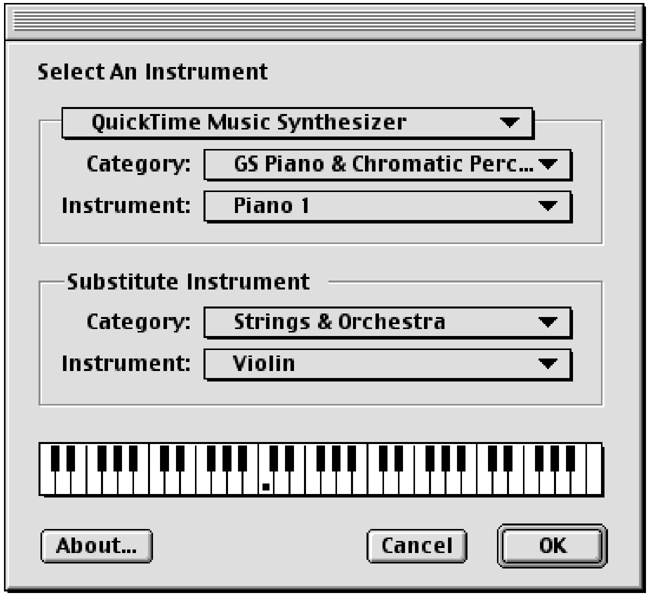
– Use Substitute Instrument

Choose: Category

Choose: Instrument

Once a selection has been made, it may be tested by 'playing' the keyboard image.

Click OK


Select An Instrument

QuickTime Music Synthesizer

Category: GS Piano & Chromatic Perc...

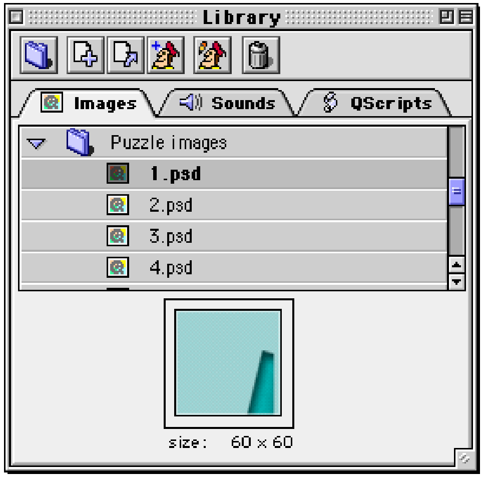
Instrument: Piano 1

Substitute Instrument

Category: Strings & Orchestra

Instrument: Violin

MEDIA LIBRARY



The Media Library is a convenient way to view and access media assets, allowing the user to drag and drop objects into a movie project. They may be inserted into the Action pane by dragging and dropping them into the window. The assets in the Media Library do not get stored within the movie project. When using an alias of the the original file, the Media Library can be used to access items on media storage devices such as a hard disk, CD-ROM, a Server, etc. This saves on project file size, and allows the most flexible way of accessing media assets to include in a movie project.

Note:
Always be sure that the source file or alias is placed in the Library folder.

The Library folders may be opened or closed by either clicking on the arrow to the left of the folder icon, or by double-clicking the folder icon itself.

New Folder

Folders of images, sounds, and QScripts can be stored in the Library folder so they can be used in any movie project. Library objects can be created, modified, and deleted at any time from within the Library window.

Add Item

The assets in the Media Library do not get stored within the movie project. Instead, they are kept independantly so they can be accessible to any project being created, at any time. Select the media tab (Images, Sounds, Scripts) and either the desired destination folder, or click on the folder icon to add a new one. Using the Add Item button, select the new file type. If it is a valid file type, it will appear in the current folder.

Add Alias

As with Add Item, Add Shortcut makes files accessable from the Media Library. In this case, however, it is working with a shortcut to a file that is stored elsewhere on the computer. Ensure the shortcut is placed in the Library folder within LiveStage.

If the item in the Library is a shortcut, the path to the original file can be shown by Control-clicking on it.

Create a New Image

LiveStage uses WebPainter 3 as it's default image and sprite editor, but library images can be created in many different image programs, then stored in the Library.

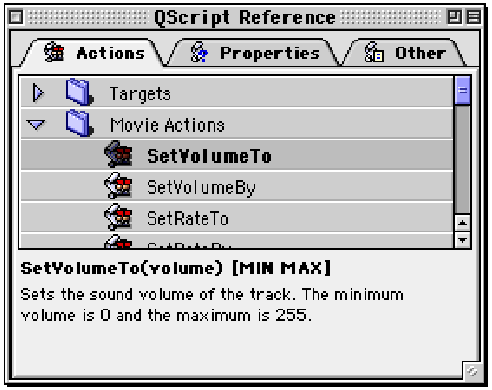
Edit Current Image

If an image needs to be modified while a project is being developed, select it in the Image tab and click the Edit Current Image icon. LiveStage will launch WebPainter and open the selected image file.

Delete Item

To remove an item from the Library, select it, then click the Delete Item icon button.

QSCRIPT REFERENCE PALETTE



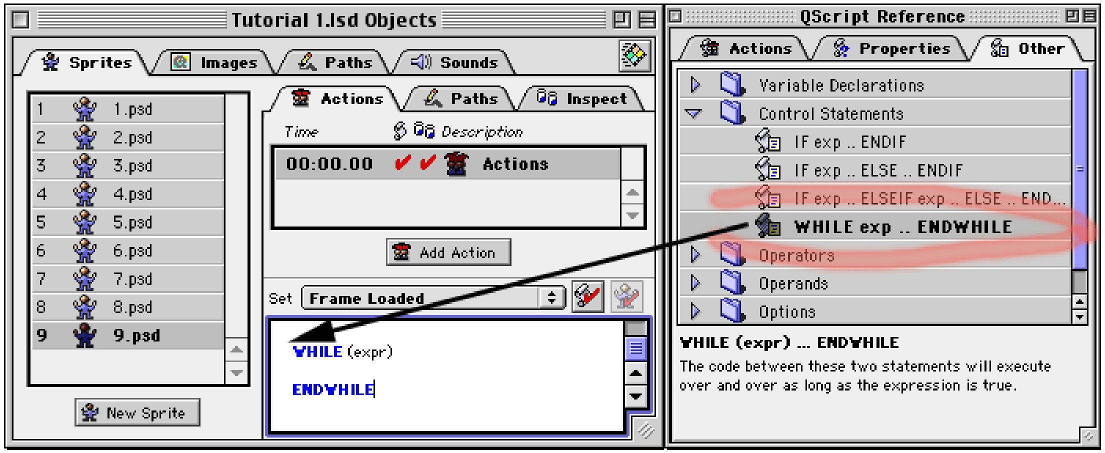
SetVolumeTo(volume) [MIN MAX]

Sets the sound volume of the track. The minimum volume is 0 and the maximum is 255.

This palette contains all predefined QScripts. It provides a reference to more than 120 statement types which are split into three main categories: Actions, Properties and Other. A QScript statement is any combination of these elements that make up a complete line of code within the LiveStage QScript scripting pane.

To view a QScript in the Reference window click on the folder that you want to find a statement or command in, and then click on the command in the list. These QScripts reference statements can be dragged and dropped into the QScript scripting pane to avoid typing in the QScript. Each time a QScript reference is clicked on in the QScript Reference Palette, the bottom of the window will provide some help on using the command in QScript.

QScript items may be inserted in the Action pane by dragging and dropping them. The QScript folders may be opened or closed by either clicking on the icon to the left of the folder icon, or by double-clicking the folder icon itself.

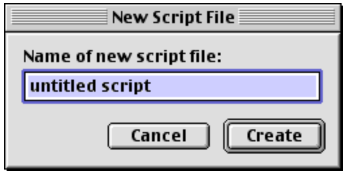


```
WHILE (expr)

ENDWHILE
```

WHILE (expr) ... ENDWHILE

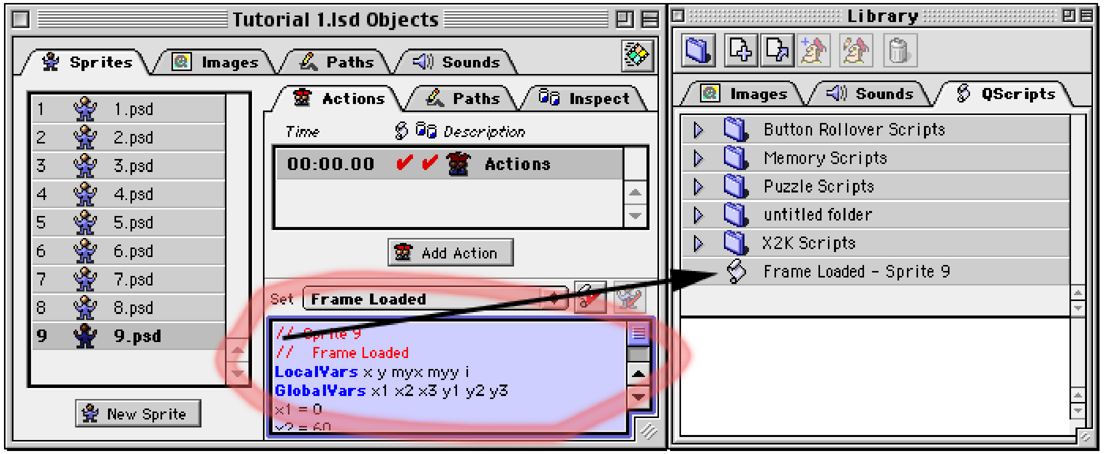
The code between these two statements will execute over and over as long as the expression is true.

Adding QScripts from the Reference palette



New Script File dialog

To add a custom script to the Library, select it from the Action pane, then drag and drop it to the desired QScript folder within the Library. A dialog will appear asking for the name of the new script being created.



```
// Sprite 9
//  Frame Loaded
LocalVars x y myx myy i
GlobalVars x1 x2 x3 y1 y2 y3
x1 = 0
x2 = 60
```

Adding custom QScripts to the Reference palette

MENU OVERVIEW

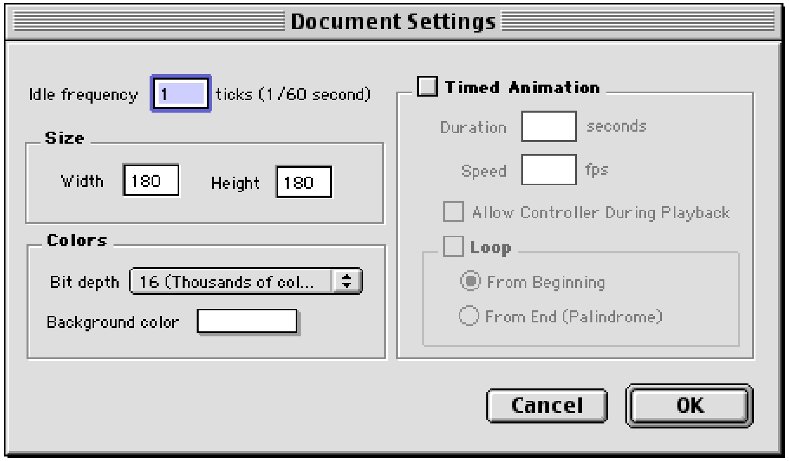
There are six standard menu choices available in the LiveStage menu bar: File, Edit, Path, Time, Window, and Help. Menu selection options can change depending on which LiveStage window or palette is active.

File Menu

The File menu selections such as New Project, Open, Close, Save, Save As, Revert, Run Wired Movie, Export Wired Movie, and Quit, apply to the LiveStage project as a whole. Some of these selections have icons associated with them in the toolbar.

NEW PROJECT

When this menu item is chosen, the Document Settings dialog appears allowing specific settings for the new project to be defined. (See Document Settings under Edit Menu)
Shortcut: Command+N


Document Settings

Idle frequency: 1 ticks (1/60 second)

Size
Width: 180
Height: 180

Colors
Bit depth: 16 (Thousands of col...
Background color:

[ ] Timed Animation
Duration: seconds
Speed: fps
[ ] Allow Controller During Playback
[ ] Loop
[● From Beginning]
○ From End (Palindrome)

OPEN

In the Open dialog box, select the project file, and click the Open button, or double-click the movie project name. The Open dialog box will display projects created with LiveStage.
Shortcut: Command+O

CLOSE

Close the current project. You will be asked if you want to save changes.
Shortcut: Command+W

Save

Save changes to the current project. This will write over the previously saved version.
When a project is saved for the first time, LiveStage produces a Save As dialog box, where a name and location can be assigned to the project file.

Files should be saved often to preserve the most recent work in case of power loss or system crash, etc.

Shortcut: Command+S

Save As...

Save the current project with another name.

Revert

Revert the current project to the last saved version of the current project. This is useful if some undesirable changes have been made to the current version of the project.

Run Wired Movie...

Compile the current project and run it. If there are no QScript errors reported, the movie file will run in a separate window on top of the LiveStage windows and palettes. If there are script errors in the project, the Scripts Errors dialog will appear notifying the user of the scripts that have errors in them. Any script errors must be corrected before a project can compile properly and run.
Shortcut: Command+R or click the Run Wired Movie button in the Objects window

Export Wired Movie...

Export the compiled project file to disk. After choosing Export Wired Movie..., the movie will compile into a stand-alone movie file if there are no errors in the movie project's QScripts. The movie name will appear in the Save dialog and the .mov extension will be appended to the file name.
Shortcut: Command+E

Quit

Quit the application. If the project(s) need to be saved, the program will prompt the user, and allow changes to be saved.
Shortcut: Command+Q

Edit Menu

The Edit menu provides standard editing commands such as Copy, Cut, Paste, Clear, which can be applied to the script editing pane in LiveStage. The Edit menu also allows the user to change the current movie document settings and preferences. If a command can be undone, the Undo and Redo menu items will be available. If the command cannot be undone, these options will be greyed out. The Edit menu commands vary depending on which LiveStage Window is currently active.

Undo

Undoes the previous action or command. There are 10 levels of Undo in LiveStage. It is not available for all items.
Shortcut: Command+Z

Redo

Repeats the previous action or command. It is not available for all items.
Shortcut: Shift+Command+Z

Cut

Cut an object or some text from the Script pane to the clipboard. This can only be applied to script editing.
Shortcut: Command+X

Copy

Copy an object, or text from the QScript pane to the clipboard. This can only be applied to script editing.
Shortcut: Command+C

Paste

Paste an object, or text in the QScript pane from the clipboard. This can only be applied to script editing.
Shortcut: Command+V

Clear

Clear a selected object or selected text. This can only be applied to script editing.
Shortcut: Del

SELECT ALL

Select all of the objects, or all of the text in the QScript pane. This can only be applied to script editing.
Shortcut: Command+A

DUPLICATE

Duplicate or make a copy of the selected object or text in a QScript pane. This can only be applied to script editing.
Shortcut: Command+D

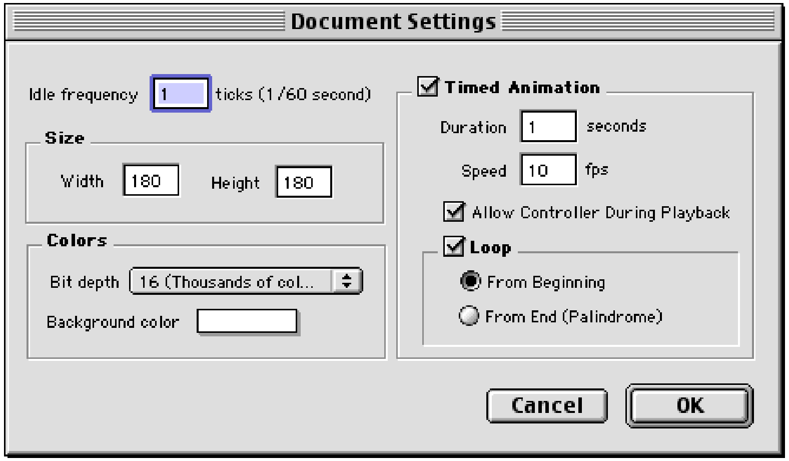
DOCUMENT SETTINGS...

Bring up the current project Document Settings Dialog. Document Settings are typically set when a new LiveStage project is started. Periodically, global settings may need to be adjusted. In either instance, this is done from the Document Settings Dialog. This dialog will appear automatically when a new project is started, or it can be selected from the Edit menu.
Shortcut: Command+I

Document Settings Dialog

Idle Frequency

The minimum time interval between idle events. The default idle frequency in a LiveStage project is 1 tick, or 1/60 of a second. If set to 2, it would be 2/60 of a second between idle loop events.



Document Settings

Idle frequency 1 ticks (1/60 second)

Size: Width 180 Height 180

Colors: Bit depth 16 (Thousands of col... ; Background color

[✓] Timed Animation: Duration 1 seconds; Speed 10 fps

[✓] Allow Controller During Playback

[✓] Loop: [● From Beginning] ○ From End (Palindrome)

Timing

Time Base: 1/600 of a second

1 Tick: 10/600 (or 1/60) of a second set in milliseconds. The default is 1 millisecond.

Size

Choose the width and height dimensions of the Stage area in pixels.

Colors

Bit Depth: From the drop-down menu, select the desired bit depth for the project. This is the number of bits used by the computer to store information for each pixel on a computer screen. The default is 16-bit (thousands of colors). Remember that the higher the bit depth, the larger the file size will be.

Background Color: Click on the color box and select the desired background color from the color picker. The default is white.

Important Note: Changing from a static (non-timed) movie to a timed movie within a partially completed project can have unpredictable results. It is recommended that this be decided prior to beginning the project.

Timed Animation

Check the box to make the project timed. This will activate the following options. The default is unchecked.

Duration: This is the duration of the entire project in seconds. The default is 1 second.

Speed: This is the number of frames per second (fps) in the project. The optimal range for smooth movement is between 15 and 30 fps. Bear in mind that this setting is the key factor in the file size of the finished movie file. If this project is being downloaded in a browser, try to balance the fps with the image quality. The default is 10 fps.

Allow Controller During Playback

By checking this box, the finished movie will have playback controls incorporated along the bottom of it.

Loop

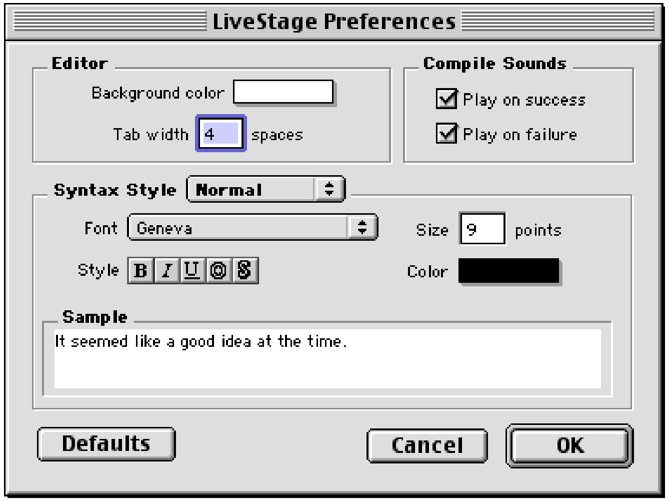
From Beginning: This option will have the movie loop indefinitely, starting at the beginning each time.

From End (Palindrome): This option will have the movie loop indefinitely, forwards from the beginning, then backwards from the end.

Preferences...

To modify general preferences for LiveStage, go to the Edit menu and select Preferences. The LiveStage Preferences dialog will appear.

LiveStage Preferences Dialog


LiveStage Preferences

Editor
Background color
Tab width 4 spaces

Compile Sounds
Play on success [✓]
Play on failure [✓]

Syntax Style: Normal
Font: Geneva
Size: 9 points
Style: B I U
Color

Sample
It seemed like a good idea at the time.

Defaults | Cancel | OK

Compile Sounds

Check Play on Success or Play on Failure to hear sounds to indicate whether or not a script has compiled successfully. The default has both options checked.

Editor

This is where the background color of the QScript editor window, and the amount of Tab width spaces are set.

Background Color: Click on the box and a color picker will appear. Select the desired Editor background color from the palette. The default color is white.

Tab Width: Indicate the number of spaces for tab width. The default tab width is 4 spaces.

Syntax Style

This indicates the formatting of the different syntax used by LiveStage. From the drop-down box, select the Syntax Style to be modified: Normal, Reserved, Comments, Hilited. After selecting the Syntax, customize it by clicking the Font button, then choosing the font, style, size, and color . A preview of the new Syntax Style will appear in the Sample pane.

- Normal is defaulted to Geneva 9-point plain black text.
- Reserved are defaulted to Geneva 9-point bold blue text.
- Comments are defaulted to Geneva 9-point plain red text.
- Hilited is defaulted to Geneva 9-point bold red text.

Path Menu

Paths are defined routes that a sprite can move around the Stage. If only one path point is selected, the Path menu will only have the Closed Path menu command available and the other two items will be dimmed and not available. If more than two path points are selected in the path, the Distribute Points, Smooth Points, and Closed Path menu commands will be available. These three features are also available as buttons within the Path Editor.

Distribute Points
Select multiple contiguous points by Shift-clicking on them. As they are selected, they will appear red. Distributes the selected points evenly within the selected portion of the path.

Smooth Points
Select multiple contiguous points by Shift-clicking on them. As they are selected, they will appear red. Smooth Points will reduce angles at the points.

Closed Path
Closes an open path or opens a closed path.

Point on Curve
On multiple selected contiguous points, this will place the path off the actual point, and curve it. The point will still control the path, but more in the way a spline does in a vector-based graphics program.
Shortcut: Option+Shift

Time Menu

The Time menu is only available when a project that is a timed animation has been created.

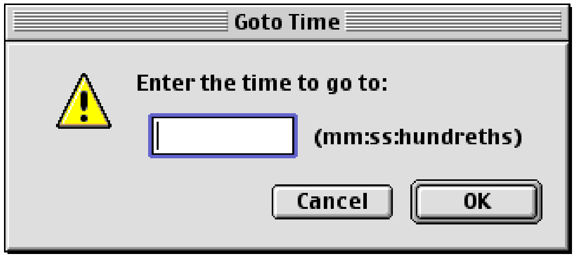
Go to Start
Go to the start of the movie, or the time of 00:00:00.

Go to End
Go to the end of the movie or the last time displayed before the movie ends or loops.

Go to Previous Frame
Go to the previous fame in the current movie project.

Go to Next Frame
Go to the next fame in the current movie project.

Go to Time...
Allows a specific time in the format of 00:00:00 (minutes/seconds/hundredths of seconds) to be entered from this dialog.


Goto Time

Enter the time to go to:

(mm:ss:hundreths)

Window Menu

The Stage and palettes within LiveStage can be turned on and off from within the Window menu. From this menu, there are three layout options available; Stack, Tile Vertical, Tile, and Zoom.

Stack
Stack open windows and palettes from the top left corner of the client area towards the lower left of the screen.

Tile
Arranges the open Windows so that they are all visible and accessible horizontally on the screen.

Tile Vertical
Arranges the open Windows so that they are all visible and accessible vertically on the screen.

The center items show or hide their corresponding window or palette.

Time Controller

Current Time

Objects Window

Debugging Window

Library

QScript Reference

The three bottom menu items display the currently open windows by name. These can be scrolled through with their corresponding shortcut commands.

Help Menu

The Developers Release of LiveStage uses Balloon Help to show the names of the various components of the dialogs and windows.

Visit the Totally Hip Web Site

This will launch the default web browser and go to the Totally Hip web site.

Get LiveStage Online Support

This will launch the default web browser and go to the support area of the Totally Hip web site. Here information, FAQs and software updates can be found.

Register LiveStage Online

It is important to register LiveStage in order to receive the QuickTime Pro unlock code, update/ upgrade information, and other information from Totally Hip.

Apple Menu

About LiveStage

This dialog box will display specific information about your version of LiveStage. This includes the exact version number, the serial number and the registered user’s name. Just click on the LiveStage graphic to close the image.

Chapter 3
Adding Interactivity with QScript

LiveStage and Sprite Animation

The sprite animation produced by LiveStage differs substantially from traditional video animation. With traditional video animation, a frame is described by specifying the color of each pixel. By contrast, with a LiveStage sprite animation, a frame is described by specifying which Sprites appear at various locations. At a given moment, a sprite displays a single image, or plays a specific sound selected from a pool of images or sounds shared by all of the sprites in the QuickTime movie.

The metaphor of a LiveStage sprite animation is a theatrical play. In a QuickTime movie, the sprite track bounds are the Stage Window. In an application such as a web browser, a QuickTime movie is the Stage. The background in a LiveStage authored QuickTime movie is the set of the play. The background of the Stage may be a single solid color, an image, or a combination of images. Essentially, the sprites in a LiveStage authored movie, are the actors in a play.

A Sprite contains properties describing its location and appearance on the Stage at a given point in time. During the course of an interactive animation, a sprite's properties are modified to cause it to change its appearance, and move around the Stage.

Each Sprite Object in a QuickTime movie refers to an image stored in the Objects window Image tab. This tab contains a list of all of the image index numbers used by the Sprites in a movie. During the animation, a sprite's image index can be changed. This means that the image linked to the sprite can be changed. For example, a QScript can be created which assigns a series of images to a sprite in succession to perform cel-based animation – each image being replaced by another, while still maintaining the original sprite. A QScript can also be created which makes a sprite move around.

The Objects Window is like a container for all the movie data such as sprites, images, paths, sounds. It allows the sprites in your movie to share the same sources. Images are stored in the Object Window by index, and each Sprite Object has an image index property that specifies the sprite's current image.

QScript is entered in the script pane of the Sprite Editor Action tab in conjunction with the Object Window.

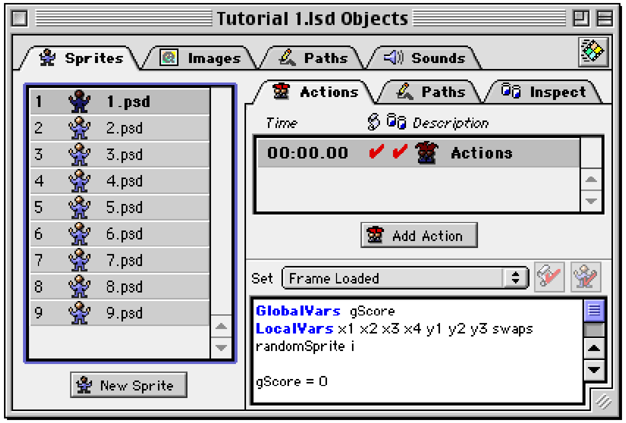
INTRODUCTION TO QSCRIPT

Now it is time to learn more about QScript, the multimedia scripting language built into LiveStage. The QScript language is what makes it possible to create interactive QuickTime movies for playback on CD-ROM, the World Wide Web, within programs such as Microsoft Word, PowerPoint, or any other media that can play QuickTime movies.

This chapter shows how QScripts work. Entering QScripts into the Qscript scripting window, and selecting the correct QScript commands from the QScript Reference Palette are described.

Scripting in the QScript Pane

Unlike traditional QuickTime movies LiveStage compiled QuickTime movies can incorporate interactive features that allow your viewers to become more actively involved with your movie creations. QScripts tell the movie how to respond to user input, as well as execute events at specified times. For example, you could write a QScript that plays musical notes when the viewer clicks on sprites, or have music play two seconds after a movie has started. Another example is an interactive user interface that allows viewers to navigate through a web site. In fact, QScripts can be written to allow user inputs to affect almost every aspect of a movie.



Set Frame Loaded

```
GlobalVars gScore
LocalVars x1 x2 x3 x4 y1 y2 y3 swaps
randomSprite i

gScore = 0
```

SCRIPT PANE WITHIN THE SPRITES TAB OF THE OBJECTS WINDOW

Basics of QScript

QScripts are written in the script pane of the Sprites tab within the Objects window.

To understand QScripts and their structure, it is important that a basic understanding of the components that comprise a script is achieved. People familiar with Lingo™ for Director™, BASIC™, C++™ or most other programming languages should find QScript very easy to understand.

A set of QScript statements is what handles each Event that is set with an Action. These scripts (Handler) are executed when an event occurs at the time specified by the Action, creating the interactive element to movies.

Basic interactivity can be created without knowing much about QScript by using the pre-made scripts from the QScript Reference palette. These can be dragged and dropped into the script pane in the Sprite Editor. This provides much greater creative flexibility and control over the many different visual and audio special effects QuickTime is capable of generating.

QScript is composed of more than 120 statement types which are split into a number of different categories. These include:

Properties

Operators and Operands

Control Statements

Constants and Variables

ANATOMY OF A QSCRIPT PROGRAM STATEMENT

A line of QScript is called a program *QScript statement*. A program statement is any combination of QScript keywords, expressions, properties, commands, etc., which collectively create a valid instruction or script that is recognized by the LiveStage compiler. Once executed, this series of commands instruct sprites to perform a specified action.

A complete program statement can be a simple keyword, such as:

```
GotoURL("http://www.totallyhip.com")
//where GotoURL is the Keyword for triggering a URL to be shown in
a browser, ("") the URL in the brackets is the destination
```

These scripts go to a specific URL. In this case, the Totally Hip home page.

A combination of elements can also be used, such as the following statement:

```
GlobalVars gWin
//where GlobalVars defines this as a global variable, 'gWin' is
the name of the global variable
IF (gWin = 1)
//if 'gWin' is returned as 'true'
   GoToURL("http://www.totallyhip.com")
//then go to the URL shown in the brackets
   gWin = 0
//'gWin' equals 'false'
ENDIF
//end
```

A script is used for two main reasons: to subdivide complex tasks or actions into more manageable portions, and to provide a way to respond to an Event in an interactive movie.

The rules of construction that must be used in building a QScript statement are called statement syntax. LiveStage QScript shares many of its syntax rules with other scripting languages, and language compilers.

Components that make up this syntax are as follows:

Sprite Object

A Sprite Object is a graphic or image object that is placed on the Stage and can be displayed by QuickTime. With sprite animation, a frame is described by specifying the images that appear at various locations on the Stage. Sprites may be parts of movies, or they may exist independently.

A Sprite Object is identified in one of three ways:

Index:

This is a value automatically assigned by LiveStage, but is not visible. It is useful when using the Random statement.

Using Index can have potential problems, however. Assume there are 3 sprites in a list and LiveStage has assigned them values of 1,2,3. If the first one is removed, the remaining two move up one number in the sequence – 2 becomes 1, and 3 becomes 2.

This number is not modifiable.

Name:

This value is automatically assigned by LiveStage, and is shown as a Sprite#. It can be renamed from the Sprite Editor.

ID:

This value is autmatically assigned by LiveStage, and is shown as a number. It can be renamed from the Sprite Editor.

Actions

A fixed time in a movie can be denoted by an action. When that time is reached, the events denoted at that particular time are executed by the scripts associated with them. When a new action is created, more events can be added at that particular point in time.

Actions can only be added to timed projects. If there are already actions at the end of the movie, more time may be needed before more actions can be added. If this is the case, go to the Document Settings from the Edit menu and either make the movie timed if it is not already timed, or add more time to the project.

Actions are created from within the Sprites Editor by clicking the New Action button. This is also where parameters associated with any actions can be viewed or modified. (See Events)

Any object; the movie, a track or Sprite Object can perform an action. Most actions can be performed on a specific object type, while some actions need no object to be specified. To tell a target to execute an action, the user must specify the Target, followed by a period, and then the action.

For example:

```
"SpriteNamed("Test").SetVisible(true)"
```

Some actions can have their values limited numerically by a minimum and maximum value. You can specify this limit by adding a `MIN(value)` and/or a `MAX(value)` immediately after the action. When a value is adjusted using an action which adds to the current value, wraparound can also be specified. This means that when the value gets past the minimum setting, it will reset to the maximum setting. When it gets past the maximum setting, it will go back to the minimum setting.

These settings will be specified in square brackets `[ ]` indicating they are optional.

For a complete list of actions see Appendix A.

Targets

Each action is always performed on an object, called a target. Targets do not always need to be specified. Each line of QScript will perform some action on a target. The target must currently exist at the time the QScript is executed in the movie.

There are three types of targets:

a *movie*

a *track* in the movie

a *Sprite Object*

A target can be any type of movie track such as a sound or MIDI track, QuickTime VR track, 3D track, or a video track. The Sprite Object executing the QScript is considered the default sprite target, and the sprite track that it is contained in is the default track target. This means that you do not have to specify a target if the target is the current default target.

For example, to change the image of a Sprite Object just type SetImageIndexTo(theIndex) instead of specifying which sprite should change its image.

Note!
There are some actions that apply to both a track and a sprite. In this instance the sprite would receive the action.

To send the action to the track, it would need to be explicitly specified by using ThisTrack to identify the target.

For a complete list of all the target types that can be specified see Appendix B

Events

After an action is set (see Actions) an event(s) must be selected to associate with that action. Events are things such as a click of the mouse or an occurance that happens over a period of time that a movie plays, or runs. After each event is chosen, QScript must be added as an Event Handler to specify exactly what is to happen when the Event occurs at the time specified by the action.

Select Events from the Set drop-down menu in the Sprites Editor in the Object Window. Once selected, add the QScript in the pane below by typing it, or dragging and dropping it from the QScript Reference, then modifying it as needed.

A QScript can be attached to the following list of common events. Most of them involve basic concepts of interactivity, such as the Mouse being pressed down, or the frame being loaded:

`Frame Loaded`

A `Frame Loaded` Event is generated before the first frame of the movie is loaded. This is a good place to initialize any global variables or set up additional things.

`Mouse Enter`

If the user moves the mouse over the screen region of the Sprite Object, the `Mouse Enter` Event message is sent to the Sprite Object in the movie.

`Mouse Exit`

If the user moves the mouse outside the screen region of the Sprite Object, the `Mouse Exit` Event message is sent to the Sprite Object in the movie which received the original Mouse Enter Event.

Mouse Click

If the user clicks the mouse in a non-transparent screen region of the Sprite Object, the `Mouse Click` Event message is sent to the Sprite Object within the movie. If multiple Sprite Objects are overlapping, then the top-most Sprite Object that is clicked on in a non-transparent area, will receive this Event message. The other Sprite Objects will not receive the Event message.

End Mouse Click (In Self)

If the mouse button is released over the screen region of the active Sprite Object, the `End Mouse Click End (In Self)` message is sent to the Sprite Object within the movie.

End Mouse Click (Anywhere)

The sprite that received the `Mouse Click` message also receives this message once the mouse is released.

Idle

When a movie is created in LiveStage, the idle frequency, or minimum time that must pass between `Idle` Events, is specified.

The things done in the Idle Event Handler should be kept short and quick. The `Idle` Handler is called between frames while QuickTime is not doing anything else. However, if too much time is spent in the Event Handler, QuickTime and its next frame update can be delayed. This can slow the frame rate down, and consequently degrade the overall performance of the movie.

If QuickTime is too busy processing other messages, or if other operations are executing on the computer, the `Idle` Event may not be received at the interval that is specified in the idle frequency field of the movie.

Note:
`Idle` *Events will not be processed faster than the idle frequency that is specified in the movie Document Settings dialog.*

Request To Modify Movie

Someone is about to modify your movie.

Custom Event

A `Custom Event` allows an Event that can be attached to a Sprite Object to be specified. This is like creating your own Event Handlers.

To create a Custom Event in QScript. Select Custom Event from the Action pop-up, and the New QuickTime Event dialog will appear. Choose a Custom Event ID number between 1-65535.

Once a Custom Event is defined, other Sprite Objects in the movie can be told to execute this particular Event for the Sprite containing the custom Event.

These Custom Events can be reused at any time in a movie, or use them with more than one Sprite Object.

Image Transforms

An Image Transform Event allows a Sprite Object within a movie to be described and have its properties changed. By modifying the `Image Transform` Event of a sprite, the x and y coordinate location of the Sprite Object can be changed so it appears to move along a smooth path on the screen, or jump from one place to another. The size of a Sprite Object can be specified so that it shrinks, grows, or stretches. Other transformations such as rotation are also supported. The layer property of a Sprite can be checked and modified to specify the layer of the Sprite Object in the animation or movie. Sprites with lower layer numbers appear in front of Sprite Objects with higher layer numbers. To designate a Sprite Object as a background Sprite, you should assign it the special background layer.

Display Settings

Display Settings for a Sprite Object within a movie can be checked, changed or modified. The visible property of a Sprite Object specifies whether the sprite is visible on the Stage. To make a sprite visible, set the sprite's visible property to TRUE. The graphics mode property of a Sprite Object specifies the graphics mode and indicating how to blend a Sprite Object with any Sprite Objects behind it, and the background of the Stage window. This allows for some very sophisticated image transformation effects on Sprite Objects within a movie:

Modes

Boolean, Arithmetic and Miscellaneous modes can be found in the pull-down menu.

Boolean Transfer Modes

The following modes describe an interaction between the pixels the application draws, and the pixels that already exist within the destination bitmap.

Copy

This completely replaces the pixels in the destination bitmap with the pixels in the source bitmap.

`Inverse Copy`

This completely replaces the pixels in the destination bitmap with a ‘photographic negative’ of the source bitmap.

`Or`

This adds the black pixels from the source bitmap to the destination bitmap.

`Inverse Or`

This takes a ‘photographic negative’ of the source bitmap, and then adds the black pixels from this negative to the destination bitmap.

`Exclusive Or`

This inverts the pixels in the destination bitmap that correspond to black pixels in the source bitmap.

`Inverse Exclusive Or`

This inverts the pixels in the destination bitmap that correspond to white pixels in the source bitmap.

`Bit Clear`

This turns pixels white in the destination bitmap, when they correspond to black pixels in the source bitmap.

`Inverse Bit Clear`

This turns pixels white in the destination bitmap, when they correspond to white pixels in the source bitmap.

`Arithmetic`

The following operations can be performed on the values of the red, green and blue components of the source and destination pixels. They produce predictable results on indexed devices because they work with RGB colors rather than with color table indexes.

`Blend`

This replaces the destination pixel with a blend of the source and destination pixel colors.

`Transparent`

This replaces the destination pixel with the source pixel if the source pixel is not equal to the background color.

Add Max
This compares the source and destination pixels, and replaces the destination pixel with the color containing the greater saturation of each of the RGB components.

Add Min
This compares the source and destination pixels, and replaces the destination pixel with the color containing the lesser saturation of each of the RGB components.

Add Over
This replaces the destination pixel with the sum of the source and destination pixel colors. If the value of the red, green or blue component exceeds 65,536, then subtract 65,536 from that value.

Add Pin
This replaces the destination pixel with the sum of the source and destination pixel colors up to a maximum allowable value.

Sub Over
This replaces the destination pixel with the difference of the source and destination pixel colors. However, if the value of a red, green or blue component is less than 0, add the negative result to 65,536.

Sub Pin
This replaces the destination pixel with the difference of the source and destination pixel colors; but not less than a minimum allowable value.

Miscellaneous

Dither
This mixes existing colors together to create the illusion of a third color that may be unavailable on an indexed device. It also improves images that are shrunk or copied from a direct-pixel device to an indexed device.

Grayish Text Or
This draws dimmed text on the screen. If the destination device is color, it will be drawn with a blend of the foreground and background colors. If the destination device is black and white, the black and white will be dithered.

Hilite
This replaces the background color with the highlight color when your application draws or

copies images between graphic ports. This has the visual effect of using a highlighting pen to select the object.

Reserved Keywords

QScript recognizes Reserved Keywords. These are words used in QScript which have special meaning. Typically, these are the names of different object types in a QuickTime movie, such as sprite, sprite track, target, sound, VR Track, music track, or movie.

Reserved Keywords are also used to define variables in QScript, such as:

```
GlobalVars gRobot
//where GlobalVars defines this code as a global variable, gRobot
is the name of the global variable
```

With this script, the variable ‘`gRobot`’ is to be made available to all scripts in the current movie. The intent of the name is actually Robot, but because it is global, the ‘g’ prefix will help to denote it as such further into the scripting. This is merely a convention and does not have to be followed.

When a Special Reserved Keyword is entered in the scripting window, QScript will recognise it and make the text blue, capitalized and bolded automatically so it can easily be recognized. This is the default text Syntax Style for Reserved keywords. The Syntax Style can be changed from the Preferences in the Edit menu (see Preferences).

Note:
Reserved Keywords cannot be used as names for sprites, images, etc.

Properties

Properties are values that describe each aspect of an object.

For example, the title of a window is a property, as is it’s width, height and any other defining characteristic. When the window is open, each of these properties are copied into memory. They can be accessed by name, and values can be obtained from them and assigned to them. To change to title of a window when a button is clicked, the title property of the window would need to be set to the new value.

Properties store different types of information. Some store text, such as the title of the window, while others store number, such as the width of the window.

Properties can be used anywhere a number can be used, including expressions.

Example:

```
TrackVolume
```

This property returns the volume setting for the Sound track. `0` (off) to `255` (full).

For a complete list of properties, see Appendix C.

Variables

A variable stores information exactly like a property, only it is not directly related to an object. It is simply a location in memory that stores a value.

Names are also assigned to variables, and should be created to describe the purpose of the variable. For example, to keep track of how many units of a product have been sold, the variable may be named 'Sales.' The variable names tell QScript where to look in the computer's memory to find their value.

As indicated by their name, variables are changeable values that can be altered dynamically and referred to during the course of script execution. Think of them as storage areas or cupboards for individual pieces of information.

Declaring Variables

In order to use variables for the storage of values you must first declare them. To declare a variable, it must be on the right side of the assignment operator (=).

For example, if the caption of a button was needed to return the value of a variable called 'Sales', it could look like this:

```
Button1.Caption=Sales
```

Naming variables can be a little tricky because the names need to be short, but intuitive and easy to remember. To avoid confusion, use the following conventions when naming variables:

- Begin each variable name with a letter. Variable names must be fewer than 256 characters long and cannot contain periods.
- Make your variable names descriptive by combining one or more of the words when it makes sense to do so. For example, the variable named AlienSpaceCraftRate is much clearer than Craft or Rate.

- Use a combination of UPPER and lower case characters and numbers if you wish. An accepted convention is to capitalize the first letter of each word in a variable; for example, SpeedOfSpaceCraft
- Do not use QScript keywords, objects, or properties as variable names.
- Use the prefix ‘g’ to help distinguish global variables from local variables at a glance.

Once declared you can assign values to variables using the ‘=’.

For example:

```
x = 1
//where ‘x’ is the name of the variable, and 1 is the value
```

Variables can also contain an array of values. Specify how many values to store by using square brackets and indicating how many are to be stored. Each value can be accessed in the same way. LiveStage supports one-dimensional arrays with multiple values in each.

For example, to keep track of a deck of cards an array can be declared that has 52 numbers stored in it like this:

```
LocalVars myCards[ 52]
//where LocalVars defines this code as a local variable, myCards
defines the name of the local variable, [ 52] denotes the number
assigned to the variable ‘myCards’
```

Access a card like this:

`myCards[ 1]` or `myCards[ currentCard + 4]`

Local Variables

Each QScript can have its own set of local variables. These variables can only be used by the QScript in which they are defined, and will not affect other QScripts in the movie.

Local variables start out in an unknown state, so they must be initialized to a known state before they can be used.

Use the `LocalVars` statement and then list all the local variables that you will use in this QScript, separated by either a comma or a space. These variables start out in an unknown state, and remember their values only until the QScript finishes executing.

For example:

```
LocalVars numClicks counter x y z
```

GLOBAL VARIABLES

QScripts can also have global variables. Global variables can be used in other QScripts within the project. They always retain their values in a movie once they are set.

Global variables should be initialized in response to a Frame Loaded Event. Global variables can be used to communicate information from one QScript to another.

Global variables can also be declared.

For example:

```
GlobalVars gAlive gWin
```

These variables remember their values all the time. They are the same for all QScripts, and can be used in any QScript. This allows different QScripts to communicate or pass information to each other.

Global variables must be declared in each QScript that uses a variable, and it is a good idea to name them differently so they can easily be recognized as being global variables.

CONTROL STATEMENTS

As you start to write scripts of greater complexity, you will find the need to have your scripts choose among different actions.

There are two ways to control the order that actions are executed. You can use either the `IF.. ELSE.. ELSEIF..ENDIF` or the `WHILE, ENDWHILE` statements.

The `IF` statement will execute the actions up until the matching `ELSEIF, ELSE` or `ENDIF`, only if the expression evaluates to true, meaning that it must be the same value set in the statement. If it does not evaluate to true, then any matching `ELSEIF` expressions are tested to see if they are true.

Here is a plain-English example to describe how an `IF ELSE` statement might react to a key being pressed:

`if` *the user presses the Command key, do something,*`else` *stop the movie.*

Note!
The first piece of code found to be true will be the one that executes. If no expressions in a script are true, then the code between the `ELSE` *statement and the* `ENDIF` *statement will be executed.*

The `WHILE` statement will execute all actions up to the matching `ENDWHILE`, as long as the expression is true. It will continue to execute the statements over and over, possibly until you reset your computer, so make sure there is a way out of the script.

For example:

```
IF (x > 1)
//if 'x' is greater than 1

x = x - 1
//subtract 1 from the value of 'x'

ELSEIF (y > 1)
//if 'y' is greater than 1

x = x + 1
//add 1 to the value of 'x'

ELSE

Z = 3
//otherwise assign 3 to 'z'

ENDIF

IF (x > 1)
//while 'x' is greater than 1
```

```
x = x - 1
//subtract 1 from the value of 'x'

ENDIF

WHILE (x > 1)
//while 'x' is greater than 1

x = x - 1
//subtract 1 from the value of 'x'

ENDWHILE
```

Expressions

An expression is a combination of one or more variables, numbers or actions and mathematical symbols. Expressions can be used anywhere a simple number can be used.

Expressions consist of operators and operands. Operators are items such as '+' and '-', while operands are variables or numbers. Operators and operands are combined using various rules. Most operators require an operand on its left and its right, such as '`1 + 2`'.

Some operators operate on the next operand only such as '`-x`'.

Operators

Unary operators require only one operand on the right. The following operators are all unary operators.

NOT

Reverses the boolean value of the following operand.

NEG or -

Reverses the sign of the following operand.

ABS

If negative, reverses the sign of the following operand.

-

Binary operators require 2 operands, one on the left and one on the right.

<

Compares the two operands and returns `true` if the left is less than the right, `false` otherwise.

<= or ≤ (option + ,)

Compares the two operands and returns true if the left is less than or equal to the right, false otherwise.

=

Compares the two operands and returns `true` if the left is equal to the right, `false` otherwise.

≠ or != (option + =)

This operator will evaluate to true if the expression on the left is not equal to the expression on the right.

-

Compares the two operands and returns `true` if the left does not equal to the right, `false` otherwise.

>

Compares the two operands and returns `true` if the left is greater than the right, `false` otherwise.

>= or ≤ (option + .)

Compares the two operands and returns `true` if the left is greater than or equal to the right, `false` otherwise.

+

Adds the two operands together and returns the result.

-

Subtracts the second operand from the first and returns the result.

*****

Multiplies the two operands together and returns the result.

÷ or / (option + /)

Divides the first operand by the second and returns the result.

OR

If either operand evaluates to `true` then returns `true` , `false` otherwise.

AND

If both operands evaluate to `true` then returns `true` , `false` otherwise.

DIV

Same as ÷ except only whole numbers are returned.

MOD

Same as ÷ except only the remainder is returned.
For example `23 MOD 10` is the remainder after dividing 23 by 10 which is 3.

It is very important to remember that operators also have precedence rules. This means that operators with higher precedence execute before those of a lower precedence.

For example, consider `1 + 2 * 3`. What is the answer?

If we add `1 + 2` and mulitply `* 3` we get 9. But mathematical rules state we should do `2 * 3` then add 1 so we get 7. This is because * has a higher precedence than +.

Highest precedence: `* ÷ DIV MOD + - < 2 = - > 3 OR AND`
Lowest precendence: ≤ ≥ ≠

You can also use parentheses to control what gets executed first like the following:

```
1 + (2 * 3)
```

Anything in parentheses is executed first. For example:

```
x + 1 * (y ÷ 3)
```

`Random(1,8) * y`

`x + 1 * 3 < y + 5 AND z = y` (remember the precedence rules)

Example code:

```
LocalVars Var3000 Var41 Var23 Var24 Var16 Var11 Var37 Var10 Var36
Var39 Var4000
//assign local variables

Var24 = Index
//assign Var24 the value in Index

Var24 = ImageIndex
//assign Var24 the value in ImageIndex

SpriteOfIndex(Var41 + Var23).SetImageIndexTo(Random(1,2))
//go to the sprite based on Index and randomly assign a new image
to the sprite

TrackNamed("Fire").SetEnabled(FALSE)
//disable the track named 'Fire'

TrackNamed("Fire").SetEnabled(TRUE)
//enable the track named 'Fire'

Var24 = -Var24
//assign Var24 to a negative value

Var24 = Var24 * Var11
//multiply Var24 by Var11 and assign new value to var24

Var11 = Var24
//make Var11 equal Var24

IF (Random(1,100) > Var23)
//if random value between 1-100 is greater than Var23
```

```
Var23 = 0
//then set Var23 to 0

ENDIF

IF (KeyIsDown(kOptionKey,‘ ’))
//if space bar is down while the option key is pressed, subtract 0

ThisSprite.ExecuteEvent(7)
//execute event #7

ENDIF
```

Comments

It is recommended to use comments in the QScript code to describe how important statements work within the project. While it may make sense when it is written, it may not days or weeks later. As well, if someone else needs to view the code and understand what has been done, commenting it will make the job a lot easier.

Comments in LiveStage QScript follow the C++ commenting standard: everything after two slashes (“//”) on a line is ignored. They will automatically appear in red.

These notes are not stored in the movie when the program is compiled; they exist only to document what the program does, and are ignored by the complier.

For example:

```
GlobalVars win
//defines the global variable ‘win’

IF (win = 1)
//if ‘win’ equals ‘1’ then

    GoToURL("<http://www.totallyhip.com/link/
productslivestage.html>T<_top>")
```

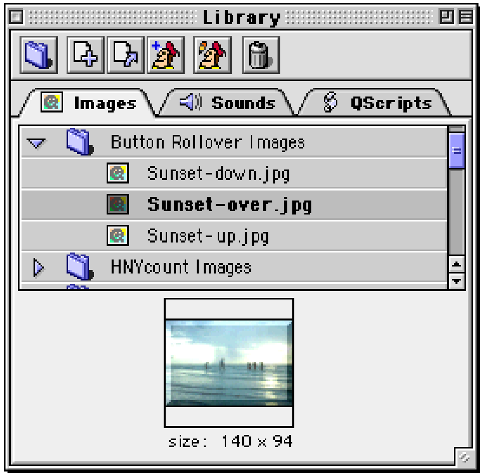
```
//Go to the LiveStage page at Totally Hip’s web site if the game
is won

    win = 0
//‘win’ equals ‘0’

ENDIF
```

Creating a QScript in the Sprite Pane

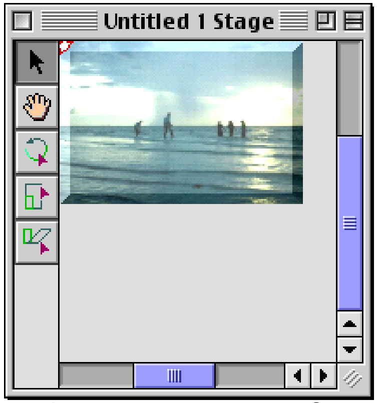
To create or write a QScript for a Sprite Object, add an image to the Stage window by dragging and dropping it from the Media Library. The image will appear in the Sprites list pane and also in the Images list pane.

1. Select a sprite to add a QScript to, and then select the action.

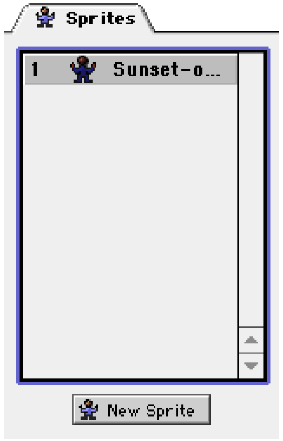
   Click on the 'Sunset-up.jpg' Sprite in the Sprites list pane. An Event action from the Set Action popup menu can now be selected.



1. Drag image file to Stage

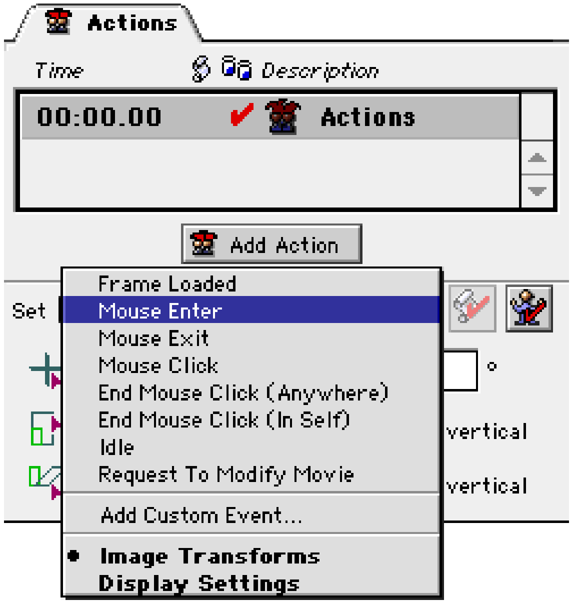


2. Image dragged to the Stage creates a new sprite



3. Click on the Sprite

2. In this example, the `Mouse Enter` Event pop-up has been chosen.
3. This Event will instruct LiveStage that a `Mouse Enter` Event is being checked when a user moves the mouse cursor over, or into the 'sunset-up.jpg' Sprite.

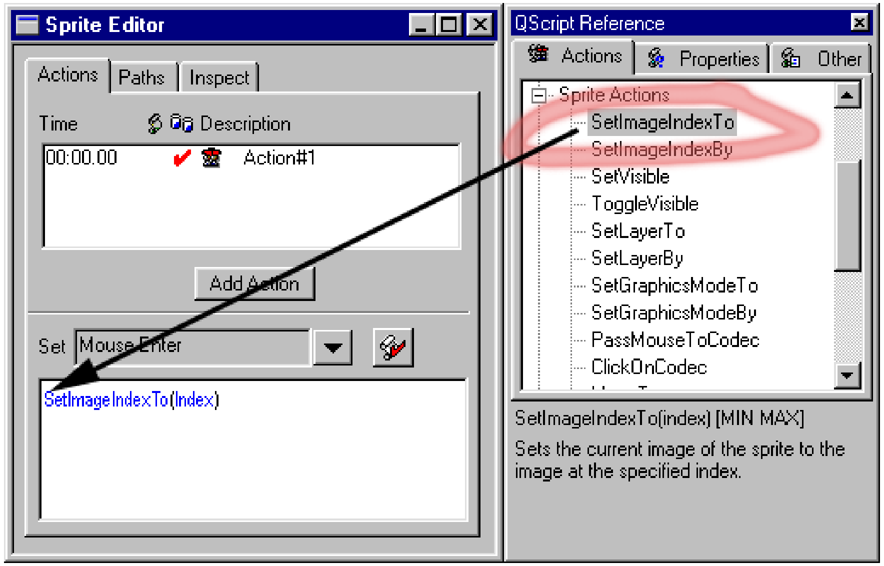


When we refer to Events in QScript, we are referring to Event Handlers. These are employed to respond to Events in a QuickTime movie. These often have to do with user actions such as moving the mouse in a specific Sprite Object or area of the Stage, clicking the mouse, or pressing a key on the keyboard.

Once an Event Handler such as Mouse Enter has been set, the QScripts can be entered in the QScript script text pane. The QScript pane can only be accessed while working with Sprite Objects.

The QScript pane within the Object Window is where you will spend most of your time during the Tutorials and Lessons in this Manual Book.
If a QScript is meant to change at a particular time in a movie, add or select the action at that time and add your new QScript to that action.
When the movie plays past the point of the new action, any earlier actions will be replaced. This allows the behavior of the sprite to be changed depending on the time of the movie.
There are a number of ways to enter a QScript into the scripting window:

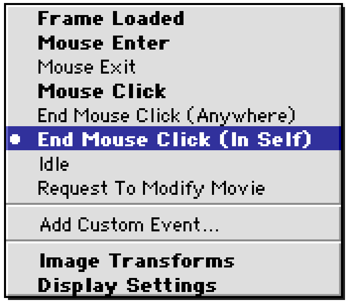
1. Drag a Reserved Keyword Action from the QScript Reference Palette and drop it into the QScript scripting pane in the Script Editor. LiveStage allows things to be drag and dropped from one window or palette to another to save time.
2. Type the QScript keywords into the script window.



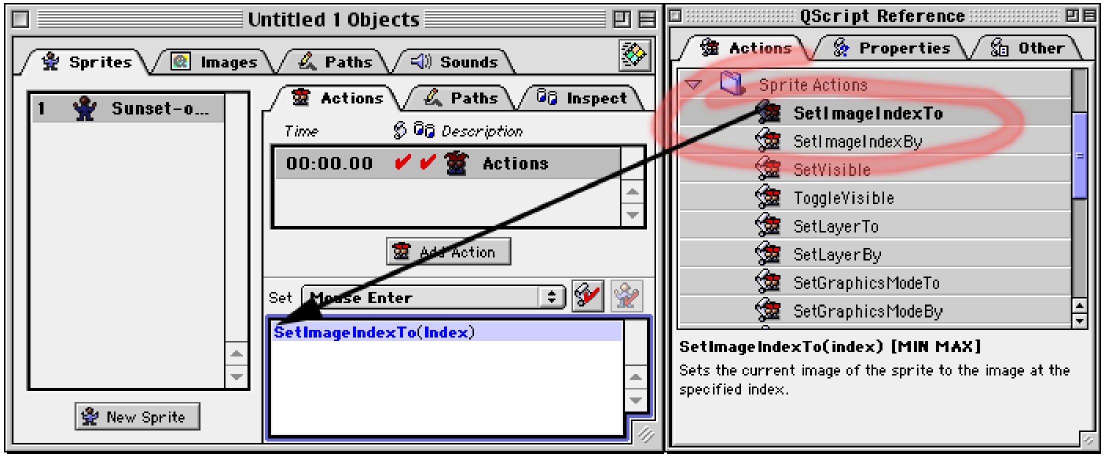
```
SetImageIndexTo(Index)
```

SetImageIndexTo(index) [MIN MAX]
Sets the current image of the sprite to the image at the specified index.

3. Drag and drop QScripts from the Library Palette into the QScript scripting pane. The Library allows previously made images, sounds, and QScripts to be stored in folders on a hard disk or other storage devices, and then accessed later to include in movie projects.



EVENTS THAT HAVE QSCRIPTS ATTACHED TO THEM WILL APPEAR IN BOLD WITH THE CURRENT SELECTION BULLETED



SetImageIndexTo(Index)

SetImageIndexTo(index) [MIN MAX]
Sets the current image of the sprite to the image at the specified index.

DRAG THE SCRIPT FROM THE QSCRIPT REFERENCE TO THE SCRIPT EDITING PANE

4. Click on the QScript Tab to show the available QScript folders in the Library Palette. Open the appropriate QScript folder to show the available QScripts.

The Library palette will display images, sounds, or QScripts that are in the Library folder. A folder can also contain an alias to images, sounds, and QScripts on a local hard disk, or on other hard drives or storage devices.

The QScript Reference Palette has reference text to help clarify what the Script does before it is used in the movie project.

The scripts are automatically compiled by LiveStage. Compilation is the process of converting a plain-English-style segment of written code into a language that the computer can read and understand. To see the effects of the scripts select Run the Wired Movie from the File menu. Changes can be made to the LiveStage QScripts at any time. When the Wired Movie is activated again it is automatically recompiled.

Check Syntax

Once the QScript has been entered, it can be checked for syntax errors by pressing the check syntax button.

Any errors found will be shown in the Errors window where they can be selected, causing the appropriate line of script to be high-lighted in your QScript.

Adding QuickTime Movies to LiveStage

LiveStage does not currently support importing regular or VR QuickTime movies directly for manipulation. Instead the QuickTime MoviePlayer Pro must be used to copy the source movie, and paste it in the LiveStage movie. This is outlined in the X2K sample within the Samples folder that is installed with LiveStage.

Version 2 will support direct import of QuickTime movies.

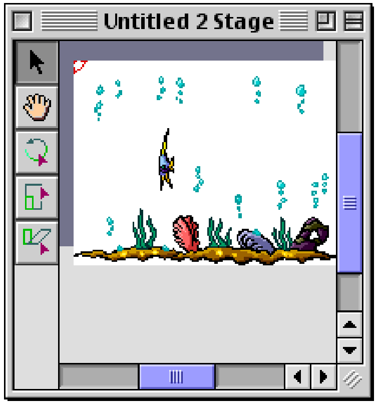
Chapter 4
Creating Animation

These are the basics of creating an animation within LiveStage from an animated GIF file.

Before beginning, save the original animation as individual frames. Copy them to a folder within the LiveStage Library Images folder. If they do not show up in this folder, click the Add Item icon in the Library pane and add them to the list, or drag them into the Library Images pane.

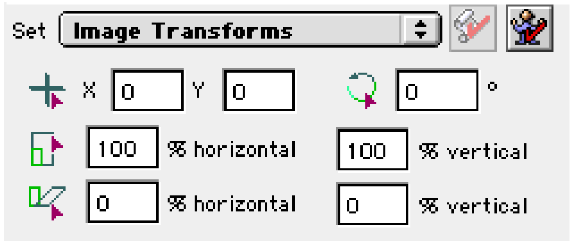
1. Create a new LiveStage document that is the dimensions of the animation cels. In this case 152 x 118.
2. Click OK.
3. Set the Idle rate to the speed in which the animation is to play. In this example it has been set to 15.
4. Drag the first image to the Stage

   A sprite will automatically be created.



4. Drag the first image to the Stage

5. In the Actions tab of the Sprite Editor, select Image Transforms from the pop-up menu. Ensure the x,y coordinates are at `0` so the image is perfectly aligned on the Stage from the top left.

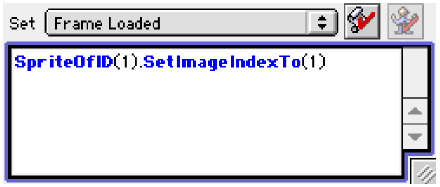


5. SET THE X,Y CORRDINATES TO 0

6. Drag the other three images in sequence to the Images tab of the Object Window.
7. Select Sprite#1 and click the Set pop-up in the Actions pane.
8. Select Frame Loaded.

   This will initialize the script, and will be set each time the movie is played from the beginning.
9. Enter the following script by typing it, or dragging the components from the script library:

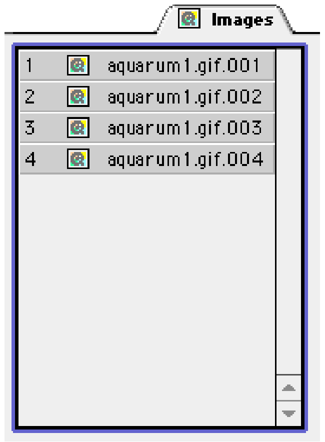
```
SpriteOfID(1).SetImageIndexTo(1)
//This sets the first image of Sprite#1 to
Image #1
```



Set Frame Loaded

```
SpriteOfID(1).SetImageIndexTo(1)
```

9. FRAME LOADED EVENT

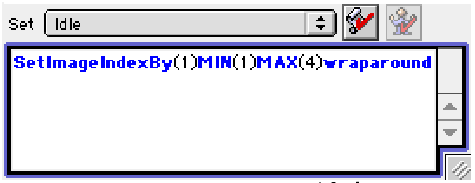


Images

1	aquarum1.gif.001
2	aquarum1.gif.002
3	aquarum1.gif.003
4	aquarum1.gif.004

6. DRAG OTHER 3 IMAGES TO THE OBJECT WINDOW

10. Now select the Idle event from the drop down list. This will add a new event to the same sprite.


Set Idle

```
SetImageIndexBy(1)MIN(1)MAX(4)wraparound
```

10. *IDLE EVENT*

```
SetImageIndexBy(1)MIN(1)MAX(4)wraparound
//This changes the image the sprite
is displaying by a value of 1 from
the list of the four images. This
means it will select them in se-
quence - 1,2,3,4. MIN indicates an increment of 1, MAX indicates
the total number of images this event is applied to. Wraparound
means the animation is meant to loop.
```

CHECK SYNTAX

11. Click the 'Check Syntax' button to verify the scripts are correct.

12. If there are no errors, select Save from the File menu. The Save dialog appears. Choose the file name and path you wish to save the project file to.

RUN WIRED MOVIE

13. Select Run Wired Movie from the File menu or the button to see the result.

Chapter 5
Creating A Button Rollover

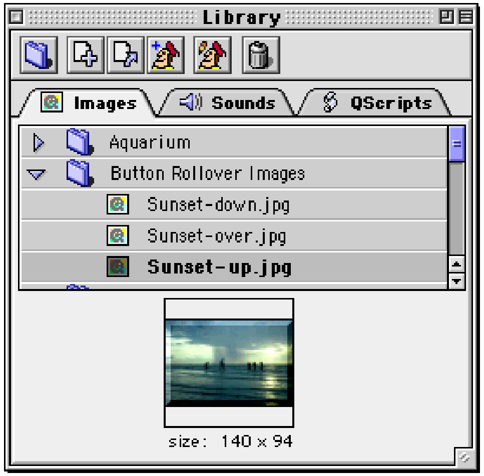
In this example, a picture button has been created which consists of three images. Three images are typically required to make a button rollover; the Mouse Up image, the Mouse Over image, and the Mouse Down image.

The images and scripts you will need are in the Library folder installed with LiveStage.

The Button RollOver example has already been completed and is in the Button RollOver Images folder inside the LiveStage folder.

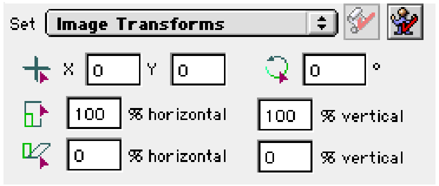
1. Launch LiveStage and create a new document.
2. Make the Size of the document 140 x 94 pixels.
3. Click OK.
4. Set the idle frequency of 1 tick count or 1/60 of a second.
5. Set the Color bit depth to 8 bits (256 colors) and ensure the background color is white.
6. Click OK.
7. Click on the Images Tab in the Media Library to make sure that it is open.
8. Now click on the arrow to the right of the Button RollOver Images folder.
9. There should be three 'Sunset jpg' images in the folder.
10. Drag the 'Sunset-up.jpg' image onto the Stage.

    Sunset-up.jpg sprite will automatically be created.



10. Drag Sunset-up.jpg to the stage

11. In the Actions tab of the Sprite Editor, select Image Transforms from the pop-up menu. Ensure the x,y coordinates are at 0 so the image is perfectly aligned on the Stage from the top left.



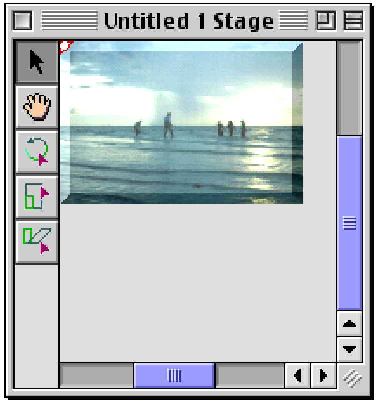
Set: Image Transforms

X 0 Y 0 0°

100 % horizontal 100 % vertical

0 % horizontal 0 % vertical

11. Set the X,Y coordinates to 0



10. Drag image to the Stage

Our goal is to have the 'up' image on the screen when the mouse is outside the image, then have it change to the 'over' image when the mouse is above it, and then change to the 'down' image when the user clicks on it. Finally we will take the user to the Totally Hip web site.

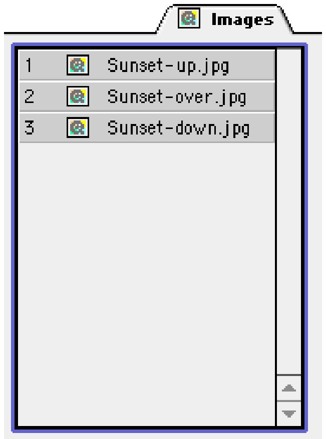
To do this we need to do the following steps:

12. Drag the 'Sunset-over.jpg', and 'Sunset-down.jpg' images in sequence to the Images tab of the Object Window.

    The Object Window should appear with the three images listed as follows:

    Sunset-up.jpg
    Sunset-over.jpg
    Sunset-down.jpg

13. Select Sunset-up.jpg sprite and click the Set pop-up in the Actions pane.



Images

1 Sunset-up.jpg
2 Sunset-over.jpg
3 Sunset-down.jpg

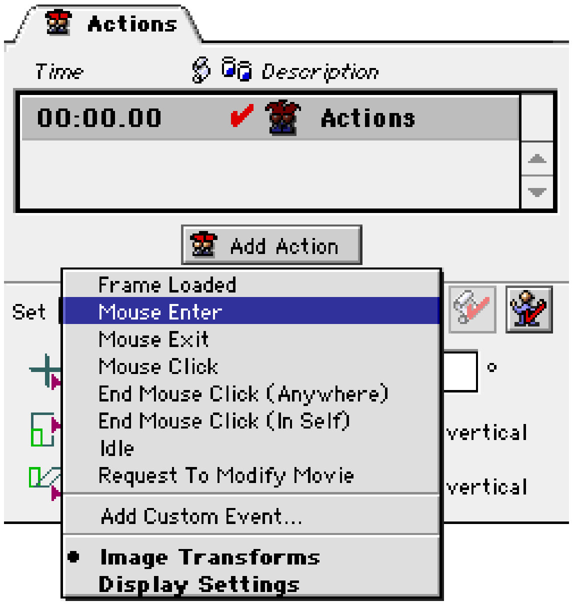
12. Three images in list

14. Select `Mouse Enter` from the event handler list.

    Now we will enter our first QScript in the Script Window.

15. In the QScript Reference Palette, select the Action tab.

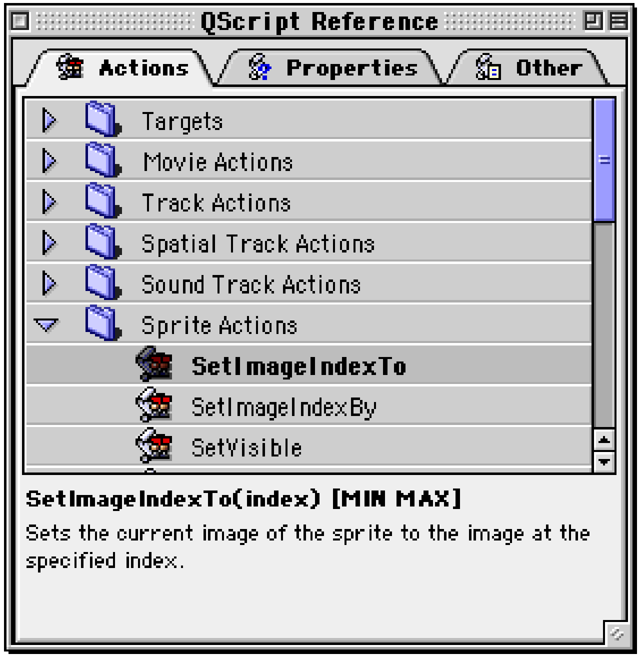
16. In the Actions tab, click on the Sprite Actions and click the arrow beside Sprite Actions.



14. SELECT MOUSE ENTER

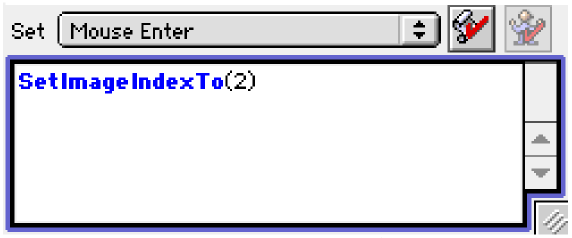
17. Choose the `SetImageIndexTo` Reference script from the Sprite Actions folder in the QScript Reference Palette, and drag it into the QScript window.
18. Delete the word ‘Index’ in parenthesis.
19. Now type ‘2’ where the word `Index` was. The new script should look like this:

```
SetImageIndexTo(2)
//This command will
change the 'Sunset-
up.jpg' to the 'Sunset-
over.jpg' when the mouse
is over it
```



SetImageIndexTo(index) [MIN MAX]

Sets the current image of the sprite to the image at the specified index.



```
SetImageIndexTo(2)
```

18. REPLACE THE WORD ‘INDEX’ WITH ‘2’

20. Select the event pop-up menu and choose `Mouse Click`.
21. Drag the `SetImageIndexTo(Index)` QScript from the QScript Reference Palette into the QScript window and change the word '`Index`' to the number '3'.

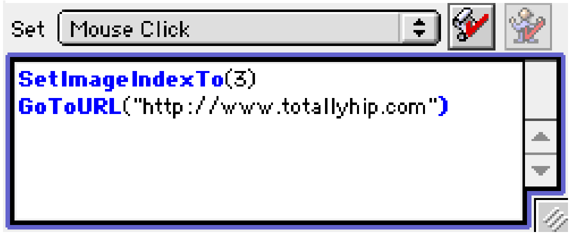
    This directs the 'Sunset-over.jpg' Sprite image to be replaced by the 'Sunset-down.jpg' image when the user clicks the mouse button.
22. Choose `Mouse Exit` from the Set pop-up menu.
23. Drag the `SetImageIndexTo(Index)` QScript from the QScript Reference Palette into the QScript window and change the word '`Index`' to the number `1`.

    This will tell the movie to change the image from 'Sunset-over.jpg' back to the 'Sunset-up.jpg' image when the user moves the mouse outside of the image without clicking on it.

    Now we want to add another QScript to take the user to a Web page URL when the button is pressed.
24. Choose `Mouse Click` from the Set pop-up menu.

    There will already be a script in the Set Mouse Click Action, so put the insertion point on a new line to create a new statement directly below the '`SetImageIndexTo(3)`' event.
25. Choose the General Actions statement `GoToURL("url")` from the General Actions folder in the Actions tab of the QScript Reference.
26. Drag this QScript into the QScript script window for the `Set Mouse Click` event.
27. Now type '`http://www.totallyhip.com`' to replace the '`url`' text. The QScript should reads as follows:

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



```
SetImageIndexTo(3)
GoToURL("http://www.totallyhip.com")
```

CHECK SYNTAX

RUN WIRED MOVIE

28. Click the ‘Check Syntax’ button to verify the scripts are correct.
29. If there are no errors, select Save from the File menu. The Save dialog appears. Choose the file name and path you wish to save the project file to.
30. Select Run Wired Movie from the File menu to see the result.

Adding a 'Button' Sound

You must do three things to create a button sound:

Add a sound to your movie in the Objects dialog.
Tell the sound to play when the 'button' sprite is clicked.
Tell the sound to stop playing when the mouse is released.

1. After creating your movie click on the Sounds tab in the Object Window.
2. Click 'New' at the bottom of the tab.
3. Pick a sound that uniquely suits the button you are creating.

   Note:
   Do not use a sound that degrades for this lesson. Sounds that degrade will only end at their predefined time.

4. In the Objects window select the Sprites tab. Select the specific sprite object to apply the sound to, and select the Action tab in the Sprite Editor.

   Select `'Set Mouse Click'` and use the `TrackOfIndex` action and `PlayNote` action together to play a sound, eg: `TrackOfIndex(2).PlayNote(1, 0, 55, 100, 999999`

   1 indicates the index of the instrument as defined by the user in the Sound Library.

   0 indicates the delay before the sound is played.

   55 indicates the pitch defined by semi-tones. with Middle C being 60, Middle B being 59, etc. The range is 32 to 95.

   100 indicates the velocity, or volume at which the note is played. The range is 0 to 255

   Note the duration of `999999`, this means the sound will play for as long as the mouse is held down. The time scale is 1/600th of a second with a range of 0 to 2047.

5. Select `'Set End Mouse Click (Anywhere)'` and do the same as above, except set the velocity to `0`, eg: `TrackOfIndex(2).PlayNote(1, 0, 55, 0, 0)`

Note:
#5 is only relevant when using non-degrading sounds. Sounds that degrade will automatically end at their predefined time.

See `PlayNote` *in Music Track Actions, Appendix A*

Chpater 6
Basics of QuickTime Movies on Web Pages

Creating QuickTime Content for the Internet

When QuickTime content for the Internet is being created, keep in mind that many people still have 14.4 or 28.8 modems. A 28.8 modem can download a movie at about 3.5 kBytes per second maximum, so a 10 second, 500k movie can take several minutes to download.

Historically, digital video movies tend to be large. A typical 30-second sound and video QuickTime movie designed for CD-ROM delivery can be several megabytes in size and can take minutes to download even over a fast connection.

Webmasters don't want to keep viewers waiting, or have the site overloaded. So if existing content is prepared for CD-ROM delivery, take short clips of the movie and re-compress them at a smaller size and lower data rate. There are several third-party products available which will help you do this.

There is a new feature in the QuickTime 2.0 (or greater) Plug-In - support for movies that reference other movies. Now a small movie can be created that points to several other tracks customized for different connection speeds. When a user downloads this movie, the plug-in ensures that the best track for the current connection type is downloaded. For more information on this feature, see the Creating Multiple-Data-Rate Movies Workshop.

The QuickTime Plug-In can play many kinds of QuickTime movies. It supports all of the QuickTime 3 track types, including text tracks, MIDI tracks, etc. Using these kinds of movies, compelling features can be added to web pages without significantly affecting the time they take to download.

Wired Movies on the Web

There are more QuickTime movies than any other video format on the World Wide Web. This is because QuickTime is extremely powerful, and easy to use, and therefore, the best choice to use for showing off interactive movies on the web.

This chapter tells how to ‘Run a Wired Movie’ and ‘Export a Wired Movie’ for distribution on the World Wide Web. Whether the final destination of the interactive QuickTime movies is the web, CD-ROM, or play back from a hard disk, there are some basics, which need to be understood.

This chapter describes the QuickTime Plug-in, and how to embed QuickTime movies into web pages. It also discusses the Embed Syntax and Tag Index, as well as more about creating QuickTime content for the Internet.

QuickTime vs. Other Streaming Media Formats

Digital media is becoming very popular on the web. There are many competing video technologies. QuickTime is the most popular and allows users the most flexibility.

Streaming media means the media, or movie, starts to play immediately instead of waiting for the entire file to download first. QuickTime movies are saved in a fast start format so it starts to play as the file is being downloaded.

QuickTime movie streaming is referred to as HTTP streaming, which differs from RTP streaming, used by dedicated servers. With HTTP streaming, you can simply put a QuickTime movie file on a web server the same way as image files and animated GIFs. When a user comes to the web page, the movies are downloaded to their computer, where the QuickTime Plug-in handles the task of playing the interactive movie. One advantage of QuickTime over other streaming media formats, is that the movie file does not have to be sent or served again over the network. The movie file is sitting on the user’s computer. If they want to play it again, they can without downloading the movie.

When an interactive movie is made, make certain that the movie files are not too large in size. We recommend that you make them as small as possible; under 30K (30 kilobytes) in size. A good size if there are a number of movies on the same page is 20K or less.

QuickTime Webmaster's Documentation

Apple's QuickTime Plug-In allows QuickTime movies to be embedded directly into web pages. LiveStage compiled interactive QuickTime movies are dealt with the same way.

Version 2.0 of the QuickTime Plug-in is required in order to view and interact with these interactive QuickTime movies. These are included with LiveStage and should have been installed during the LiveStage installation process.

These versions of QuickTime and the QuickTime Plug-in support the Mac OS, Windows 95, Windows 98, and Windows NT. For the latest version of QuickTime or to recommend to Web page visitors where to acquire QuickTime, tell them to visit the Apple web site.

Apple's QuickTime site:

```
http://www.apple.com/quicktime/
```

Apple's QuickTime site for WebMasters:

```
http://www.apple.com/quicktime/authors/webmas.html
```

For the latest version of LiveStage, and to get the most recent documentation and other valuable resources on LiveStage and WebPainter visit the Totally Hip Web site.

Totally Hip's Web site for the most recent information on LiveStage:

```
http://www.totallyhip.com/
```

Apple's QuickTime plug-in for the Internet displays QuickTime content directly in a browser window. A 'helper application' to view QuickTime content over the Internet is no longer needed. QuickTime VR (Virtual Reality) Panoramas and Objects can also be viewed right in a browser.

Using the QuickTime Plug-In

The QuickTime Plug-in works with existing QuickTime movies, and with movies prepared to take advantage of it's 'streaming' feature. This feature will present the first frame of the movie almost immediately, and can begin playing even before the movie has been completely downloaded.

When a Wired movie is created or Exported, it will be prepared for cross-platform or web delivery, and can be viewed by anyone using Macintosh and Windows computers with QuickTime.

It is a good idea to create a QuickTime movie that plays immediately without making the user wait. This requires a data rate that is no greater than the bandwidth of the network connection the user is expected to have when they visit the site.

If the data rate of the movie is higher than the available bandwidth, the user will have to wait until a portion of the movie file has been downloaded before it starts to play.

There are a few things to keep in mind about Wired movies. Wired movies are QuickTime movies, but they have special data in them to tell the movie to do certain things at certain times. So a good rule of thumb is to remember the following:

Wired movies have their data stored in a *sprite track* within the movie. This enables the Movie Player to compress and change the data rate etc, so long as the sprite track is not deleted or moved.

There are two streaming formats: lossless, where no data is lost when it is streaming down to the user's browser, or lossy, where certain bits of data are lost as the movie streams down to the user's browser. Wired movies must use lossless (HTTP) streaming. This is because it is important to not lose any of the interactive data that tells the movie what to do.

Apple has some excellent documentation on their Web site that describes QuickTime movies and Movie Player. Another valuable resource to learn more about QuickTime is the Visual QuickSart Guide entitled ‘QuickTime and MoviePlayer Pro’ for Macintosh and Windows. This is available from PeachPit Press and is written by Judith Stern and Robert Lettieri.

The PeachPit web site is:

`http://www.peachpit.com http://www.peachpit.com`

Embedding QuickTime Movies into a Web Page

For a QuickTime movie to appear on a web page, HTML will need to be inserted into the web page. For a movie to play on a certain page, an embed tag must be included. It can be typed, or made using a WYSIWIG web authoring tool that embeds QuickTime movies into a web page.

EMBED Syntax and Tag Index

The `<EMBED>` tag is used to embed different kinds of contents within an HTML page such as QuickTime movies. When the document specified in the SRC parameter is a QuickTime movie or any media type that the QuickTime Plug-in can display, then the QuickTime Plug-In will be used. As with the rest of HTML, all parameter keywords listed below are case insensitive.

`CACHE= value`

`CACHE` is an optional parameter. Possible values are `TRUE` or `FALSE`, or simply `CACHE (implies TRUE)`. If `CACHE` or `CACHE=TRUE` is specified, the browser will cache movies when possible just like other documents. If the movie is still in the cache when the user returns to the page, it will not need to be downloaded again.

Note:
The functionality of this tag is only supported by Netscape Navigator 3.0 or 4.0. However, the plug-in settings dialog allows for the user to always save the movie to disk cache.

If the `CACHE` is not specified, it defaults to the setting specified by the user in the Preferences dialog.

`VOLUME= value`

`VOLUME` is an optional parameter. Possible values are 0 through 100. A setting of 0 effectively mutes the audio; a setting of 100 is maximum volume. If you do not specify `VOLUME`, it defaults to 100.

`SCALE= value`

`SCALE` is an optional parameter. Possible values are `TOFIT`, `ASPECT`, or a number. If `TOFIT`, the movie is scaled to fit the embedded box as specified by the `HEIGHT` and `WIDTH` tags. If `ASPECT`, the movie is scaled to fit the embedded box but maintain the aspect ratio. If a number, the movie is scaled by that number (e.g. 1.5).

Note:
Using the number scale tag with a QTVR Panorama movie can degrade the performance of the movie even on high-end systems. If `SCALE` *is not specified, its default value is 1.*

`PLUGINSPAGE= url`

`PLUGINSPAGE` is an optional parameter. The `PLUGINSPAGE` parameter is a URL to be specified from which the user can fetch the necessary plug-in if it is not installed. This parameter is handled by a browser. If a browser cannot find the plug-in when loading the page, it will warn the user and allow them to bring up the specified URL, from which one could download the QuickTime Plug-In.

IMPORTANT:
Please set this parameter to: “`http://www.apple.com/quicktime/download/`” *which will point to the latest version of the plug-in. This option is appropriate for both QuickTime movies and QuickTime VR Objects and Panoramas.*

`WIDTH= size in pixels`

The `WIDTH` attribute specifies the width of the embedded document, in pixels. The `WIDTH` parameter is required. Never specify a width of less than 2 as this can cause problems with some browsers. If the width and height is tiny to hide the movie, use the `HIDDEN` tag instead, as explained below. If you don’t know the width of the movie, open your movie with MoviePlayer that comes with QuickTime and select ‘Get Info’ from the Movie menu. Otherwise, use the `SCALE` tag, then supply a width that is smaller than the actual width of the movie and the movie will be cropped to fit the width. If a width that is greater than the width of the movie is supplied, the movie will be centered inside this width.

HEIGHT= size in pixels

The HEIGHT attribute specifies the height of the embedded document, in pixels. To display the movie's controller, add 16 pixels to the HEIGHT. The HEIGHT parameter is required unless the HIDDEN parameter (below) is used. Never specify a height of less than 2 as this can cause problems with some browsers. If the width and height is tiny to hide the movie, use the HIDDEN tag instead, as explained below. If you don't know the height of the movie, open your movie with MoviePlayer and select 'Get Info' from the Movie menu. Otherwise, use the SCALE tag, then supply a width that is smaller than the actual width of the movie and the movie will be cropped to fit the width. If a width that is greater than the width of the movie is supplied, the movie will be centered inside this width.

HIDDEN

HIDDEN is an optional parameter. The HIDDEN parameter controls the visibility of the movie. There are no values to supply for this parameter. If HIDDEN is not supplied, then the movie will be visible. If HIDDEN is supplied, the movie is not visible on the page. This option is not appropriate for QuickTime VR Objects or Panoramas. The HIDDEN setting can only be used to hide a sound-only movie.

If you do not specify HIDDEN, the movie will be visible.

AUTOPLAY= value

AUTOPLAY is an optional parameter. When set to TRUE, the AUTOPLAY parameter causes the movie to start playing as soon as the QuickTime Plug-In estimates that it will be able to play the entire movie without waiting for additional data. Acceptable values for this parameter are TRUE and FALSE.

If AUTOPLAY is not specified, the default is specified by the user setting in the QuickTime Plug-in Preferences.

CONTROLLER= value

CONTROLLER is an optional parameter. The CONTROLLER parameter sets the visibility of the movie controller. Acceptable values for this parameter are TRUE and FALSE . Now there can be a controller with QTVR 2.1 on VR Panarama or Object Movies. If you do not specify CON-TROLLER, the default is TRUE for QuickTime movies. For compatibility with existing web pages, the default is FALSE for QuickTime VR movies.

LOOP= value

LOOP is an optional parameter. When set, the LOOP parameter makes the movie play in a loop. Acceptable values for this parameter are TRUE , FALSE and PALINDROME . Setting LOOP to PALINDROME causes the movie to play alternately forwards and backwards. This option is not appropriate for QuickTime VR Objects and Panoramas.

If you do not specify LOOP, the default is FALSE .

PLAYEVERYFRAME= value

PLAYEVERYFRAME is an optional parameter. When set, the PlayEveryFrame parameter causes the movie to play every frame even if it is necessary to play at a slower rate to do so. This parameter is particularly useful to play simple animations. Acceptable values for this parameter are TRUE and FALSE. This option is appropriate for QuickTime movies. Note: PLAYEVERYFRAME=TRUE will turn off any audio tracks your movie may have. If PLAYEVERYFRAME is not specified, the default is FALSE .

HREF= url

HREF is an optional parameter. When set, the HREF parameter provides a link to another page when the movie is clicked on. Note: if you are using a relative pathname for the HREF then it should be relative to location of the movie specified in the SRC= parameter.

Note that a TARGET frame can be additionally specifed with the HREF parameter.

BGCOLOR= hex value

BGCOLOR is an optional parameter. This option can be used to specify the background color for any space that is not taken by the movie — for example, if a 160x120 movie is embedded in a space of 200 x 120. This would leave 40 pixels of white space in the width. The hex value of the background color used can be specified to match the web page background color. For example:

```
BGCOLOR="#FFFFFF"
```

This would specify a black background color. The first 2 values represent red, the next 2 green and the last 2 blue, giving a total of 16.7 million colors to choose from.

PAN= integer

PAN is an optional parameter. The PAN parameter allows the initial pan angle for a QuickTime VR movie to be specified. The range of values for a typical movie would be 0.0 to 360.0 degrees. This parameter has no meaning for a standard QuickTime movie.

If no value for PAN is specified, the value stored in the movie is used.

TILT= integer

TILT is an optional parameter. The TILT parameter allows the initial tilt angle for a QuickTime VR movie to be specified. The range of values for a typical movie would be -42.5 to 42.5 degrees. This parameter has no meaning for a standard QuickTime movie. If no value for TILT is specified, the value stored in the movie is used.

FOV= integer

FOV is an optional parameter. The FOV parameter allows you to specify the initial field of view angle for a QuickTime VR movie. The range of values for a typical movie would be 5.0 to 85.0 degrees. This parameter has no meaning for a standard QuickTime movie.

If no value is specified for FOV the value stored in the movie is used.

NODE= integer

NODE is an optional parameter. The NODE parameter allows the initial node for a multi-node QuickTime VR movie to be specified.

If no value is specified for NODE, the default NODE and view (specified at creation time of the panorama movie) is used.

CORRECTION= value

CORRECTION is an optional parameter. Possible values are NONE, PARTIAL, or FULL. This parameter is only appropriate for QuickTime VR objects and panoramas.

If no value is specified for CORRECTION, the default correction used is FULL.

Configuring a Web Server for QuickTime

In order to play or serve QuickTime movies from a server, the movies need to have a listing for QuickTime files in its MIME types configuration file. Most Web servers already have this set. If the QuickTime movie is not playing correctly or appearing, check the EMBED tag in the HTML or check to make certain that there is no problem with the server configuration. The MIME type for QuickTime should be as follows:

`‘video/quicktime’` and the suffix should be `‘mov’` .

Minimal HTML to EMBED QuickTime Movies

The following example shows some simple HTML, or a simple EMBED tag with only the required parameters to allow your QuickTime movies to play.

The minimal steps are:

1. If the QuickTime movie does not already have a name ending in .mov, then rename it so it does. The .mov extension is required so that the servers and browsers can recognize the file as a QuickTime file.
2. Open an HTML editor and locate the place in the HTML body where the QuickTime movie is to appear.
3. Type `<EMBED SRC=“demo2.mov”, and replace “demo2.mov”` with the location of the QuickTime movie file.
4. Type a space, then the `WIDTH=w Height=h`, replacing the ‘`w`’ with the width and the ‘`h`’ with the height of the movie.
5. Type a final > to close the EMBED tag.

Example:

```
<EMBED SRC=”Demo2.mov” HEIGHT=100 WIDTH=350>
```

Note!
Add 16 to the height of the movie for the default movie controller unless CONTROLLER=FALSE has been specified.

Telling Web Site Visitors How To Get QuickTime

QuickTime Plugin

In order for web site visitors to play your QuickTime movies, you will need to tell them to get QuickTime and the QuickTime plug-in if it did not come installed on their computers. This will allow them to get the latest version of QuickTime.

Just put a link on your page to Apple's Web site:

```
http://www.apple.com/quicktime/
```

Chapter 7
Using QuickTime Movies with Macromedia Director

Director 6.5

When bringing a QuickTime 3 movie into Director you must use the ‘Insert / Media / Element / QuickTime 3’ command instead of the ‘Import’ command. This is because the import command does not use the QuickTime 3 asset xtra, and instead uses QuickTime 2.x, causing playback on Windows to fail completely.

Ensure that QuickTime 3 is installed on the end-user’s machine by checking:

```
if quickTimeVersion() >= 3 then
  -- go to movie with qt3 content
else
  -- alert, or prompt to install qt3
end if
```

Important Note:
Since releases of Director earlier than 6.5 do not include the QuickTime 3 asset xtra you will not be able to playback and control QuickTime 3 movies.

Director 7

Unlike Director 6.5, you may use the ‘Insert / Media / Element / QuickTime 3’, or the ‘Import’ command to bring a QuickTime 3 movie into Director 7.

Ensure that the ‘direct-to-stage’ is selected in the QuickTime 3 member’s properties dialog box, or set it with a Lingo script. This enables mouse interaction with the movie file.

The movie must not have any other elements in front of it. This is because, with ‘direct to stage’ on, the movie will play back in the front most drawing layer. If something is in front of it, it will either be covered by the movie or produce an ugly flashing.

Since untimed, or static movies made within LiveStage do not play on a timeline like a typical QuickTime movie, they are actually 'paused' already. These movies will work better within Director when it's member '`paused`' property is set to '`true`'. This can be done in either the member's dialog box, or with Lingo.

To enable mouse clicks, It may be necessary to set the `mouselevel of sprite` to `#all`. This is not the case however, for movies that are strictly rollovers.

Director intercepts mouse events if it is not told to ignore them. Setting the mouselevel of a sprite to `#all` makes the movie the only receiver of mouse events within the sprite's boundary. If both Director and the QuickTime movie require mouse events, try `#shared` instead of `#all`.

```
on beginsprite me
  sprite(me.spritenum).mouselevel = #all
end
```

Appendix A: Actions

Movie Actions

`SetVolumeTo(`*`volume`*`) [MIN MAX]`
Sets the sound volume of the movie. Ranges from `0` (off) to `255` (full volume).

`SetVolumeBy(`*`volume`*`) [MIN MAX wraparound]`
Changes the sound volume of the movie by the amount specified. Ranges from `0` (off) to `255` (full volume).

`SetRateTo(`*`rate`*`) [MIN MAX]`
Sets the playback speed of the movie. A speed of `0` indicates stop, `1.0` means go forward at normal speed, `-1.0` means go backward at normal speed ('normal' as specified by the speed in the Document Info settings).

`SetRateBy(`*`rate`*`) [MIN MAX wraparound]`
Changes the playback speed of the movie by the amount specified. A speed of `0` indicates stop, `1.0` means go forward at normal speed, `-1.0` means go backward at normal speed ('normal' as specified by the speed in the Document Info settings).

`SetLoopingFlags(`*`flags`*`)`
Sets the looping mode for the movie. Valid values are `kNoLoop`, `kLoop`, or `kLoopPalindrome`.

`GoToTime(`*`time`*`)`
This action causes the movie to go to a particular time.

`GoToTimeByName(`*`time_name`*`)`
This action tells the movie to go to the specified chapter as defined in `(time_name)`.

`GoToBeginning`
This action tells the movie to go to the beginning or the start frame. This will re-execute any frame-loaded events.

GoToEnd

This action tells the movie to go to the end or the last frame.

StepForward

This action tells the movie to step forward a bit.

StepBackward

This action tells the movie to step backward a bit.

SetSelection(start_time, end_time)

This action sets the selection of the movie to the time range specified in `(start_time, end_time)`.

SetSelectionByName(start_name, end_name)

This action sets the movie's selection to the time range specified by chapter names as defined in `(start_name, end_name)`.

SetPlaySelection(*selection_only*)

If `selection_only` is true, then the movie will play only the current selection.

TogglePlaySelection

Reverses the current setting, that is set by calling `SetPlaySelection()`.

SetLanguage(*language*)

Sets the current language of the movie to the `language`. This is a numerical value (0-66) as outlined in Inside Macintosh, Chapter 6, Page 108.

Some examples include:

```
langEnglish = 0
langFrench = 1
langGerman = 2
langItalian = 3
langDutch = 4
langSwedish = 5
langSpanish = 6
```

```
langNorwegian = 9
langJapanese = 11
```

Note: In order to add another language track, the track must be created using the new language.

Track Actions

`SetEnabled(`*`enable`*`)``-(True / False)`

Sets the enabled state of the track. Tracks such as a sound tracks that change the sound being played in the movie can be enabled and disabled, or an overlaid video track can be made invisible

`ToggleEnabled`

Reverses the enabled state of the track.

Spatial Track Actions

`SetLayerTo(`*`layer`*`) [MIN MAX]`

Sets the layer of the track. The smaller the number for the layer, the closer it is to the front, or the top.

`SetLayerBy(`*`layer`*`) [MIN MAX wraparound]`

Changes the layer of the track by the amount specified. The smaller the number for the layer, the closer it is to the front, or the top.

Sound Track Actions

LiveStage and QuickTime are well suited to managing and using audio for interactive movie development. These sound track actions allow control of a number of key aspects of sound playback with scripting.

`SetVolumeTo(`*`volume`*`) [MIN MAX]`

Sets the sound volume of the track by the amount specified in (volume). Ranges from `0` (off) to `255` (full volume).

`SetVolumeBy(`*`volume`*`) [MIN MAX wraparound]`

Changes the sound volume of the track by the amount specified in (volume). Ranges from `0` (off) to `255` (full volume).

SetBalanceTo(*volume*) [MIN MAX]
Sets the balance of the sound track from left to right by the amount specified in `(volume)`. Ranges from `–128` (left) to `128` (right).

SetBalanceBy(*volume*) [MIN MAX wraparound]
Changes the balance of the sound track from left to right by the amount specified in `(volume)`.

Sprite Actions

SetImageIndexTo(*index*) [MIN MAX]
Sets the current image of the sprite to the image at the index specified in `(index)`. Sprite images are stored in the order they appear in the Sprites Object window. The Index of a Sprite Object's image is stored in the pool of shared images in the movie. To access the index of a Sprite Object image, click on the Images Tab in the Object Window. The image index will be listed in the Image list pane in order.

SetImageIndexBy(*index*) [MIN MAX wraparound]
Changes the current image index of the sprite by the amount specified in `(index)`.

SetVisible(*visible*) `- (True / False)`
Sets the visible state of the sprite. If the state is `true` then you can see the sprite.

ToggleVisible
Reverses the visible state of the sprite.

SetLayerTo(*layer*) [MIN MAX]
Sets the layer of the sprite to a number specified in (layer). The smaller the number specified for the layer, the closer it is to the front or the top.

SetLayerBy(*layer*) [MIN MAX wraparound]
Changes the layer of the sprite by the amount specified in (layer). The smaller the number specified for the layer, the closer it is to the front or the top.

SetGraphicsModeTo(*mode,red,green,blue*) [MIN MAX]
Sets the graphics mode of the sprite. Valid (modes) are: `srcCopy`, `srcOr`, `srcXor`, `srcBic`, `notSrcCopy`, `notSrcOr`, `notSrcXor`, `notSrcBic`, `blend`, `addPin`, `addOver`, `subPin`, `transparent`, `addMax`, `subOver`, `adMin`, `grayishTextOr`, `hilite`, `ditherCopy`.

Some modes make use of the `red`,`green`,`blue` settings. These modes are described in Draw Modes.

`SetGraphicsModeBy(`*`mode,red,green,blue`*`) [MIN MAX wraparound]`
This action changes the graphics mode of the sprite by the amount specified in (`mode, red, green, blue).`

`PassMouseToCodec`
When using the ripple codec, this action will cause the Sprite Object to ripple.

`ClickOnCodec(`*`x,y`*`)`
This action passes the mouse click to the codec for it to interpret. For instance, the Ripple codec will ripple from the codec.

`MoveTo(`*`x,y`*`)`
This action causes the sprite to move to the `x,y` coordinate on the screen.

`MoveBy(`*`x,y`*`)`
This action causes the sprite to move to the right by the number of pixels specified in `(x)`, and down by the number of pixels specified in `(y)`.

`Scale(`*`x,y`*`)`
This action tells the sprite to enlarge by the amount specified in `(x)` and `(y)`.

`Rotate(`*`degrees`*`)`
This action tells the sprite to rotate by the specified amount in `(degrees)`. The amount of degrees is `0` to `360`, or `-1` to `-359`.

`Stretch(p1x,p1y,p2x,p2y,p3x,p3y,p4x,p4y)`
This action warps the sprite's image so that its four corners are matched up with the points specified in `(p1x, p1y, p2x, p2y, p3x, p3y, p4x, p4y)`. The points are in clockwise order; top left, top right, bottom right, bottom left. There are also some undocumented limits in QuickTime's handling of angles; if the polygon stretched to, has angles sharper than some indeterminate limit, the stretched sprite will disappear.

`ExecuteEvent(`*`Event_id`*`)`
This action causes the sprite to execute the custom Event specified in `(Custom Event…)`.

VR Track Actions

`SetPanAngleTo(`*`angle`*`) [MIN MAX]`

This action sets the pan angle in the movie to the angle specified in (angle).

`SetPanAngleBy(`*`angle`*`) [MIN MAX wraparound]`

This action changes the pan angle of the VR track in the movie by the angle specified in `(angle)`.

`SetTiltAngleTo(`*`angle`*`) [MIN MAX]`

This action sets the tilt angle of the VR track in the movie to the angle specified in `(angle)`.

`SetTiltAngleBy(`*`angle`*`) [MIN MAX wraparound]`

This action sets the tilt angle of the VR track in the movie by the angle specified in `(angle)`.

`SetFieldOfViewTo(`*`fov`*`) [MIN MAX]`

This action sets the field of view of the VR track in the movie to the number specified in `(fov)`.

`SetFieldOfViewBy(`*`fov`*`) [MIN MAX wraparound]`

This action changes the field of view by the amount specified in `(fov)`.

`ShowDefaultView`

This action returns the track to the default view.

`GoToNode(`*`node_id`*`)`

This action sends the track to the specified VR node.

Music Track Actions

Music Track Actions manipulate the QuickTime music architecture (QTMA), which allows QuickTime movies to play individual musical notes, sequences of notes, and a broad range of sounds from a variety of instruments and synthesizers.

`PlayNote(instrument_Index,delay,pitch,velocity,duration)`

This causes the target music track to play a note. The instrument is the index of the custom instrument sound as shown in the Objects window in the Sounds tab.

Parameters

`instrument_Index`

The `instrument_Index` parameter specifies which instrument to use within the instrument track.

`delay`

If you want the note to be delayed, you may pass a positive value for the delay parameter, which is interpreted in the time scale (1/600th of a second) of the music track. Pass 0 for no delay.

`pitch`

The `pitch` parameter selects which note to play; for example, Middle C is `60`, Middle B is `59`. (`32` to `95`)

`velocity`

The `velocity` specifies the volume that the note is played at. (`0` to `255`)

`duration`

The `duration` specifies the length of time that the note is played for and is interpreted in the time scale (1/600th of a second) of the music track. (`0` to `999999`)

Note:
By setting duration to 999999, the note will continue playing until you `PlayNote` *again for the same* `pitch` *with a* `velocity` *of* `0`*. This allows you to hold a note for an interactive period of time. For example, on* `mouseDown`*, you play a note, and on* `mouseUp` *to turn it off.*

`SetController(Sample,Instrument_Index,Delay,Controller,Value)`

Sets a controller value for a part in the target music track.

Parameters:

`Sample`

Since the current selection of instruments for a music track is determined by its sample description, the `Sample` parameter is used to select which sample description to use.

`Instrument_Index`

The `Instrument_Index` parameter specifies which part to use within the sample descriptions list.

`Delay`

If the controller change is to be delayed, a positive value for the `delay` parameter may be passed, which is interpreted in the time scale of the music track. Pass `0` for no delay.

`Controller` **&** `Value`

Controller values control items such as `pitch bend` 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.

Controller Descriptions:

`kControllerModulationWheel (1)`

This controls the modulation wheel. A modulation wheel adds a periodic change to the volume or pitch of a sounding tone, depending on the modulation depth knobs.

`kControllerBreath (2)`

This controls breath.

`kControllerFoot (4)`

This controls the foot pedal.

`kControllerPortamentoTime (5)`

This adjusts the slur between notes. Set the time to o to turn off portamento; there is no separate control to turn portamento on and off.

`kControllerVolume (7)`

This controls volume.

`kControllerBalance (8)`

This controls balance between channels.

`kControllerPan (10)`

This controls balance on the QuickTime music synthesizer and some others. Values are 256–512, corresponding to left to right.

`kControllerExpression (11)`

This provides a second volume control.

`kControllerLever1 (16) through kControllerLever4 (19)`

`kControllerLever5 (80) through kControllerLever8 (83)`

General-purpose controllers.

`kControllerPitchBend (32)`

This controller bends the pitch. Pitch bend is specified in positive and negative semitones, with 7 bits per fraction.

`kControllerAfterTouch (33)`

This controller controls channel pressure.

`kControllerSustain (64)`

This controller controls the sustain effect. The value is a Boolean—positive for on, `0` or negative for off.

`kControllerSostenuto (66)`

This controller controls sostenuto.

`kControllerSoftPedal (67)`

This controller controls the soft pedal.

`kControllerReverb (91)`

This controller controls reverb.

`kControllerTremolo (92)`

This controller controls tremolo.

`kControllerChorus (93)`

This controller controls the amount of signal to feed to the chorus special effect unit.

`kControllerCeleste (94)`

This controller controls the amount of signal to feed to the celeste special effect unit.

`kControllerPhaser (95)`

This controller controls the amount of signal to feed to the phaser special effect unit.

`kControllerEditPart (113)`

This controller sets the part number for which editing is occurring. For synthesizers that can edit only one part.

`kControllerMasterTune (114)`

This controller offsets the entire synthesizer in pitch.

Controller Range

These constants specify the maximum and minimum values for controllers.

`kControllerMaximum (0x7FFF)`

The maximum value a controller can be set to.

`kControllerMinimum (0x8000)`

The minimum value a controller can be set to.

General (no target needed)

`GotoURL("`*`url`*`")`

This action tells the web browser to go to the specified URL. If the movie is not executing in a web browser, the user's default web browser will be launched. To specify a URL, use the standard web address format – for example, `("http://www.totallyhip.com")`.

You may optionally use angle brackets–for example, `<http://www.totallyhip.com>`.

To specify a particular frame within a URL, angle brackets must be used, followed by `<frameName>` – for example, `<http://www.totallyhip.com>T<frameName>`.

`ApplicationNumberAndString(`*`num, "string"`*`)`

This action sends the number and string to the application playing the movie.

`DebugStr("`*`string`*`")`

Sends the string to the debug window within LiveStage.

`PushCurrentTime`

This action saves the current time of the movie at the specified mark.

`PushCurrentTimeWithLabel("`*`string`*`")`

This action saves the current time of the movie and labels it to a string specified in `("string")`.

`PopAndGotoTopTime`

This action goes to the last saved time.

`PopAndGotoLabeledTime("`*`string`*`")`

This action goes to the last saved time that matches the label as specified in `("string")`.

Appendix B: Targets for Actions

`ThisMovie`

Specifies the current movie as the target explicitly.

`ThisTrack`

Specifies the current track as the target explicitly.

`ThisSprite`

Specifies the current sprite as the target explicitly.

`TrackNamed("name")`

Specifies the track with the name provided as the current target. The user must provide the name of the track. The sprite track is named 'Sprites', the music track containing your custom instrument sounds is named 'Instruments' and the sound tracks have the same name as the sound file.

`TrackOfType(`*`type`*`)`

Specifies the track matching the type provided as the target. The user must provide the type of track.

`TrackOfIndex(index)`

Specifies the track at the index provided as the target. The user must provide the index of the track. The sprite track is always index 1 and is followed by the instrument track if needed, and then followed by the sound tracks.

`TrackOfID(id)`

Specifies the track with the ID provided as the target. The user must provide the ID of the track. The sprite track is always ID 1, and is followed by the instrument track if needed, and then followed by the sound tracks. In general, the ID and index are the same.

`SpriteNamed(`*`"name"`*`)`

Specifies the Sprite with the name provided as the target. The user provides the name of the sprite.

`SpriteOfIndex(index)`

Specifies the Sprite at the index provided as the target. The program provides the index of the sprite. Sprites are stored and indexed in the same order as they appear in the Sprite Objects window.

`SpriteOfID(id)`

Specifies the Sprite with the ID specified as the target. The user provides the ID of the sprite. Each sprite has a unique ID and it need not have the same value as the sprite's index. The index of the sprite may change, but the ID will stay the same unless the user explicitly changes it.

Appendix C: Properties

Movie Properties

`MovieVolume`
Returns the current volume level.

`MovieRate`
Returns the current playback rate as specified by the speed in the Document Info settings.

`MovieIsLooping`
Returns `true` if the movie is looping or `false` otherwise.

`MovieLoopIsPalindrome`
Returns `true` if the movie is set to loop in palindrome mode.

`MovieTime`
Returns the current time of the movie.

Track Properties

`TrackEnabled`
Returns `true` if the `track` is enabled or `false` otherwise.

Spatial Track Properties

`TrackLayer`
Returns the layer of the track.

`TrackWidth`
Returns the width of the track in pixels.

`TrackHeight`
Returns the height of the track in pixels.

`MouseHorizontal`
Returns the horizontal location of the mouse in pixels from the left origin.

`MouseVertical`
Returns the vertical location of the mouse.

Sound Track Properties

`TrackVolume`
Returns the volume setting for the Sound track. `0` (off) to `255` (full).

`TrackBalance`
Returns the balance of the Sound track. –`128` (left only) to `128` (right only).

Sprite Track Properties

`NumSprites`
Returns or retrieves the number of sprites that currently exist in the sprite track of the movie.

`NumImages`
Returns or retrieves the number of images that currently exist in the sprite track of the movie.

VR Track Properties

`PanAngle`
Returns the current pan angle in a VR track.

`TiltAngle`
Returns the current tilt angle in a VR track.

`FieldOfView`
Returns the current field of view in a VR track.

`NodeID`
Returns the current node ID in a VR track.

Sprite Properties

`BoundsLeft`

Returns the pixel position of the left edge of the sprite.

`BoundsTop`

Returns the pixel position of the top edge of the sprite.

`BoundsRight`

Returns the pixel position of the right edge of the sprite.

`BoundsBottom`

Returns the pixel position of the bottom edge of the sprite.

`ImageIndex`

Returns the index of the current image in the sprite.

`IsVisible`

Returns `true` if the sprite is visible or `false` otherwise.

`Layer`

Returns the layer number of the currently displayed sprite.

`ID`

Returns the unique ID of the currently displayed sprite.

`Index`

Returns the index of the sprite. This is the Image Index number used in the Objects Window Image Tab list.

`FirstCornerX`

Returns the `x` coordinate of the first corner (top left) of the sprite.

`FirstCornerY`

Returns the `y` coordinate of the first corner (top left) of the sprite.

`SecondCornerX`

Returns the `x` coordinate of the second corner (top right) of the sprite.

`SecondCornerY`

Returns the `y` coordinate of the second corner (top right) of the sprite.

`ThirdCornerX`

Returns the `x` coordinate of the third corner (bottom right) of the sprite.

`ThirdCornerY`

Returns the `y` coordinate of the third corner (bottom right) of the sprite.

`FourthCornerX`

Returns the `x` coordinate of the fourth corner (bottom left) of the sprite.

`FourthCornerY`

Returns the `y` coordinate of the fourth corner (bottom left) of the sprite.

`ImageRegistrationPointX`

Returns the `x` coordinate of the registration point of the sprite.

`ImageRegistrationPointY`

Returns the y coordinate of the registration point of the sprite.

Note!
Sprite images have a default registration point of `0,0` *with the origin being from the top left.*

General Properties

`KeyIsDown(modifiers,'key')`

Returns `true` if the key and modifiers are pressed or `false` otherwise. Modifiers can be `kOptionKey`, `kShiftKey`, `kCommandKey` or `kNone`.

Modifiers can be combined by separating them with the vertical bar ‘|’.

```
KShiftKey|kOptionKey
```

Specify the key using a lowercase letter. For example, to detect the ‘a’ key and the ‘shift’ key use

```
KeyIsDown(kShiftKey,'a')
```

`Random(`*`min,max`*`)`

Returns a number between min and max inclusive. These cannot be variables.

`MouseButtonDown`

Returns `true` if the mouse button is currently being pressed.