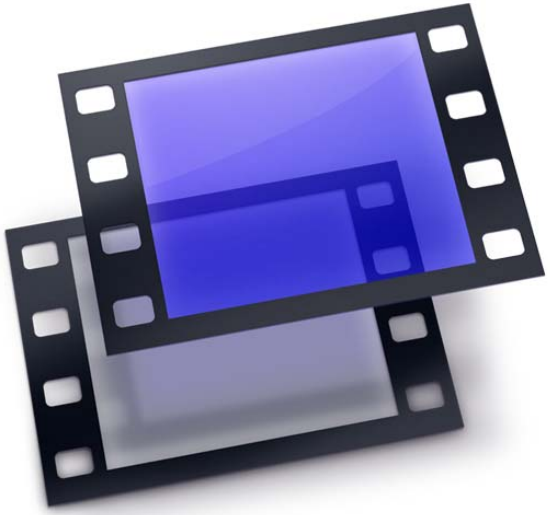
Shake 4
User Manual



Apple Computer, Inc.
1 Infinite Loop
Cupertino, CA 95014-2084
408-996-1010
www.apple.com

distributed, then this README file must be included, with this copyright and no-warranty notice unaltered; and any additions, deletions, or changes to the original files must be clearly indicated in accompanying documentation. (2) If only executable code is distributed, then the accompanying documentation must state that this software is based in part on the work of the Independent JPEG Group. (3) Permission for use of this software is granted only if the user accepts full responsibility for any undesirable consequences; the authors accept NO LIABILITY for damages of any kind. These conditions apply to any software derived from or based on the IJG code, not just to the unmodified library. If you use our work, you ought to acknowledge us. Permission is NOT granted for the use of any IJG author's name or company name in advertising or publicity relating to this software or products derived from it. This software may be referred to only as the Independent JPEG Group's software. We specifically permit and encourage the use of this software as the basis of commercial products, provided that all warranty or liability claims are assumed by the product vendor.

Sam Leffler and Silicon Graphics, Inc. (TIFF library):
© 1988-1996 Sam Leffler. Copyright © 1991-1996 Silicon Graphics, Inc.

Permission to use, copy, modify, distribute, and sell this software and its documentation for any purpose is hereby granted without fee, provided that (i) the above copyright notices and this permission notice appear in all copies of the software and related documentation, and (ii) the names of Sam Leffler and Silicon Graphics may not be used in any advertising or publicity relating to the software without the specific, prior written permission of Sam Leffler and Silicon Graphics.
© THE SOFTWARE IS PROVIDED AS-IS AND WITHOUT WARRANTY OF ANY KIND, EXPRESS, IMPLIED OR OTHERWISE, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL SAM LEFFLER OR SILICON GRAPHICS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES OF ANY KIND, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER OR NOT ADVISED OF THE POSSIBILITY OF DAMAGE, AND ON ANY THEORY OF LIABILITY, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Photron USA, Inc. (Primatte Keyer): © 2004 Photron, USA

Glen Randers-Pehrson, et al. (png): libpng version 1.0.8 - July 24, 2000. © 1998-2000 Glenn Randers-Pehrson, © 1996, 1997 Andreas Dilger, © 1995, 1996 Guy Eric Schalnat, Group 42, Inc.

COPYRIGHT NOTICE, DISCLAIMER, and LICENSE
For the purposes of this copyright and license, Contributing Authors is defined as the following set of individuals: Andreas Dilger, Dave Martindale, Guy Eric Schalnat, Paul Schmidt, Tim Wegner.

The PNG Reference Library is supplied AS IS. The Contributing Authors and Group 42, Inc. disclaim all warranties, expressed or implied including, without limitation, the warranties of merchantability and of fitness for any purpose. The Contributing Authors and Group 42, Inc. assume no liability for direct, indirect, incidental, special, exemplary, or consequential damages, which may result from the use of the PNG Reference Library, even if advised of the possibility of such damage.

Permission is hereby granted to use, copy, modify, and distribute this source code, or portions hereof, for any purpose, without fee, subject to the following restrictions: 1. The origin of this source code must not be misrepresented. 2. Altered versions must be plainly marked as such and must not be misrepresented as being the original source. 3. This Copyright notice may not be removed or altered from any source or altered source distribution. The Contributing Authors and Group 42, Inc. specifically permit, without fee, and encourage the use of this source code as a component to supporting the PNG file format in commercial products. If you use this source code in a product, acknowledgment is not required but would be appreciated.

Julian R. Seward (bzip2): © 1996-2002 Julian R Seward. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required. 3. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software. 4. The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Julian Seward, Cambridge, UK.
jseward@acm.org
bzip2/libbzip2 version 1.0.2 of 30 December 2001

Contents

Shake 4 Documentation and Resources

Welcome to the world of Shake 4 compositing. This chapter covers where to find help, how the keyboard and mouse work on different platforms, and how to set up Shake for use with a stylus.

What Is Shake?

Shake is a high-quality, node-based compositing and visual effects application for film and video. Shake supports most industry-standard graphics formats, and easily accommodates high-resolution and high bit depth image sequences and QuickTime files (Mac OS X only).

Among Shake's many built-in tools are industry-standard keyers for pulling bluescreens and greenscreens, a complete suite of color-correction tools, features for high-quality motion retiming and format remastering, motion tracking, smoothing, and stabilization capabilities, integrated procedural paint tools, and a rotoscoping and masking environment that provides complete control over animated and still mattes. Shake also supports an extensive list of third-party plug-ins, and is compatible across both the Mac OS X and Linux platforms.

Shake is also an image-processing tool that can be used as a utility for media being passed along a pipeline of many different graphics applications. Large facilities can use Shake to process and combine image data from several different departments—for example, taking a project from initial film recording; providing processed images and tools for use by the 3D animation, digital matte, and roto departments; recombining the output from all these groups with the original plates for compositing; and ultimately sending the final result back out for film recording.

Shake's tools can be accessed in several different ways. While most artists work within the graphical interface, advanced users can access a command-line tool running from the Terminal. Likewise, more technically oriented users can perform complex image processing by creating scripts (the Shake scripting language is similar to C), thereby using Shake as an extensive image-manipulation library.

Using the Shake Documentation

There are several components to the documentation accompanying Shake, including printed user manuals and tutorials, onscreen documentation in PDF and HTML formats, and contextual help available directly from within the Shake interface.

User Manual

The *Shake 4 User Manual* is divided into two volumes:

- *Volume I—The Interface:* Explains the basics of the Shake interface and provides instructions for working with media, file formats, nodes, and so on.
- *Volume II—Compositing:* Discusses the specific features Shake provides for image compositing. Part I of this volume covers such topics as image processing, rotoscoping, color correction, and so on. Part II delves into Shake's advanced functionality, including optimizing, creating macros, and using expressions. This section also includes "The Cookbook," a repository of useful Shake tips and techniques.

Tutorials

If you are new to Shake, you are encouraged to work through the *Shake 4 Tutorials.* These interactive lessons provide you with a solid introduction to Shake's functionality and workflow.

Onscreen Help

Onscreen help (available to Mac OS X users in the Help menu) provides easy access to information while you're working in Shake. Onscreen versions of the *Shake 4 User Manual* and *Shake 4 Tutorials* are available here, along with other documents in PDF format and links to websites.

To access onscreen help in Mac OS X:

- In Shake, choose an option from the Help menu.

Note: You can also open PDF versions of the user manual and tutorials from the *Shake/doc* folder.

Viewing Shake Onscreen Documentation on Linux Systems

To view Shake onscreen documentation on a Linux system, you'll need to download and install Adobe Acrobat Reader, then configure the PDF browser path in the Shake application.

To configure the PDF browser path in Shake:

1. Open the Globals tab.
2. Open the guiControl subtree (click the "+" sign).

   The subtree expands.

3 Click the folder icon next to the pdfBrowser Path parameter.

The Choose Application window appears.

4 In the Choose Application window, browse to and select the Adobe Acrobat Reader application.

To save your PDF browser settings in Shake:

1 Choose File > Save Interface Settings.

The "Save preferences to" window appears.

2 In the "Save preferences to" window, save your settings to a *defaultui.h* file.

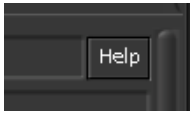
Contextual Help

In addition to the onscreen help, the Shake interface provides immediate contextual help from within the application. Moving the pointer over most controls in Shake displays their function in the Info field, located at the bottom-right side of the Shake interface. The Info field provides immediate information about each control's function. For example, moving the pointer over the Warp tool tab displays the following information in the Info field.

In addition to the information available from the Info field, each node in Shake has a corresponding HTML-based contextual help page, available via a special control in the Parameters tab.

To display a node's contextual help page:

- Load a node's parameters into the Parameters tab, then click the Help button to the right of the node name field.



Note: Contextual help pages are opened using your system's currently configured default web browser.

Apple Websites

There are a variety of discussion boards, forums, and educational resources related to Shake on the web.

Shake Websites

The following websites provide general information, updates, and support information about Shake, as well as the latest news, resources, and training materials.

For more information about Shake, go to:

- http://www.apple.com/shake

To get more information on third-party resources, such as third-party tools and user groups, go to:

- http://www.apple.com/software/pro/resources/shakeresources.html

An useful listserver, archive, and extensive macro collection are accessible at the unofficial Shake user community site, HighEnd2D.com:

http://www.highend2d.com/shake

For more information on the Apple Pro Training Program, go to:

- http://www.apple.com/software/pro/training

Keyboard and Mouse Conventions on Different Platforms

Shake can be used on the Mac OS X and Linux platforms. Functions or commands that are platform-specific have been documented whenever possible. This section summarizes the main differences.

- *Keyboard:* Hot keys or keyboard commands that vary between the Macintosh and Linux platforms are documented when possible. In most cases, the Command and Control keys are interchangeable. The Macintosh Delete key located below the F12 key is the equivalent of the Linux Backspace key; the Macintosh Delete key grouped with the Help, Home, and End keys is the equivalent of the Linux Delete key.

  Important: Macintosh users should remember that the Delete key used in Shake is *not* the key located below the F12 key but, rather, the one grouped with the Help, Home, and End keys.

- *Mouse:* Shake requires the use of a three-button mouse. A three-button mouse provides quick access to shortcut menus and navigational shortcuts. Shake also supports the middle scroll wheel of a three-button mouse.

Shake documentation refers to the three mouse buttons as follows:

Mouse Button	Documentation Reference
Left mouse button	Click
Middle mouse button	Middle mouse button or middle-click
Right mouse button	Right-click

Note: This manual uses the term "right-click" to describe how to access shortcut menu commands.

The following table lists the user manual notation system.

Notation	Example
Hot keys/keyboard commands	To break a tangent handle in the Curve Editor, Control-click the handle.
Some hot keys/keyboard commands vary depending on the platform. The Mac OS X command appears first, followed by the Linux command. The two hot keys/commands are separated by a forward slash. In general, the Command and Control keys are interchangeable.	In the Node View, you can press Control-Option-click / Control-Alt-click to zoom in and out.
Menu selections are indicated by angle brackets.	To open a script, choose File > Open Script.
File paths and file names appear in italics. Also, directories and file paths are divided by forward slashes.	Temp files are saved in the *..//var/tmp/* directory.
Node groups (Tool tabs) appear in the default font, followed by the name of the node in italics. A dash appears between the tab and node names.	In the Node View, select the *Cloud* node, and insert a Transform–*CornerPin* node.
Command-line functions appear in italics.	*shake -exec my_script -t 1-240*
Modifications to preferences files appear in italics.	Add the following lines to a .h file in your startup directory: *script.cineonTopDown = 1;* *script.tiffTopDown = 1;*

Using a Stylus

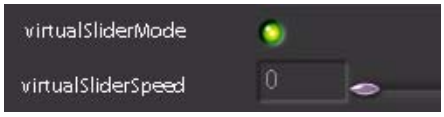
Shake is designed to be used with a graphics tablet and stylus.

To optimize the Shake interface for use with a tablet and stylus:

1. In the guiControls subtree of the Globals tab, enable virtualSliderMode.
2. Set the parameter virtualSliderSpeed to 0.

When virtualSliderMode is enabled, the left button always uses the virtual sliders when when you click a value field. Normally, you have to press Control and drag. However, when virtualSliderMode is on, dragging left or right in a value field adjusts the value beyond normal slider limits.

Note: The stylus does not allow you to use your desk space the same way as with a mouse; consequently, you have to enable virtualSliderMode.



Window Navigation Using a Stylus

Shake makes extensive use of the middle-mouse button to facilitate navigation within each tab of the interface. To navigate and zoom within Shake easily using a stylus, you should map the middle mouse button to one of the stylus buttons. Once mapped, you can use that button to pan around within any section of the Shake interface, or Control-click and drag with that button to zoom into and out of a section of the interface.

Using Dual-Head Monitors

You can choose View > Spawn Viewer Desktop to create a new Viewer window that floats above the normal Shake interface. You can then move this Viewer to a second monitor, clearing up space on the first for node editing operations.

Important: This technique only works when both monitors are driven by the same graphics card.

Part I: Interface, Setup, and Input

Part I presents information about the Shake graphical user interface as a whole, with detailed information about all the major interface components.

An Overview of the Shake User Interface

1

This chapter provides a fast introduction to all aspects of the Shake graphical user interface. It also provides in-depth information about navigating the interface, and customizing it to suit your needs.

Opening Shake

When you open the Shake interface, a blank Shake *script* appears. Shake scripts (otherwise known as project files) are unique in that they're actually a text document containing the command-line script representation of the node tree that you assemble in the interface. You can open Shake scripts in any text editor to examine their contents, and if you're a power user, you can make modifications to your composite right within the text of the script itself (this is only recommended if you're conversant with Shake's scripting language, covered in more detail in Part III of this book).

Most of the time, however, you'll likely stay within Shake's graphical interface, which provides specialized controls for performing a wide variety of compositing tasks (many of which would be far too unwieldy to manipulate from the command line).

Opening Two Scripts at Once

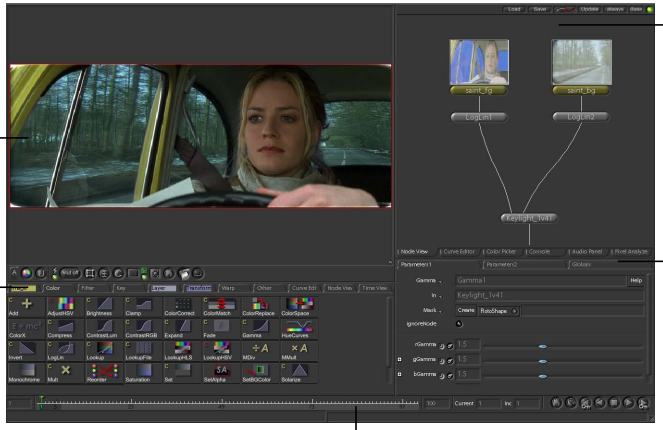
Shake is designed to have only one script open at a time. Typically, each script is used to create a single compositing project, with a single frame range and a single node tree. Although Shake supports multiple independent node trees within the same script, all trees share the same duration, defined by the timeRange parameter in the Globals tab.

If necessary, it is possible to open two scripts simultaneously into interface windows. In this case, what you're really doing is launching two instances of Shake at once. This is primarily useful if you need to copy information from one script to another.

Important: When youopen Shake twice, the first instance of Shake is the only one that's able to write to and read from the cache. (For more information on caching in Shake, see Chapter 13, "Image Caching," on page 343.)

Overview of the Shake User Interface

The Shake user interface is divided into five main areas: the Viewer, the Tool tabs, the Parameters/Globals tabs, the Node View/Curve Editor/Color Picker/Audio Panel/Pixel Analyzer tabs, and the Time Bar at the bottom.



Viewer area

Displays the image at the selected node in the node tree.

Tool tabs

All of the available nodes in Shake are organized into eight tabs. Click a node's icon to add that node to the node tree.

Node View

One of many tabs that can be displayed here. The Node View displays the node tree, which defines the flow and processing of image data in your project.

Parameters tabs

The parameters of selected nodes appear in the Parameters1 and 2 tabs. The global parameters of your project appear in the Globals tab.

Time Bar area

Lets you navigate among the frames of your project using the playback buttons and the playhead.

Images from *The Saint* provided courtesy of Framestore CFC and Paramount British Pictures Ltd.

Node View

The *Node View* is the heart of Shake, and displays the tree of connected nodes that modify the flow of image data from the top of the tree down to the bottom. Every function in Shake is represented as a separate node that can be inserted into the node tree. You use the Node View to modify, select, view, navigate, and organize your composite.

For more information on the Node View, see Chapter 7, "Using the Node View," on page 217.

Viewer Area

The Viewer area is capable of containing one or more *Viewers,* which display the image of the currently selected node. You have explicit control over which part of the node tree is displayed in the Viewer—in fact, the ability to separate the node that's displayed in the Viewer from the node being edited in the Parameters tabs is central to working with Shake. Each Viewer allows you to isolate specific channels from each image. For example, you can choose to view only the red channel of an image while you make a color correction, or only the alpha channel when you're adjusting a key.

Tool Tabs

The *Tool tabs* contain groups of nodes, organized by function. Nodes you click in these tabs are added to the node tree. For example, to add a *Keylight* node, click the Key tool tab, and click the *Keylight* node. The *Keylight* node then appears in the node tree. If you right-click a node in any of the Tool tabs, you can choose to insert that node into the node tree in a variety of different ways, using the shortcut menu.

The Tool tabs area can also display the Curve Editor, Node View, or Time View.

The Time Bar Area

The *Time Bar area*, at the bottom of the Shake window, displays the currently defined range of frames. Three fields to the right of the Time Bar show the displayed number of frames in the Time Bar (not the time range), the current position of the playhead, and the Increments (Inc) in which the playhead moves. To the right of these fields, the Viewer playback controls let you step through your composite in different ways.

Command and Help Lines

Underneath the Time Bar area are two additional fields. The Command Line field lets you enter Shake script commands directly, effectively bypassing the graphical interface. The Info field provides immediate information about interface controls that you roll the pointer over.

Parameters Tabs

The two *Parameters* tabs can be set to display the parameters within a selected node. You can load two different sets of parameters into each of the two Parameters tabs. The *Globals* tab to the right contains the parameters that affect the behavior of the entire script (such as proxy use, motionBlur, and various interface controls).

Curve Editor

The *Curve Editor* is a graph on which you view, create, and modify the animation and Lookup curves that are associated with parameters in the nodes of your script. In addition to adding and editing the control points defining a curve's shape, you can change a curve's type, as well as its cycling mode.

For more information on using the Curve Editor, see Chapter 10, "Parameter Animation and the Curve Editor."

Color Picker

The *Color Picker* is a centralized interface that lets you assign colors to node color parameters by clicking the ColorWheel and luminance bar, clicking swatches from a color palette, or by defining colors numerically using a variety of color models. You can also store your own frequently-used color swatches for future use in the Palette.

For more information on how to use the Color Picker, see "Using the Color Picker" on page 620.

Audio Panel

The *Audio Panel* lets you load AIFF and WAV audio files for use by your project. Several different files can be mixed down to create a single file. The audio waveforms can be displayed inside the Curve Editor. Sound playback can be activated in the Time Bar playback controls (Mac OS X only).

Note: Because audio playback is handled through the use of Macintosh-specific QuickTime libraries, you can only hear audio playback on Mac OS X systems. You can still analyze and visualize audio in Linux.

For more information on the Audio Panel, see Chapter 9, "Using the Audio Panel," on page 277.

Pixel Analyzer

The *Pixel Analyzer* is a tool to find and compare different color values on an image. You can examine minimum, average, current, or maximum pixel values on a selection (that you make), or across an entire image.

For more information on how to use the Pixel Analyzer, see "Using the Pixel Analyzer" on page 627.

Console

The *Console tab* displays the data that Shake sends to the OS while in operation. It's a display-only tab. Two controls at the top of the Console tab let you change the color of the text, and erase the current contents of the console. The maximum width of displayed text can be set via the consoleLineLength parameter, in the guiControls subtree of the Globals tab.

Getting Help in Shake

There are three ways you can get more information about the Shake interface:

- As you pass the pointer (no need to click) over a node or Viewer, information for the node appears either in the title bar of the Viewer, or in the bottom-right Info field. The displayed information includes node name, type, resolution, bit depth, and channels.
- You can also right-click most buttons to display a pop-up menu listing that button's options. You can use this to select a function or to find out what a button does.
- The Help menu contains detailed information on how to use Shake, including the full contents of this user manual, specifics on new features introduced with the current release, and late-breaking news about last-minute changes and additions made to Shake.

Making Adjustments to the Shake Window

As you work with Shake, there are several methods for resizing and customizing the various areas of the Shake interface.

To resize any area of the interface:

- Position the pointer at any border between interface areas and drag to increase or decrease the size of that area. If you drag an intersection, you can resize multiple areas at once.

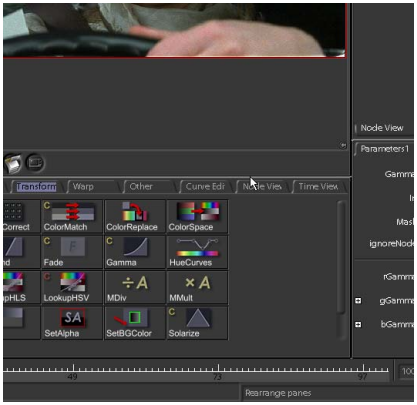
To expand any one area to take up the full screen:

- Position the pointer in the area you want to expand, and press the Space bar.
- Press the Space bar again to shrink the area back to its original size.

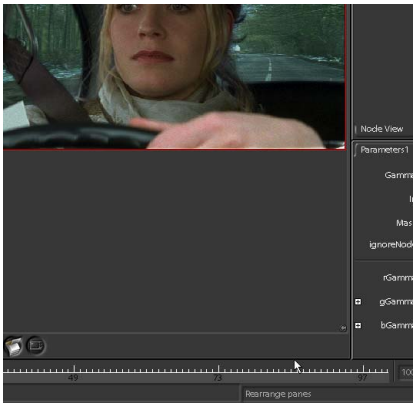
Note: Use of the Space bar is especially helpful in the Curve Editor, when you are working with high-resolution elements or large scripts.

To temporarily hide an area, do one of the following:

- Drag the top border of the Tool tab or Parameters tab areas down to the bottom.
- Drag the bottom border of the Viewer, Node View, or any other area up to the top.



Before collapsing Tool tabs



After collapsing Tool tabs

That area remains hidden until you drag its top or bottom border back out again.

Navigating in the Viewer, Node View, and Curve Editor

The Viewer, Node View, and Curve Editor are all capable of containing much more information than can be displayed at one time. You can pan and zoom around within each of these areas in order to focus on the elements you want to adjust in greater detail.

Important: Shake requires the use of a three-button mouse—the middle mouse button is key to navigating the Shake interface. If, in Mac OS X, you map Exposé functionality to the middle mouse button, this will interfere with navigation in Shake, and you should disable this functionality.

To pan across the contents of an area, do one of the following:

- Press the middle mouse button and drag.
- Option-click (Mac OS X) or Alt-click (Linux) and drag.

To zoom into or out of an area, do one of the following:

- Hold down the Control key and drag while holding down the middle mouse button.
- Control-Option-drag or Control-Alt-drag.
- Use the + or - key to zoom in or out based on the position of the pointer.

To reset an area to 1:1 viewing, do one of the following:

- In the Viewer, click the Home button in the Viewer shelf.
- Move the pointer to an interface area, then press Home.

To fit the contents to the available space within an area:

- In the Viewer, click the Fit Image to Viewer button in the Viewer shelf.
- Move the pointer to an interface area, and press F.

Saving Favorite Views

If you find yourself panning back and forth within a particular area to the same regions, it might be time to create a Favorite View within that area.

- In the Node View, you could save several views in your node tree where you'll be making frequent adjustments.
- If you're doing paint work on a zoomed-in image in the Viewer, you can save the position and zoom level of several different regions of the image.
- In the Curve Editor, you can save several different pan, zoom-level, and displayed-curve collections that you need to switch among as you adjust the animation of different nodes in your project.
- In the Parameters tab, you can save the parameters being tweaked, as well as the node being displayed in the Viewer.

Once you've saved one or more Favorite Views in each interface area, you can instantly recall the position, zoom level, and state of that area by recalling the Favorite View that you saved. You can save up to five Favorite Views.

To define a Favorite View:

1. Pan to a position in an area that contains the region you want to save as a Favorite View. If necessary, adjust the zoom level to encompass the area that you want to include.
2. Depending on the area you're adjusting, you can save additional state information particular to that area. Make additional adjustments as necessary so that you can recall the desired project elements:
   - In the Node View, you can save the state of the nodes that are currently loaded into the Viewer and Parameters tabs.
   - In the Viewer, you can save the node that's currently being viewed.
   - In the Curve Editor, you can save the curves that are currently loaded and displayed.
   - In the Parameters tab, you can save the parameters that are being tweaked, as well as the node displayed in the Viewer.
3. To save a Favorite View, move the pointer into that area and do one of the following:
   - Right-click anywhere within the area, then choose Favorite Views > View N > Save from the shortcut menu (where N is one of the five Favorite Views you can save).
   - Press Shift-F1-5, where F1, F2, F3, F4, and F5 correspond to each of the Favorite Views.

Restoring Favorite Views

Once you've defined one or more Favorite Views, you can restore them in one of two ways. Simply restoring the *framing* results in the current contents of that area being panned and zoomed to the saved position. Restoring the *framing and state*, on the other hand, results in the restoration of additional state information that was adjusted in step 2.

To restore the framing of a Favorite View, do one of the following:

- Right-click in the Viewer, Node View, or Curve Editor, then choose Favorite Views > View N > Restore Framing from the shortcut menu (where N is one of the five Favorite Views you can save).
- Press F1-5, where F1, F2, F3, F4, and F5 correspond to each of the Favorite Views.

That area is set to the originally saved position and zoom level.

To restore the framing and state of a Favorite View, do one of the following:

- Right-click in the Viewer, Node View, or Curve Editor, then choose Favorite Views > View N > Restore Framing & State from the shortcut menu (where N is one of the five Favorite Views you can save).
- Press Option-F1-5 or Alt-F1-5, where F1, F2, F3, F4, and F5 correspond to each of the Favorite Views.

Depending on the area, the originally saved position and zoom level are recalled, as well as the following state information:

- In the Node View, the node or nodes that were loaded into the Viewer and Parameters tabs when you saved the Favorite View
- In the Viewer, the node that was viewed when you saved the Favorite View
- In the Curve Editor, the curves that were loaded and displayed when you saved the Favorite View
- In the Parameters tab, the parameters that were being tweaked, as well as the node that was displayed, when you saved the Favorite View

Working With Tabs and the Tweaker

Each area of the Shake window has several tabs that reveal more of the interface. These tabs can also be customized. For example:

To move a tab to another area:

- Select a tab using the middle mouse button or Option-click / Alt-click, and then drag the tab into a new window pane.

To detach a tab and use it as a floating window:

- Shift-middle-click or Shift-Option-click / Shift-Alt-click the tab.

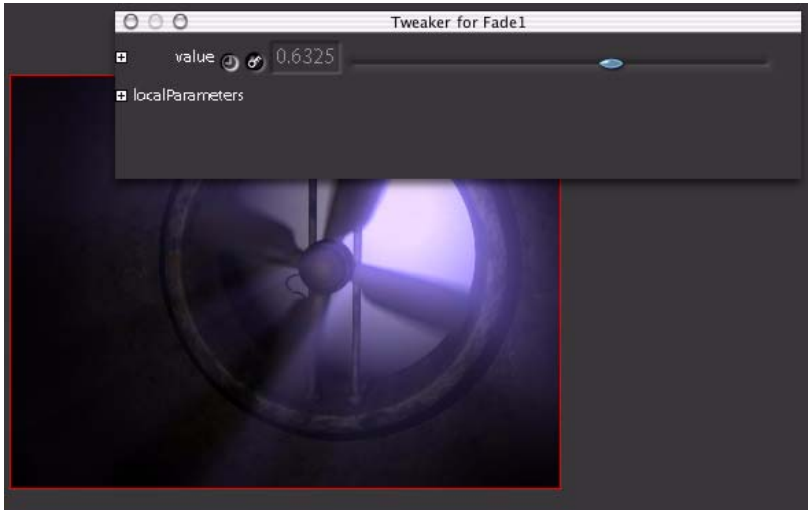
A good example of this last operation is to detach a Parameters tab, then press the Space bar while the pointer is positioned over the Viewer. You can then tune your image in full-screen mode.

Using the Tweaker

The parameters of individual nodes can be opened into a floating window, called the Tweaker.

To open a floating Tweaker window:

- Select the node you want to tune and press Control-T. A movable, floating Tweaker window for the node appears.



Note: To save your window settings for later use, choose File > Save Interface Settings.

OS Window Functions

Shake responds to OS windowing, so you can resize the entire window, expand it to full screen, or stow it as an icon by clicking the standard buttons in the upper-right corner of the Shake Viewer title bar.

Menus and the Title Bar

This section discusses the Shake title bar and the Shake, File, Edit, Tools, Viewers, Render, and Help menus.

Title Bar Information

The title bar of the full Shake window displays the current version of Shake, the name of the currently open script, and the current proxy resolution in use.

Shake Menu (Mac OS X Only)

The following table shows the Shake menu options. The Shake menu appears only in the Macintosh version of Shake.

Menu Option	Description
About Shake	Displays the Shake version number and copyright information.
Services	Services provide a quick way to perform tasks with several applications.
Hide Shake (Command-H)	Hides Shake. To show Shake again, click the Shake icon in the Dock.
Hide Others (Option-Option-Command-H)	Hides all running applications other than Shake. To show the applications again, choose Shake > Show All.
Quit Shake	Quits the Shake application.

File Menu

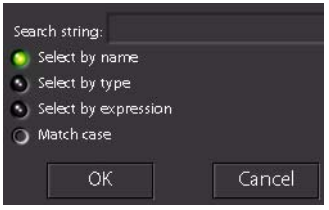
The following table shows the File menu options.

Menu Option	Description
New Script (Command-N or Control-N)	Deletes all nodes currently in the Node View. (You can also press Command-A or Control-A in the Node View to select all nodes and then press Del.)
Open Script (Command-O or Control-O)	Opens the Load Script window. The script selected in the Browser replaces what is already in the Node View. You can also use the Load button in the title bar.
Import Photoshop File	Imports a Photoshop file. If the Photoshop file contains multiple layers, you can import the layers as separate *FileIn* nodes that are fed into a *MultiLayer* node, or as a single, composited image by using a normal *FileIn* node.
Reload Script	Reloads the script listed in the title bar.
Add Script	Opens the Load Script window. Adds a second set of nodes to those currently in the Node View. The added nodes are renamed if a naming conflict arises. (For example, *FileIn1* becomes *FileIn2* if *FileIn1* already exists.) Global settings are taken from the added script, as is the new script name.
Save Script (Command-S or Control-S)	Saves the script without prompting you for a script name (if you have already saved). You can also use the Save button in the title bar.
Save Script As (Shift-Command-S or Shift-Control-S)	Opens the Save Script window. Enter the new script name, and click OK to save the script.
Save Selection As Script	Saves the currently selected nodes in the Node View as a separate script.

Menu Option	Description
Recover Script (Shift-Command-O or Shift-Control-O)	Loads the last autoSave script and is usually done when the user has forgotten to save a script and quits Shake, or when Shake has unexpectedly quit. The script is found under *$HOME/nreal/autoSave*. (The *$HOME* directory is your personal Home directory, for example, the */Users/john* directory.) If you have environment variables set, you can launch Shake on the command line with the same option using *-recover*: *shake -recover* For more information on environment variables, see Chapter 14, "Customizing Shake."
Load Interface Settings	Opens the Load Preferences From window. Select an interface settings file from disk, and click OK to load the file.
Save Interface Settings	Opens the Save Preferences To window. This lets you save the various default Shake settings, including your window layout to a file in your *$HOME/nreal/settings* file. If you call it *defaultui.h*, it is automatically read next time you launch Shake. You can save the settings file anywhere, but it is not read automatically unless the file is in the settings directory.
Flush Cache	When you choose Flush Cache, all appropriate images are copied from the memory cache to the disk cache (depending on how the cacheMode parameter is set), but the memory cache *is not* cleared. This command is similar to what Shake does when you quit (the delay that occurs when you quit is Shake flushing the memory cache to disk).
Purge Memory Cache	Similar to the Flush Cache command, but the memory cache is cleared afterwards. This is useful if most of your RAM is filled with cache data, and you want to free it up to create and play a Flipbook without needing to exit Shake first in order to clear the memory cache. The cacheMode parameter in the Globals tab controls whether or not images in the cache are used (regardless of whether they are coming from the disk or memory).
Recent Scripts	Lists the last five scripts you worked on. Choosing a script from this list opens it within Shake.
Exit (Linux only)	Exits the program. You can also use the standard OS exit buttons in the upper corner of the interface.

Edit Menu

The following table shows the Edit menu options.

Menu Option	Description
Undo (Command-Z or Control-Z) Redo (Command-Y or Control-Y)	Undoes previous commands; up to 100 levels of undo. Layout, viewing, and parameter changes are saved in the Undo list. You can also click the Undo/Redo button.  You can change the amount of undo/redo levels in your *ui.h* file. See "Menus and the Title Bar" on page 31 for more information. If you do an Undo and you have not changed anything, click Redo to go back to your previous settings.
Find Nodes (Command-F or Control-F)	Opens the Select Nodes by Name window that allows you to dynamically select nodes that match your criteria in the search string.  • Select by name. Nodes that match the search string are immediately activated. For example, if you enter *f*, *FileIn1* and *Fade1* are selected. If you enter *fi*, just *FileIn1* is selected. • Select by type. Select nodes by type. For example, enter *Move*, and all *Move2D* and *Move3D* nodes are selected. • Select by expression. Allows you to enter an expression. For example, to find all nodes with an angle parameter greater than 180, enter: *angle>180* • Match case. Sets case sensitivity.

Tools Menu

The Tools menu provides a menu listing for each of the nodes in the Tool tabs (for example, Image, Color, Filter, and so on). You can also right-click a tab to display the tools list. More information about each of these nodes is available in Part II of this manual.

Viewers Menu

The following table shows the Viewers menu options.

Menu Option	Description
New Viewer	Creates a new Viewer in the Viewer area, and automatically stretches it to fill the Viewer area. While in a Viewer, you can also right-click and select New Viewer, or press N.
Spawn Viewer Desktop	Launches a floating Viewer window that can be moved independently of the interface. The Viewer Desktop is ideal for dual-monitor setups.

Render Menu

The following table shows the Render menu options. For more information on rendering, see Chapter 12, "Rendering With the FileOut Node," on page 333.

Menu Option	Description
Render Flipbook	Renders a Flipbook of the current Viewer. Opens the Flipbook Render Parameters window, which allows you to override the Global parameters (if necessary). To cancel the render, press Esc (Escape) when in the Flipbook window.
Render Disk Flipbook	Mac OS X only. Launches a disk-based Flipbook into QuickTime. This has several advantages over normal Flipbooks. It allows for extremely long clips, allows you to attach audio (loaded with the Audio Panel in the main interface), and lets you write out the sequence as a QuickTime file after viewing, bypassing the need to render the sequence again. For more information, see "Creating a Disk-Based Flipbook" on page 326.
Render FileOut Nodes	Renders *FileOut* nodes in the Node View. In the Node View, press F to frame all active nodes. You have the option to render only the active *FileOut* nodes, or all *FileOut* nodes.
Render Cache Nodes	Immediately caches sections of the node tree where *Cache* nodes have been inserted. This command lets you cache all Cache nodes in the Node View over a specific duration. For more information on using *Cache* nodes, see Chapter 13, "Image Caching," on page 343.
Render Proxies	Renders your proxy files for your *FileIn* nodes, and leaves your *FileOuts* untouched. For more information on proxies, see "Using Proxies" on page 137.

Script Management

The following section discusses the buttons in the upper-right corner of the Shake interface, which let you Load and Save scripts, undo and redo changes you've made, and control when and how the Viewer updates the images generated by your script.

To load or save a Shake script:

- Click Load or Save to open the Load Script window, or to save the current script with the same name.

You can also press Command-S or Control-S to save the script quickly. If the script is not yet named, the Save Script window opens.

To save a script with a new name:

- Choose File > Save Script As, and enter a new file name in the Save Script window.

To reload the same script:

- Choose File > Reload.

The script that appears in the Shake title bar is reloaded.

Customizing AutoSave

A backup script is stored automatically every 60 seconds in your *$HOME/nreal/autoSave* directory. The last saved script can be accessed with the File > Recover menu command (*shake -recover* in the Terminal), or browsed to under the Directories pull-down menu in the File Browser.

The backup time interval can be changed in your *ui.h* files in *include/startup/ui/myPreferenceFile.h*. Enter the following line (with the desired time interval, in seconds, in place of the "60"):

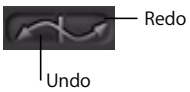
```
script.autoSaveDelay = 60;
```

Four other autosave behaviors can be customized within a .h preference file in the *include/startup* directory:

- *script.autoSaveDirectory:* Setting a directory with this declaration overrides the default behavior of placing autosave scripts in ~/nreal/autosave/.
- *script.autoSavePrefix:* Defines text to be prepended to autosave script names.
- *script.autoSaveNumSaves:* Sets the total number of autosave scripts to be saved.

To undo or redo, do one of the following:

- Click the Undo or Redo button.



- Press Command-Z or Control-Z to undo, or press Command-Y or Control-Y to redo.

By default, there are 100 steps of undo and redo in Shake.

Changing the Possible Levels of Undo

To change the level of undos, enter the following line (with your desired number of undos in place of the "100") in one of your *ui.h* files:

```
gui.numUndoLevels = 100;
```

For more information, see Chapter 30, "Installing and Creating Macros," on page 905.

Update

The Update button controls what is updated in the Viewer, and when. The Update button has three modes:

- *Always:* Updates the Viewer with every change that's made to a parameter, and every time you move the playhead in the Time Bar.
- *Manual:* The scene is not updated until you do one of the following:
  - Click Update.
  - Click the left side of a node in the Node View.
  - Press the U key.
- *Release:* Waits until you finish adjusting a parameter, or moving the playhead in the Time Bar, before updating the image in the Viewer.

By default, clicking Manual once toggles this setting to Always. Click and hold this control to see the pop-up menu, from which you can choose Always, Manual, or Release.

Proxy

The proxy button, labeled "Base," allows you to quickly get to one of your four proxy settings.

Click Base once to toggle to P1. Click and hold the Base button for other proxy options.

For more information on proxies, see Chapter 4, "Using Proxies," on page 137.

The File Browser

The File Browser is an interactive browser that serves many purposes. It lets you navigate the local volumes (both fixed and removable media) on your computer, or remote volumes over your network. You use it to open or save scripts, load images via a *FileIn* node, and to load and save lookup files and expressions.

Using the File Browser, image sequences can be listed either as a long list of individual files or as a single object. You can bookmark favorite directories. You can also use it to create and delete directories, and delete files directly in the Browser.

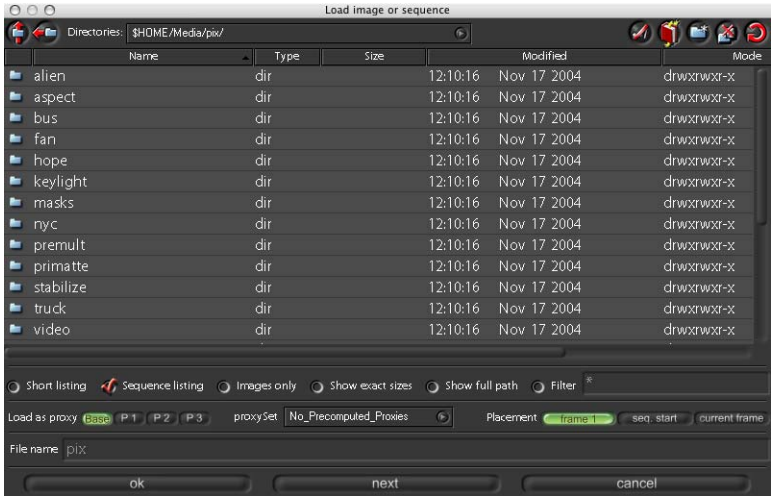
Opening the File Browser

There are several operations that open the File Browser.

To open the File Browser, do one of the following:

- Create a *FileIn* or a *FileOut* node.
- Click the Load Script button, located in the upper-right section of the Shake interface.
- Click the Save Script button, located in the upper-right section of the Shake interface.
- To open the File Browser from an existing *FileIn* or *FileOut* node (for example, if the source media becomes disconnected), click the folder icon next to the file path in the Parameters tab.

The Browser opens. If you're using Mac OS X, this window appears very different from the standard file navigation sheet, but it has much of the same functionality, and includes additional options that are particularly useful to Shake projects.



Navigating in the File Browser

There are several ways you can navigate to the directory you need using the File Browser:

Using the File List

A list of directories and files appears in the center of the File Browser. You can double-click any directory in this list to make it into the current directory. The following list of controls lets you navigate the volumes accessible to your computer.

Icon, Button, or Key	Description
	Indicates a folder.
	Indicates a drive.
	Takes you to the last viewed directory.
	Takes you up one directory. You can also press Delete (Macintosh) or Backspace (Linux).

Icon, Button, or Key	Description
Up Arrow/Down Arrow key	Moves up and down in the list.
Any letter key	Once you have clicked in the file listings, press a letter key on the keyboard to jump to the next occurrence of a file or directory that starts with that letter.

Using the Pull-Down Menu at the Top

The pull-down menu reveals the entire directory tree, including your root directory, the directory you launched Shake from, the $HOME directory, the Shake installation, and any favorite directories you have entered. This menu also automatically lists recently visited directories.

Using a File Path

You can also type an entire path, with or without the file name itself, into the File Name field at the bottom of the Browser.

You can format absolute file paths in any of several styles:

- /my_proj/my_directory/my_file.iff
- /d4/my_proj/my_directory/my_file.iff
- //MyMachine/D/my_proj/my_directory/my_file.iff

Enable/Disable Local File Paths

The Relative Path control, to the left of the Add to Favorites control in the File Browser, gives you the option to enter a relative file path into the File Name field.

Relative file paths can take one of two forms:

- ./myDirectory/myFile/
- ../myDirectory/myFile/

Adding Directories to the Favorites List

If there are one or more directories with content you frequently need to access, you can add them to the Favorites list. The Favorites list is a customizable list of directories that you can add to at any time. You can explicitly add directories to the list in two ways:

Note: As of Shake 4, entries you add to the Favorites list are permanent.

To add an entry to the Favorites list:

1 Open the File Browser.

2 Click the Bookmark button.

The currently open directory is added to the Favorites list. All favorite directory paths you add are saved in the *favoritePaths.h* file, located in the *username/nreal/settings/* directory. By default, the *favoritePaths.h* file contains:

- Your home directory
- The nreal directory
- The Shake application directory

When you add more directories to the Favorites, they're automatically appended to the code in the *favoritePaths.h* file. For example, if you add following directory to the Favorites:

```
/Users/MyAccount/Media/
```

The resulting *favoritePaths.h* file looks like this:

```
// User Interface settings
SetKey(
    "globals.fileBrowser.favorites", "/;$HOME;/Users/MyAccount//nreal/;/
     Applications/shake-v4.00.0201/;/Applications/shake-v4.00.0201/doc/pix;/
     Users/MyAccount/Media/;"
);
```

Note that each directory path is separated by a semicolon. *MyAccount* is the name of the user directory.

To remove directories from the Favorites list:

1 Open the *favoritePaths.h* file (located in the */nreal/settings/* directory).

2 Delete the paths you want to remove from the Favorites list, and save the file.

You can also instruct Shake to look in certain directories when you start the software, using the following *ui.h* settings. Each listing is for a type of file—images, scripts, expressions, and so on. Note the slash at the end of the path:

```
gui.fileBrowser.lastImageDir= "/Documents/my_directory/";
gui.fileBrowser.lastScriptDir= "$MYPROJ/shakeScripts/";
gui.fileBrowser.lastExprDir= "//Server/shakeExpressions/";
gui.fileBrowser.lastTrackerDir= "$MYPROJ/tracks/";
gui.fileBrowser.lastAnyDir= "/";
```

For more information on a *ui.h* file, see Chapter 14, "Customizing Shake," on page 355.

Selecting Files

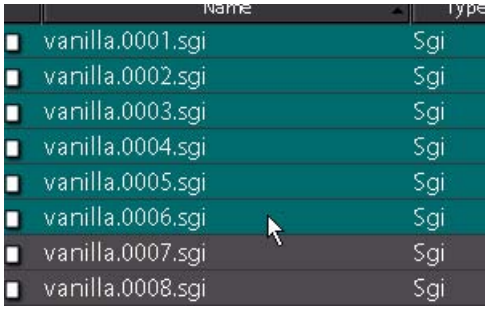
If you're selecting one or more files for a *FileIn* operation, you can select them in several ways.

To select single files, do one of the following:

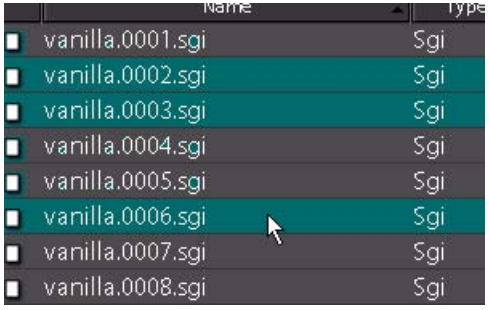
- Double-click the file.
- Press the Up Arrow or Down Arrow, then click OK (or press Return).
- Press the first letter of the file you want. Press it again to jump to the next file that starts with that letter. Click OK (or press Return).

To select multiple files in the same directory, do one of the following:

- To select multiple files, drag to select the files, then click OK.



- To select multiple individual files, press Shift and select the files.



To select multiple images in different directories, do one of the following:

- Click Next in the Browser to load the current image or images and keep the Browser open to continue to add files. When you have reached the last file, click OK. At any time in this process, the Node View may be accessed to examine *FileIn* nodes.
- Select one or more files, and press the Space bar to add them to an invisible queue of files to be added to your script, without closing the File Browser. Once you click OK, every file in this invisible queue is added to the currently open script.

Viewing Controls

There are several tools to help you identify files in the Browser.

The File List

Click the title of a column to arrange the list according to that type of information. For example, click Modified to list files by creation date. Click Modified again to reverse the order of information.

Toggle Buttons

The following buttons also change what is listed in the Browser:

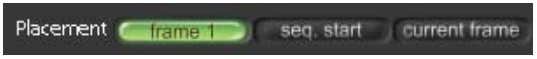
Button	Description
Short Listing	Lists only file names, type, and size.
Sequence Listing	Toggles the listing of an image sequence as one listing or as several. To read in the entire sequence, ensure that this Sequence Listing is enabled. These icons signify single or sequence files:  Indicates a single file.  Indicates an image sequence.
Images Only	Lists only recognized image types.
Show Exact Sizes	Shows the exact file size in kilobytes, rather than rounded off in megabytes.
Show Full Path	Lists the entire path of the selected file.
Filter	Filters out information. Use * and ? as your wildcards: **Wildcard** * **Example** *.cin *.cin.* *.cin* image.*.tga **Means** Any set of characters for any length. **Lists** a.cin, image.cin, image.0001.cin, ... a.cin.0001, image.cin.hr a.cin, image.0001.cin, image.cin.0001 image.1.tga, image.10.tga, image.0100.tga **Wildcard** ? **Example** ?.cin ??.iff a.??? image.????.iff image.???1.iff **Means** Any character in that single position. **Lists** a.cin, 1.cin ab.iff, 01.iff a.cin, a.iff, a.tga image.0001.iff, image.9999.ifff image.0001.iff, image.0011.iff, image.1111.iff

Updating the File Browser

Click the Update button to refresh the listing of the current directory in case files have been added or deleted while the File Browser has been open.

Specifying Media Placement

Three buttons let you set the first frame at which new media is placed when it's read into your Shake script. This affects the timing of the media inside of your script, and can be seen in the Time View tab.



- *frame 1:* The first frame of media is placed at the first frame of your script.
- *seq. start:* The first frame of media is placed to the corresponding frame of the script, depending on its frame number. If you import frames 9-50 of an image sequence, the first frame of media appears at frame 9 of the Time View.
- *current frame:* The first frame of media is placed at the current position of the playhead.

Additional Controls for Image Output

When you're writing out a file using the *FileOut* node, you also use the File Browser to select a directory and enter the file name for the rendered output. For more information about exporting media from Shake, see Chapter 12, "Rendering With the FileOut Node," on page 333.

File Management Controls

There are additional file management controls that are primarily useful when exporting ("writing out") media.

Button	Description
	Creates a new directory using the current path.
	Deletes currently selected files and directories.

Naming Files for Output

If you are writing out an image sequence, be sure to insert a # or an @ sign where you want the frame number to go in the name. When you are finished, click OK to validate.

The following is a table of examples.

Files	Shake Notation
image.0001.cin, image.0002.cin	*image.#.cin*
image.1.tif, image.2.tif	*image.@.tif*
image.iff.0001, image.iff.0002	*image.iff.#*
image1.tga, image2.tga	*image@.tga*
image.001.tga, image.002.tga	*image.@@@.tga*
image.01, image.02	*image.@@*

Using and Customizing Viewers

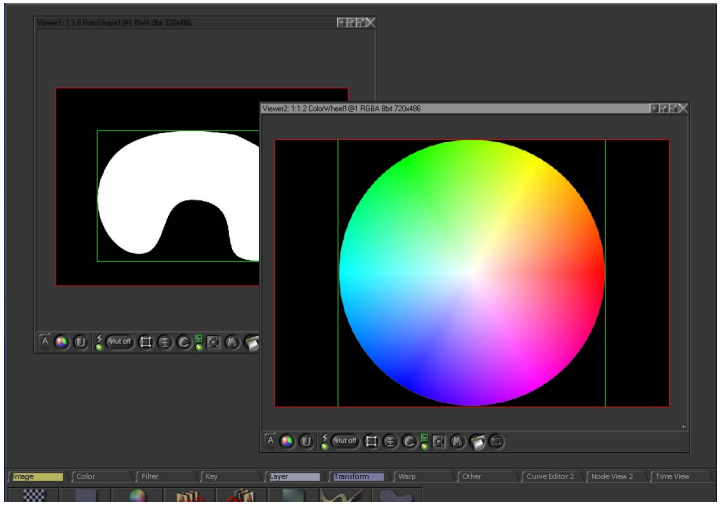
Shake displays the currently selected image for your project in the Viewer, located in the Viewer workspace at the upper-left quadrant of the interface. Additional controls in the Viewer shelf at the bottom of the Viewer workspace let you customize how the image is displayed. For example, you can view each channel of an image individually. You can also change the Gamma of the Viewer to simulate how the image might look on other output devices. Other tools such as the Histogram and Plot Scanline Viewer scripts can help you analyze your image.

Image from The Saint provided courtesy of Framestore CFC and Paramount British Pictures Ltd.

Some nodes feature onscreen controls (which also appear in the Viewer) that let you make adjustments to an image. For example, the *Move 2D* and *Rotate* nodes display transformation controls you can drag to manipulate the image directly in the Viewer. Other nodes make additional tools available in the Viewer shelf. For example, the *RotoShape* node places drawing and editing tools in the Viewer shelf that let you create and manipulate shapes directly in the Viewer.

Using Multiple Viewers

You can create as many Viewers within the Viewer workspace as you need. Each additional Viewer you create appears within the Viewer workspace area, and each Viewer can be set to independently display any image channel from any node in the current node tree. Each Viewer has its own Viewer shelf with its own controls, and each Viewer you create has two buffers that you can use to compare images.



Note: Because each Viewer has two buffers, using multiple Viewers at once can consume a lot of RAM, depending on the resolution of the images you're working with.

When additional Viewers are displayed, each Viewer is updated dynamically. You can use this capability to simultaneously view the results of a downstream node in one Viewer, while also seeing the image from an upstream node in another Viewer. A good example of this is when you're refining a key in one Viewer, while watching the effect this has on the finished composite in another.

To create additional Viewers:

1. Position the pointer anywhere within the Viewer area.
2. Do one of the following:
   - Press N.
   - Right-click, then choose New Viewer from the shortcut menu.
   - Choose Viewers > New Viewer to create a new Viewer.

A new Viewer named "Viewer2" is created above Viewer1 in the Viewer workspace. Additional Viewers are numbered in the order in which they're created.

Note: Each Viewer you create uses additional memory, so you may want to close higher-resolution Viewers when rendering. Also, more open Viewers can slow the display rate due to the increased processing demands of updating each Viewer simultaneously.

To close a Viewer:

- Click the Close Window button in the selected Viewer's title bar.

When you close a Viewer, you release whatever RAM that Viewer was utilizing.

To bring a Viewer to the foreground:

- Click anywhere within that Viewer.

When a Viewer is selected, its title bar is highlighted.

To display the image from another node in a Viewer:

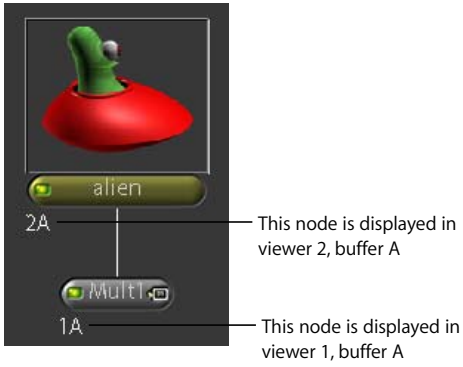
1. Double-click anywhere within the Viewer you want to use.

   Its title bar becomes highlighted to show that it is selected.

2. Do one of the following:

   - Double-click a node in the Node View to display its image in the currently selected Viewer and to display its parameters in the Parameters1 tab.
   - Click the left side of a node to display its image in the currently selected Viewer without loading its parameters into the Parameters tab.

When a node is loaded into the Viewer, an indicator appears on the left side of the node. Additionally, a number and a letter appear below it, specifying which Viewer and compare buffer that node's image occupies.

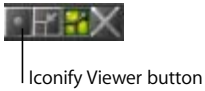


To collapse a node to a minimized state:

- Click the Iconify Viewer button in the Viewer title bar to minimize its size.

To expand a node from a minimized state:

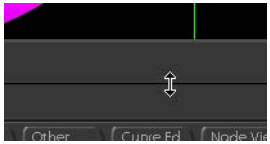
- Click the Iconify Viewer button of the minimized Viewer to restore it to its original size.



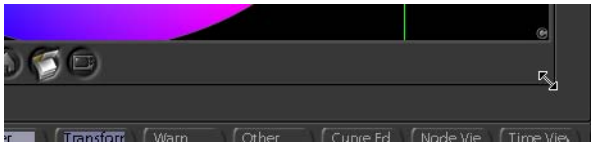
Minimized Viewers are only as large as the title and the upper-right window controls.

To resize a Viewer, do one of the following:

- Drag a Viewer's left, right, or bottom side.



- Drag a Viewer's bottom-right corner to resize its width and height simultaneously.



To resize a Viewer to fit the image within:

- Click the Fit Viewer to Image button in the Viewer title bar.

To lock a Viewer to the full size of the Viewer workspace:

- Click the Grip to Desktop button in the Viewer title bar.

The full-size Viewer now obscures any other Viewers underneath it, and resizes itself to match the total size of the Viewer workspace. To see other Viewers underneath, click the Grip to Desktop button again to release the Viewer, then resize it or move it to make room in the Viewer workspace for the other Viewers.

Note: By default, the first Viewer that was created along with your project is locked to the full size of the Viewer workspace.

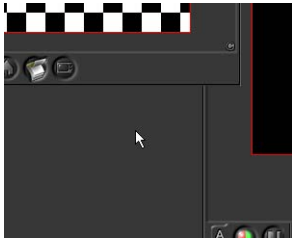
To exand the Viewer workspace to the full size of the Shake window:

- Press the Space bar.

  Afterwards, you can press the Space bar again to reset the Viewer workspace to its original size.

To reposition all Viewers within the Viewer workspace at once:

- Click the middle mouse button anywhere in the Viewer workspace outside of any of the Viewers, then drag to move all of the Viewers around the workspace.



To create a separate Viewer workspace for use on a second monitor:

- Choose Viewers > Spawn Viewer Desktop. The second monitor must be run from the same graphics card as the primary monitor.

To help prevent accidentally rendering an enormous image (for example, if you enter 200 into your zoom parameter instead of 20), the Viewer's resolution is limited to 4K. This limit is configurable. The Viewer resolution has no effect on rendered images—it only crops the view in the Shake interface to the set resolution.

> ***Warning:*** If you get strange Viewer behavior, delete the Viewer (right-click, then choose Delete Viewer from the shortcut menu), and create a new Viewer.

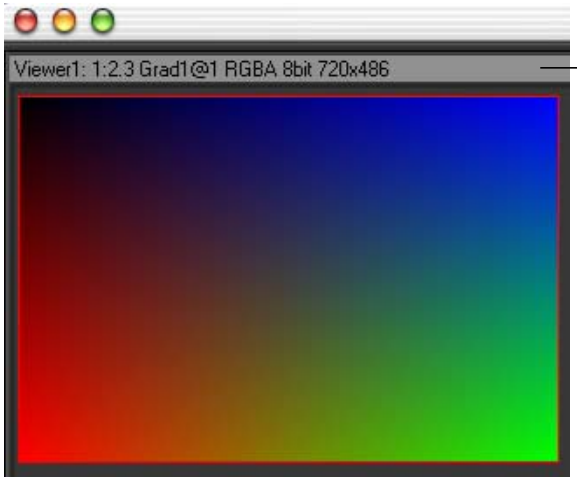
Looking at Images in a Viewer

To load a node into the current Viewer, click the left side of the node. The green Viewer indicator appears. Double-click the node to simultaneously load the node's parameters in the Parameters tab.

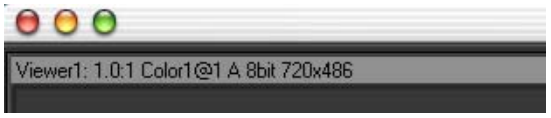
In the following image, the *Grad* node is loaded into the Viewer.

Viewer indicator: Click once to display node in Viewer. Double-click to load node parameters.

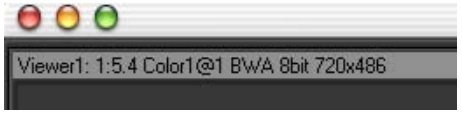
The information (node name, channels, bit depth, size, and so on) for the selected node appears in the Viewer title bar. When the *Grad* node is loaded into the Viewer, the following appears in the Viewer title bar.



Note: The channels displayed in the Viewer are the non-zero channels. Non-zero channels are not the same as active channels. For example, a *Color* node set to black (black and white values are zero) displays the alpha channel (A) in the Viewer title bar:



When the Color node is adjusted toward gray (the black and white values are no longer zero), the black, white, and alpha channels (BWA) are displayed:



Every Viewer has the built-in capability to analyze color.

To quickly analyze colors in the Viewer:

- Click and scrub with the mouse in the Viewer to display the X, Y, R, G, B, and alpha values in the Viewer title bar.

These values are also displayed in the Info field in the bottom-right corner of the interface. You can also use the Pixel Analyzer and Color Picker windows to analyze this data with more extensive options.

Note: To display the values in the Terminal window that launched Shake, press Control and scrub in the Viewer.

Suspending Rendering and Viewer Redraw

There are two ways you can suspend rendering the node tree. This can help if you're making adjustments to a render intensive node and you don't want to wait for the image to render and display in the Viewer every time you make an adjustment.

To immediately stop rendering at any time:

- To stop any processing at any time, press Esc.

Rendering is suspended until another operation occurs that requires rendering.

To suspend rendering and redraw in the Viewer altogether:

- Change the Viewer's update mode to Update Off via the Update Mode control in the Viewer shelf.

Rendering is suspended until the Viewer's update mode is changed to Update On or Progress.

Controls in the Viewer Shelf

The Viewer shelf has many controls that let you customize how images are displayed in the Viewer. These controls can be used directly, but many Viewer controls also correspond to shortcut menu items and keyboard shortcuts with the same function.

To use the two different click-and-hold button behaviors:

- Click the View Channel button in the Viewer to toggle between the RGB and the alpha views.

- Click and hold the View Channel button to open a pop-up menu from which you can choose a specific channel view.

Click.

Click and hold.

You can override the default channel display progression when the View Channel button is clicked. For example, clicking the button in RGB view displays the alpha view. When you click again, the RGB view is displayed.

To go from RGB to red to alpha channels:

1. Command-click or Control-click and hold the View Channel button, then select the Red Channel button.
2. Command-click or Control-click and hold the View Channel button, then select the Alpha Channel button.

   When you click the View Channel button, you now toggle through RGB, red, alpha, and back to RGB.
3. To save this behavior, choose File > Save Interface Settings.

Note: You can cycle through some Viewer functions using number keys. Press 2 to cycle forward through the channel views, and press Shift-2 to toggle backward through the channel views. Press 1 to toggle between the A and B compare buffers.

The following table shows the Viewer buttons, the keyboard or hot key shortcuts, and describes the button functions.

Button		Shortcut	Description
	Pointer		Drag the pointer in the Viewer to display the X and Y position values, and the RGBA color values in the Viewer title bar. The values are also displayed in the Info field.
	Iconify Viewer		Stows the current Viewer.
	Fit Viewer to Image	Control-F	Fits the Viewer to the image.
	Grip to Desktop	Shift-F	Fits the frame to the Viewer workspace. When enabled, the Viewer "sticks" to the workspace. You can then resize the workspace and the Viewer expands to match.
	Close Window	Right-click menu > Delete	Deletes the Viewer. (A good strategy if a Viewer is misbehaving. Press N to create a new Viewer.)
	**Buffer Tabs**	**1**	You can have two different buffers in a Viewer to compare images. See "Using the Compare Buffers" on page 57.
	View Channel	R, G, B, A, C; 2/Shift-2 cycle; right-click menu	Toggles through the color channels: RGB (color), red, green, blue, alpha.
	Update Mode–On	Right-click menu; 3/Shift-3	Update mode that displays a rendered image only after it is finished rendering. This is for relatively fast renders.
	Update Mode–Progress	Right-click menu; 3/Shift-3	Scrolling update mode that displays each line (starting from the bottom) as the image renders. Used for slower renders.
	Update Mode–Off	Right-click menu; 3/Shift-3	The Viewer does not update. Use this to load an image into a Viewer, then switch to the second buffer (see below) and do some changes. You can then compare it with the original.

Button		Shortcut	Description
	Incremental Update		Updates the changing portion of the image. For example, if Toggle Incremental Update is disabled and you composite a 10 x 10-pixel element on a 6K plate and pan the element, the entire 6K plate updates. When enabled, only the 10 x 10-pixel area is updated. To fix this, turn off the Incremental Update and adjust again—the glitches are corrected. This button has no effect on the output file or batch rendering speed, only on the image in the Viewer.
	VLUT Off	Right-click menu	VLUTs (Viewer lookup tables) differ from Viewer scripts in that you can scrub from the unmodified plate. See "Viewer Lookups, Viewer Scripts, and the Viewer DOD" on page 61.
	Truelight VLUT	Right-click menu	The Truelight VLUT combines monitor calibration with the previsualization of film recorders and other output devices. See "Viewer Lookups, Viewer Scripts, and the Viewer DOD" on page 61.
	*Gamma*/Offset/ LogLin VLUT	Right-click menu	The Gamma/Offset/LogLin VLUT allows you to apply different quick lookups to your image. See "Viewer Lookups, Viewer Scripts, and the Viewer DOD" on page 61.
	Viewer DOD	Right-click menu	Turns on Region of Interest (ROI) rendering (limits your rendering area). See "Viewer Lookups, Viewer Scripts, and the Viewer DOD" on page 61.
	Viewer Script–Off	Right-click menu; 4/Shift-4	See "Viewer Lookups, Viewer Scripts, and the Viewer DOD" on page 61.
	Viewer Script–Aperture Markings	Right-click menu	Applies aperture markings. You can also right-click the Viewer Script button, then choose Aperture Markings. See "Viewer Lookups, Viewer Scripts, and the Viewer DOD" on page 61.
	Viewer Script–Plot Scanline	Right-click menu	Applies a plot scanline. You can also right-click the Viewer Script button, then choose Plot Scanline. See "Viewer Lookups, Viewer Scripts, and the Viewer DOD" on page 61.
	Viewer Script–Histogram	Right-click menu	Applies a Histogram. You can also right-click the Viewer Script button, then choose Histogram. See "Viewer Lookups, Viewer Scripts, and the Viewer DOD" on page 61.

Button		Shortcut	Description
	Viewer Script–Z Channel	Right-click menu	Views the Z channel. You can also right-click the Viewer Script button, then choose ViewZ. See "Viewer Lookups, Viewer Scripts, and the Viewer DOD" on page 61.
	Viewer Script–Superwhite/Subzero	Right-click menu	Displays superwhite and subzero pixels. You can also right-click the Viewer Script button, then choose Float View. See "Viewer Lookups, Viewer Scripts, and the Viewer DOD" on page 61.
	Viewer Script–Frames/Timecode	Right-click menu	Displays frames or timecode in the active Viewer. See "Viewer Lookups, Viewer Scripts, and the Viewer DOD" on page 61.
	Compare Mode–No Compare	5/Shift-5	Only one buffer is displayed. See "Using the Compare Buffers" on page 57.
	Compare Mode–Horizontal (Y Wipe)	5/Shift-5	You can also right-click the Compare Mode button, then choose Y Wipe. See "Using the Compare Buffers" on page 57.
	Compare Mode–Vertical (X Wipe)	5/Shift-5	You can also right-click the Compare Mode button, then choose X Wipe. See "Using the Compare Buffers" on page 57.
	Compare Mode–Blend (Fade)	5/Shift-5	You can also right-click the Compare Mode button, then choose Blend. See "Using the Compare Buffers" on page 57.
	Show/Hide DOD Border	Right-click menu	Displays the green DOD (Domain of Definition) border and the red frame border. It has no effect on processing or the rendered image.
	Reset Viewer	Home key	Centers the image and sets the zoom level to 1:1.
	Fit Image to Viewer	F	Fits the image to the frame. Be careful, since you may get a non-integer zoom (for example, instead of 2:1, you get 2.355:1), which may result in display artifacts. Do not use this option when "massaging" pixels.

Button		Shortcut	Description
	Launch Flipbook		Renders a RAM-based image player. Left-mouse click: Renders with the current settings. Right-mouse click: Displays the Render Parameters window.
	Broadcast Monitor		Mirrors the selected node in the Viewer on a video broadcast monitor. The broadcast monitor option is only available in the Mac OS X version of Shake. For more information, see "Viewing on an External Monitor" on page 330.

For a table of additional common buttons related to onscreen controls, see "Node-Specific Viewer Shelf Controls" on page 70.

Using the Compare Buffers

You can use the A and B buffers in the Viewer to load two images at once. The following example uses two images from Tutorial 5, "Using Keylight," in the *Shake 4 Tutorials* book.

To load two images at once into the A and B compare buffers:

1. Using two *FileIn* nodes, read in two images from the "Using Keylight" tutorial.

   The images are located in the *$HOME/nreal/Tutorial_Media/Tutorial_05/images* directory.

2. In the Viewer, ensure that buffer A is open.

3. Load one of the tutorial images into the Viewer by clicking the left side of the node.

   The Viewer indicator appears on the left side of the node, and the node's image is loaded into the Viewer.

4 To switch to buffer B, click the A tab, or press 1 (above the Tab key, not on the numeric keypad).

The A tab switches to B when clicked.

5 Load the second image into buffer B by clicking the right side of the image's node.

6 Press 1 to toggle between buffers. You can also click the A and B tabs.

Images from *The Saint* provided courtesy of Framestore CFC and Paramount British Pictures Ltd.

You can also put the Viewer into split-screen mode to more directly compare two images.

To create a vertical split screen in the Viewer:

- Drag the Compare control (the small gray "C" the lower-right corner of the Viewer) to the left.

The Compare Mode button in the Viewer shelf indicates that you are in vertical compare mode.

To create a horizontal split screen:

- Drag the Compare control up on the right highlighted edge.

The Compare Mode button in the Viewer shelf indicates that you are in horizontal compare mode.

Alternatively, you can toggle between vertical and horizontal split screens by using the Compare Mode button.

Note: A common mistake is to slide the Compare control all the way to the left or right (or top or bottom)—one image disappears and only the second image is revealed. The result is that changes to a node's parameters don't update the Viewer. To avoid this, turn off the Compare Mode to ensure you are looking at the current image.

To turn off split-screen viewing:

- Click the Compare Mode button in the Viewer shelf.

The split screen is removed and the button is no longer highlighted.

If the Compare Mode button is set to No Compare and the Viewer is still not updating, make sure that the Update Mode is set to On.

If the Update Mode is not the problem, check to make sure that the manual Update button at the top of the interface is not set to "manual."

Viewer Lookups, Viewer Scripts, and the Viewer DOD

There are three similar controls that affect how your images are viewed:

- Viewer lookup tables (VLUTs)

- The Viewer DOD

- Viewer scripts

These functions modify the image for efficiency or previsualization purposes, and do not affect the output image. If necessary, it's possible to apply these settings to a render that is launched from the interface.

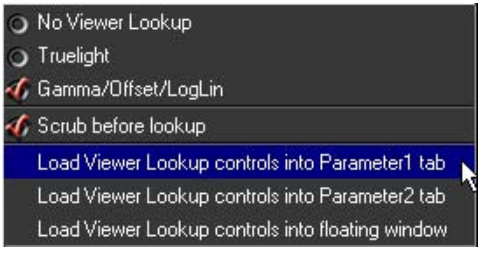
The following is an example of using a VLUT with a log image.

When LogLin conversion is enabled in VLUT 2, you still work on the log image in the process tree, but you see the linearized plate. (For more information about logarhtymic-to-linear conversion, see "The Logarithmic Cineon File" on page 437.)

To activate the VLUT or Viewer Script controls:

1 Apply your VLUT or script.

2 Right-click the VLUT (Viewer Lookup Table) button and select one of the three Load Viewer Lookup options.

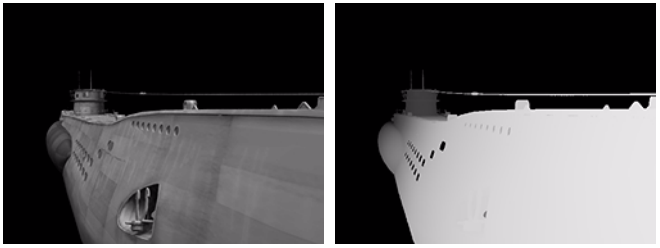


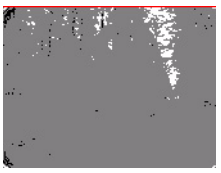
3 In the designated window or tab (selected in step 2), adjust any necessary parameters.

VLUTs have an additional right-click function to specify whether pixel values are scrubbed from before or after the VLUT. Right-click and hold the VLUT button, and enable (or disable) "Scrub before lookup."

The following table includes the current default scripts and VLUTs.

Button	Description
	*VLUT*: Three options let you turn on or off a VLUT for the Viewer. • By default, the VLUT is turned off. • A second option lets you use the Truelight VLUT control, combining monitor calibration with the previsualization of film recorders and other output devices. • A third option, VLUT 2, allows Gamma, Add, and LogLin operators to be applied to the Viewer.
	*Viewer Script–Aperture:* Displays a field chart with safe zones. To load script controls into the Parameters tab, right-click the button, then choose Load Viewer Script Controls from the shortcut menu. 
	*Viewer Script–Plot Scanline:* Displays a plot scanline of your image. For more information, see "Using the PlotScanline to Understand Color-Correction Functions" on page 674.  Displays pixel values along the horizontal axis (where the light gray line is). The sample image bluescreen is evenly lit. You can view RGB, A, or RGBA, and calculate according to color, luminance, or value.

Button	Description
	*Viewer Script–Histogram:* Displays a Histogram of your image. Viewer Script controls (right-click the Viewer Script button to select Load options): • *ignore:* Ignores pixels with a 0 or 1 value. • *maxPerChannel:* Pushes the values up on a per-channel basis. • *fade:* Fades the display of the Histogram.  The colors are squeezed down in a limited range, an indication that this is probably a logarithmic image. Notice the big healthy chunk of blue near the high end. That is good.
	*Viewer Script–Z Channel:* Displays the Z depth of an image either normalized or between a set range. A very important note: Closer pixels are white, so the image can fade to infinity (black) without a visual discontinuity. Viewer Script controls (right-click Viewer Script button to select Load options): • *floatZinA:* Puts the Z values in the alpha channel to scrub and retrieve these values. The values are either Off, the Original values, or Distance (normalized between 0 and 1). If you have an object that moves from far away toward the screen over several frames, Original returns your Z values relative to each other; Normalized indicates only the Z values within that frame. • *zNormalize:* Indicates whether the render came from Maya or 3ds max. The subparameter zInfinity sets the limit at which point pixels are considered infinite, and are therefore clipped. • *zRangeSource:* Evaluates the original values, or the near/far Input values. 

Button	Description
	*Viewer Script–Superwhite/Subzero:* Displays pixel values above 1 or below 0 for float images. The alpha channel is also tested. **Viewer Script controls (right-click the Viewer Script button to select Load options):** • *view:* This parameter controls how the pixels are displayed: *per channel:* Sets subzero pixels to 0, sets pixels between 0 and 1 to 0.5, and pixels above 1 to 1. This is applied on a per-channel basis. *per image:* Turns subzero pixels black, pixels between 0 and 1 gray, and pixels above 1 to 1. This is applied across the entire image, so if any channel is beyond 0 or 1, it is indicated. *on Image:* Mixes the subzero and superwhite pixels back onto the image. The colors are controlled with the two color controls. • *SubZero Color:* Only active when view is set to "on Image," it indicates the subzero pixels. • *SuperWhite Color:* Only active when view is set to "on Image," it indicates the superwhite pixels. **In this example, do the following:** • Read in the *saint_bg.@* from the *$HOME/nreal/ Tutorial_Media/Tutorial_05/images* directory. • Apply an Other–*Bytes* node and a Color–*LogLin* node. • Toggle *Bytes* from 8 to float. • Apply the Float View Viewer Script. Since LogLin pushes values above 1, the sky loses its punch when you go back out to Log and you process the image in only 16 bits.  **Tree**  **Input Log image**  **LogLin (linear) image**  **per channel float view**  **per image float view**  **on image float view** The per-channel view indicates that most of the superwhite values are in the blue channel. The per-image view indicates the dark areas more clearly. The on-image view codes the highlights yellow and the darks blue.

Button	Description
	*Viewer Script–Frames/Timecode:* Displays frames or timecode in the active Viewer. **To show and modify the frames/timecode display:** • Right-click the Viewer Script button and select timecode, or click and hold the Viewer Script button and select the Timecode button. By default, timecode is displayed in the Viewer. • Right-click the Viewer Script button and select Load Viewer Script into Parameters2 tab. The timecode parameters are loaded into the tab. • Click the mode pop-up menu to choose Frames, Padded Frames, Timecode, or Timecode Dropped Frame. • Use the Time Offset subtree to offset by hours, minutes, seconds, or frames. • *Color:* Click the color control to change the color of the text display. • *BgColor:* Click the color control to change the color of the timecode display background box. • *BgOpacity:* Controls the opacity of the timecode display background box. • *size:* Controls the size of the frames/timecode display. • *xPos:* Controls the X position of the frames/timecode display. You can also use the onscreen controls to reposition the display. • *yPos:* Controls the Y position of the frames/timecode display. You can also use the onscreen controls to reposition the display.

More About Using VLUTs

The VLUTs and the Viewer scripts are similar in that they apply an arbitrary set of functions that modify the image. A typical example is a color lookup table to compensate for the display properties of your computer's monitor.

The key difference is that VLUTs allow you to scrub pixel values from the unmodified image (this feature can be disabled) whereas you always scrub the modified pixel values when using Viewer scripts. For example, you may want to work on Cineon plates in logarithmic space without converting the plates to linear space. However, you want a rough idea of what the images look like in linear color space. To do this, apply a VLUT to convert the images to linear space. The results when scrubbing colors in the Viewer are still derived from the original, unmodified input logarithmic plates, which ensures accurate processing for your output images.

VLUTs are typically used during color correction, and Viewer scripts are typically used for unusual operations—for example, when creating an image for stereoscopic viewing. Both methods let you use any series of pre-made functions.

Shake includes two VLUTs, the Truelight VLUT, and VLUT 2, which can be customized any way you need. You can also create as many additional VLUTs as you need, for different situations. You can only turn on one VLUT and one Viewer script at the same time, but both can be activated simultaneously. To apply multiple color corrections, build your VLUTs and scripts to have multiple controls.

Note: The Truelight VLUT control in the Viewer shelf lets you set the Viewer's lookup table to use calibration profiles that you can create with the TLCalibrate node, or that are created using Truelight's monitor probe. Use the Load Viewer Lookup Controls into Parameters1 Tab command to make adjustments to the Truelight VLUT parameters. For more information on using Truelight, see the Truelight documentation, located in the Documentation folder on the Shake installation disc.

Important: Currently there is no version control of the Viewer script. If you extend the functionality of existing Viewer scripts that may have been saved in existing Shake scripts, you should rename the new version of the Viewer script to something other than the original name.

Using the Viewer's Domain of Definition (DOD)

The Viewer Domain of Definition (DOD) limits the area of the image that is rendered to the interior of a user-definable rectangle, in order to reduce the amount of unnecessary processing. For example, if you are doing a head replacement, you may want to activate the Viewer DOD and limit the Domain of Definition to a box surrounding just the head, eliminating the need for your computer to render the rest of the image.

When using the Viewer DOD, keep the following in mind:

- The Viewer DOD limits the rendering area on the Viewer, but does not affect the output image.

- Display DOD displays the green internal DOD box associated with each node and the red frame boundary (does not affect processing).

The following image has both the VLUT 2 and the Viewer DOD applied.

Right-click the Viewer DOD button to access the DOD control options. For example, using Frame DOD to Viewer (sets the DOD to the Viewer frame), you can zoom in on an area you want to focus on and limit your DOD to that area. Note that the DOD is not dynamic, as it would need to constantly recalculate as you pan. For more information, see "The Domain of Definition (DOD)" on page 82.

Creating Your Own VLUTs and Viewer Scripts

The preset examples are stored in the end of the *nreal.h* file. To roll your own, you first declare them in a *startup* directory following the same guidelines as for macros. The following functions do absolutely nothing:

```
image ViewerLookup1_(image img)
{
  return img;
}

image ViewerScript1_(image img)
{
  return img;
}
```

Next, also in a *startup* file, hook them into the Viewers:

```
nfxDefViewerLookup("Lookup1", "ViewerLookup1_()", "default");
nfxDefViewerScript("Script1", "ViewerScript1_()", "default");
```

The first argument ("Lookup1","Script1") is the name of the VLUT/Script as it appears in the list in the interface. The second arguments ("ViewerLookup1_()","ViewerScript1_()") are the actual functions they call when activated. These must be declared in a *startup* .h file. The third arguments are the optional icon files, relative to the *icons/viewer* directory. It is assumed there is an .nri extension and that you also have a focused version called *[icon].focus.nri*.

Therefore, if you want to load a button called *icons/viewer/vluts/dufus.nri*, you also create a focused version called *icons/viewer/vluts/dufus.focus.nri*. You then use "vluts/myVLUT" as your icon name. "default" means it is looking for *vlut.@.nri*, *vlut.@.focus.nri*, *vscript.@.nri,* and *vscript.@.focus.nri* (@ = 1, 2, 3, etc.). All paths are relative to *icons/viewer*. The icons for Viewer scripts are 30 x 30 pixels, no alpha. The standard VLUT buttons are 51 x 30 pixels, no alpha. See "Other Macros–VLUT Button" in Chapter 32, "The Cookbook." Other macros required to run the VLUT Button macro can be found in *doc/html/cook/macros*.

Viewer Keyboard Shortcuts

The following table contains additional Viewer hot keys.

Keyboard	Function
N	Create/Copy New Viewer.
F	Fit Image to Viewer.

Keyboard	Function
Control-F	Fit Viewer to Image.
Shift-F	Fit Viewer to Desktop.
Alt-drag	Pan image.
+ or -	Zoom image in Viewer.
Home	Reset view.
R, G, B, A, C	Toggle Red, Green, Blue, alpha, and Color views.

Also, see the table on page 54 for keyboard equivalents to Viewer buttons.

The Viewer Shortcut Menu

Shortcut menus differ depending on the location of the pointer in the interface, or what function/button the pointer is on. The following table shows the shortcut menu commands available in the Viewer.

Menu	Option	Keyboard	Description
Edit	Undo	Command-Z or Control-Z	Undo the last operation. Does not work with *RotoShape* or *QuickPaint*.
	Redo	Command-Y or Control-Y	Redo the last undo command.
View	Zoom In/Out	+ or - (next to the Delete (Mac) / Backspace (Linux) key	Zooms in and out by increments. You can also Control-middle drag or Control-Alt-drag to zoom in or out with non-integer increments.
	Reset View	Home	Sets the Viewer ratio to 1:1. The Viewer ratio is listed in the upper-left corner of the Viewer title bar.
	Fit Image to Viewer	F	Resizes the image to the Viewer boundaries.
	Fit Viewer to Desktop	Shift-F	Fits the Viewer window to the larger desktop window. Does not change the Viewer zoom; it just helps you when resizing the larger Desktop pane.
	Fit Viewer to Image	Control-F	Snaps the Viewer to the image size.
Render	Render Flipbook	. (period)	Renders a non-permanent Flipbook.
	Render Disk Flipbook		Renders a disk-based Flipbook.
	Render FileOut Nodes		Renders *FileOut* nodes to disk.
	Render Proxies		Renders proxy images.

Menu	Option	Keyboard	Description
Clear Buffer A/B			Clears buffer A or B.
New Viewer		N	Creates a new Viewer. If the mouse is over a Viewer, it clones that Viewer.
Delete Viewer			Deletes that Viewer. Helps to clear up graphic/refresh problems.
Minimize or Restore Viewer			Stores the Viewer as a small bar.
Viewer Lookups			Lets you load Viewer lookup controls into the Parameters1 or Parameters2 tab, or into a floating window.
Viewer Scripts			Lets you load Viewer script controls into the Parameters1 or Parameters2 tab, or into a floating window.
Viewer DOD			Lets you load Viewer DOD controls into the Parameters1 or Parameters2 tab, or into a floating window.
View Channel			Like the View Channel button, views the channel you select.

Node-Specific Viewer Shelf Controls

Some nodes, mainly transformations, have onscreen controls to help you interactively control your images in the Viewer. These controls appear whenever the node's parameters are loaded.

When you adjust an active node with onscreen controls, a second row of controls appears in that Viewer, at the top of the Viewer shelf. These controls disappear when you load a different node's parameters.

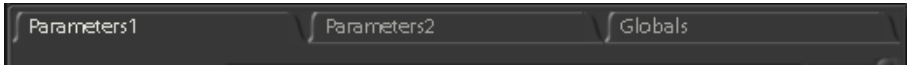
The following table shows the common onscreen control buttons.

Button		Description
	Onscreen Controls–Show	Displays the onscreen controls. Click to toggle between Show and Hide mode.
	Onscreen Controls–Show on Release	Hides onscreen controls while you modify an image. To access this mode, click and hold the Onscreen Controls button, then choose this button from the pop-up menu, or right-click the Onscreen Controls button, then choose this option from the shortcut menu.
	Onscreen Controls–Hide	Turns off the onscreen controls. To access this mode, click and hold the Onscreen Controls button, then choose this button from the pop-up menu, or right-click the Onscreen Controls button, then choose this option from the shortcut menu.
	Autokey	Auto keyframing is on. A keyframe is automatically created each time an onscreen control is moved. To enable, you can also right-click, then choose Onscreen Control Auto Key On. To manually add a keyframe without moving an onscreen control, click the Autokey button off and on.
	Delete Keyframe	Deletes the keyframe at the current frame. This is used because controls for functions such as *Move2D*, keyframes for xPan, yPan, xScale, yScale, and angle are created simultaneously. Delete Key deletes the keyframes from all the associated parameters at the current frame. To delete all keyframes for a parameter, such as *Move2D* on all frames, right-click the Delete Keyframe button and select Delete All Keys.
	Lock Direction–Off	Allows dragging of onscreen controls in both the X and Y directions.
	Lock Direction to X	Allows dragging of onscreen controls in the X direction only. To enable, click and hold the Lock Direction button, then choose this button from the pop-up menu.
	Lock Direction to Y	Allows dragging of onscreen controls in the Y direction only. To enable, click and hold the Lock Direction button, then choose this button from the pop-up menu.
	Onscreen Color Control	Click this swatch to change the color of onscreen controls.
	Path Display–Path and Keyframe	Displays motion path and keyframe positions in the Viewer. You can select and move the keyframes onscreen.

Button		Description
	Path Display–Keyframe	Displays only the keyframe positions in the Viewer. To access this mode, click and hold the Path Display button, then choose this button from the pop-up menu.
	Path Display–Hide	The motion path and keyframes are not displayed in the Viewer. To access this mode, click and hold the Path Display button, then choose this button from the pop-up menu.

The Parameters Tabs

The controls that let you adjust the parameters for each of the nodes in the node tree, as well as the global parameters of your script, are located in the Parameters tabs. Two Parameters tabs let you load parameters and make adjustments for two nodes at once.



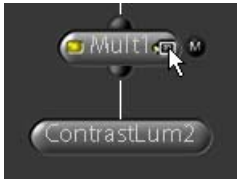
Accessing a Node's Controls Using the Parameters Tabs

You must first load a node's parameters into the Parameters1 or Parameters2 tab in order to make changes to them. The Parameters tabs are empty until you load a node's parameters into them.

To load a node's parameters into the Parameters1 tab:

- Click the right side of the node.

The parameters indicator appears on the right side of the node, and the node's parameters are loaded into the Parameters1 tab. The node does not have to be selected in order to load its parameters into the Parameters1 tab.

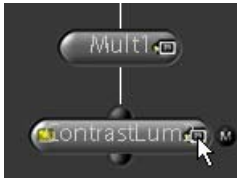


Double-click anywhere on the node to load its parameters into the Parameters1 tab and its image into the Viewer.

To load a node's parameters into the Parameters2 tab:

- Shift-click the right side of the node.

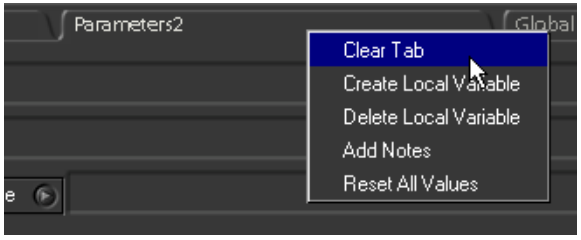
The parameters indicator appears on the right side of the node, and the node's parameters are loaded into the Parameters2 tab. The node does not have to be selected in order to load its parameters into the Parameters2 tab.



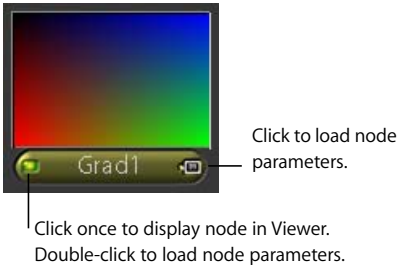
Loading a node's parameters into a tab automatically clears out whatever previous parameters were loaded. If necessary, you can clear a Parameters tab at any time.

To clear a tab so that no parameters are loaded into it:

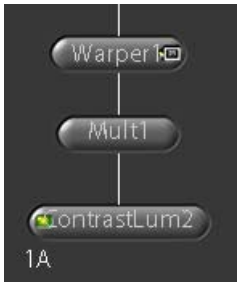
- Right-click the Parameters1 or Parameters2 tab, then choose Clear Tab from the shortcut menu.



It's important to bear in mind that you can load the image from one node into the Viewer, while loading another node's parameters into the Parameters tab.



For example, you can view the resulting image from the bottommost node in a tree, while adjusting the parameters of a node that's farther up in that tree.



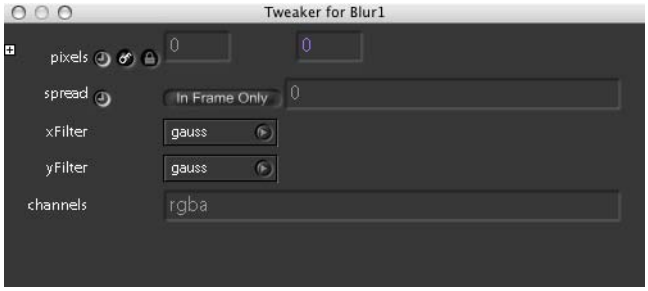
The indicator on the left shows which nodes are loaded into Viewers, and the indicator on the right shows which nodes have been loaded into one of the Parameters tabs.

Using Tweaker Windows

You can also open a node's parameters in a floating "Tweaker" window.

To open a Tweaker window:

- Select a node and press Control-T.



The Tweaker window appears, floating above the Shake window.

Global Parameters

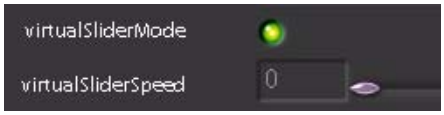
You can double-click an empty area in the Node View to open the Globals tab, or click the Globals tab itself. For more information on the global parameters, see Chapter 2, "Setting a Script's Global Parameters," on page 91.

Adjusting Parameter Controls

The Shake interface incorporates a variety of parameter controls. Many parameters found in the Parameters tab and Globals tab have subparameters. A plus sign (+) beside a parameter indicates that there are related subparameters. Click the plus sign to open the parameter subtree and access the subparameters.

Each parameter has several types of controls that you can use to change that parameter's numerical value.

- Sliders: Move the slider (if available) to modify the parameter's value.
- Virtual sliders: These sliders—controlled by dragging in a value field—allow you to increase or decrease a parameter's value beyond the limits of a standard slider. Drag left or right in a value field to decrease or increase a parameter's numeric value.



Note: If you are using a Wacom tablet, open to the Globals tab, open the guiControls subtree, enable virtualSliderMode, then set virtualSliderSpeed to 0.

When a parameter has an expression attached to it, a plus sign (+) appears beside the parameter name. (The plus sign is also used to indicate a parameter that has hidden subparameters.) An expression can be an animation curve, a link to a different parameter, or a function. Clicking the plus sign beside a parameter that's linked to an expression reveals the expression field. To clear a non-curve expression, move the slider, enter a new value in the value field, or right-click and select Clear Expression. For an introduction to expressions, see "Using Expressions in Parameters" on page 78.

- Some parameters have associated toggle buttons. You can enter a value in the value field, or click the toggle button. You can also enter expressions in the value field.

- Press Tab or Shift-Tab to advance or retreat into adjacent value fields.

Keyframing and Curve Editor Controls

Two controls let you load a parameter into the Curve Editor, and enable it to be animated using keyframes.

The Load Curves button (the clock-shaped button to the immediate right of a parameter name) loads parameters into the Curve Editor.

When the Load Curves button is enabled (checked), the parameter is displayed in the Curve Editor. When disabled, the parameter does not appear in the Curve Editor.

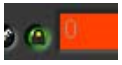
The Autokey button enables keyframing for that parameter.

For more information on animating parameters, see Chapter 10, "Parameter Animation and the Curve Editor," on page 291.

Locking a Parameter

Most parameters have a lock button next to the Autokey button. This control lets you lock that parameter so that it can't be modified.

When you lock a parameter, its value field turns red to indicate that it's locked.



Locked parameters cannot be edited, but if they contain keyframes, an expression, or a link to another parameter, these values continue to animate that parameter.

Using Color Controls

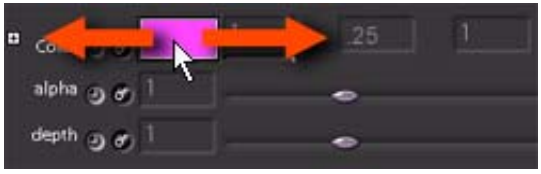
Some parameters have associated Color controls.

To use a Color control, do one of the following:

- Click the Color control (the color swatch)—the Color Picker opens, and you select your color from the Color Picker or Viewer.

- Click the plus (+) sign to the left of the Color control to access color subparameters. The first row in the subtree contains a slider to modify one channel at a time. Select the button that corresponds to the desired channel: (R)ed, (G)reen, (B)lue, (O)ffset, (H)ue, (S)aturation, (V)alue, (T)emperature, (M)agenta-Cyan, or (L)uminance. Move the slider to calculate according to the selected channel, but convert the numbers back to RGB.

- Edit the individual channels or add expressions in the subtree.
- You can also keep the subtree closed, and use the Color control (the color swatch) itself as a virtual slider. Using the channel buttons as the keyboard guide, press and hold the desired key (R, G, B, H, S, V, and so on) and drag left or right in the Color control. In the following illustration, when G is pressed and the pointer is dragged, the green channel increases or decreases.



Note: To adjust the red, green, and blue color channels at the same time, press O and drag in the Color control (O represents Offset).

Using Pop-Up Menus

Some parameters have associated pop-up menus, such as the font parameter in the *Text* node.

There are two ways to choose items from a pop-up menu:

- Click a menu item to choose that item, then close the menu.
- Right-click a menu item to choose it and remain in the menu. When you first click the menu item, hold down the left mouse button and move the pointer off of the menu. Then return the pointer to the menu and right-click. This allows you to quickly test different parameters.

Using Expressions in Parameters

An expression is any non-numeric entry, such as a variable or a mathematical calculation. Any parameter can use an expression. Some expressions, such as *time*, are extremely simple. When you type the expression variable "time" into a value field, Shake returns the numeric value of the current playhead position. For example, if the playhead (in the Time Bar) is parked at frame 1, typing "time" into a value field returns a value of 1 in that field.

Entering an expression consisting of any letter (whether valid or not) activates a plus sign (+) to the left of the parameter name. To edit a parameter's expression, click the plus sign to open an expression field underneath.

If the parameter has an expression and you make an adjustment to that parameter's slider, the expression is removed in favor of your numerical change. If the parameter is animated, however, these special expressions are recognized by Shake and are not removed when the slider is adjusted. For more information on animating parameters, see Chapter 10, "Parameter Animation and the Curve Editor," on page 291.

Note: You can also remove an expression by right-clicking the field and selecting Clear Expression from the shortcut menu.

You can modify expressions in various ways:

- To load or save an expression, use the right-click menu.
- To create extra sliders to build complex expressions (and still allow interactive input), right-click the field and select Create Local Variable. To remove a local variable, right-click and select Delete Local Variable.

For a lesson on using local variables and expressions, see Tutorial 4, "Working With Expressions," in the *Shake 4 Tutorials*.

For a list of mathematical expressions and variables, see Chapter 30, "Installing and Creating Macros," on page 905.

Linking One Parameter to Another

You can link any parameter to any other parameter.

To link parameter A to parameter B within the same node:

- Enter the name of parameter A into the value field of parameter B, then press Return. A plus sign appears to indicate that the parameter now contains an expression.

For example, in a *Move2D* node, you would link yPan to xPan by typing the following into the *yPan* parameter:

```
xPan
```

Note: The default state of many parameters is an expression which links them to an accompanying parameter. This is most common for pairs of parameters that define X and Y values. For example, the default argument for *yScale* is a link to *xScale*.

To link to a parameter in a different node, preface the link with the following syntax:

```
nodeName.parameter
```

For example, to link the red channel parameter of the *Add1* node to the red channel parameter of the *Add2* node, enter the following expression in the *Add1* red channel:

```
Add2.red
```

To interactively copy a parameter from one field to another in the same node, do one of the following:

- Click the parameter name you want to copy, and drag it to the parameter name (a value or expression) that you want to copy the value to. This copies the value from the first field to the second.

Note: This drag and drop behavior also works when you drag color from one Color control to another.

- Select text in a value field and press Command-C or Control-C to copy the information. Go to the second value field and press Command-V or Control-V to paste.
- To interactively link two parameters together, Shift-drag a parameter name and drop it onto the parameter you want to link to. This creates an expression linking back to the first parameter by listing its name.

Combining Links With Expressions

Parameter links can be used in conjunction with mathematical expressions as well. For example, to double the value of a link, enter:

```
Add2.red*2
```

To link to a parameter at a different frame than the current one, use the @@ signs. For example, to link to *Mult1's* red parameter from two frames earlier, use the following expression:

```
Mult1.red@@(time-2)
```

Displaying Parameter Values in the Viewer

You can dynamically display the values of parameters using the *Text* and *AddText* nodes. To differentiate a parameter name from regular text in the value field, surround it with a pair of braces. For example:

```
The current frame is:  {time}
```

displays the following in the Viewer:

The current frame is: x

where x automatically updates as each frame progresses.

In another example, if there is a node called *Gamma1*, and its rGamma value is 1.7, entering the following expression into the *text* parameter of a Text node:

```
My red value = {Gamma1.rGamma}
```

displays the following in the Viewer:

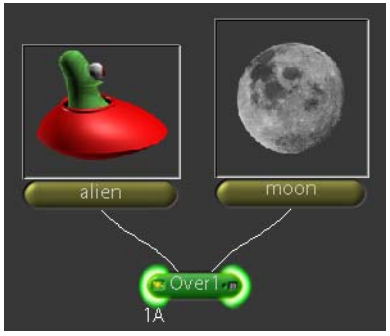
My red value is 1.7

Note: There is a macro called Wedge in the "Cookbook" section of this manual that can be used to print out wedging values for color timing Cineon files.

For a lesson on linking parameters, see Tutorial 4, "Working With Expressions," in the *Shake 4 Tutorials*.

Copying and Pasting Script Code in Shake

If you copy a group of nodes from the node tree, open a text editor, and paste the result, you'll see the actual code that Shake is using to perform the operations those nodes represent. The following screenshots show a simple composite in the Shake interface, and those same three nodes copied and pasted into a text editor.



At times, various script code and expressions are featured in the Shake documentation. Many times, examples and expressions you see presented in a coded format can be copied from the onscreen documentation and pasted into Shake, for immediate use.

The Parameters Tab Shortcut Menu

The following table lists the options that appear when you right-click the top portion of the Parameters tab.

Option	Description
Clear Tab	Unloads the current parameters from the tab.
Create Local Variable	Allows you to create a variable specific to a given node. Use this option when you want to link one or more parameters to other parameters. See Tutorial 4, "Working With Expressions," in the *Shake 4 Tutorials*.
Delete Local Variable	Deletes the local variable for the selected parameter.
Add Notes	A dedicated local variable in string format. Allows you to add notes to any node (to help you remember what you were thinking at the time).
Reset All Values	Resets all values in the node to their default state.

The following table lists the options that are available when you right-click a parameter.

Option	Keyboard	Description
Copy	Command-C or Control-C	Copies the selected nodes onto the Clipboard.
Paste	Command-V or Control-V	Pastes the Clipboard contents into the Node View. You can also copy nodes from the Node View and paste the nodes into a text document, and copy the text and paste it into the Node View.
Load Expression		Loads an expression from disk. The expression should be in Shake format. You can use this if you have a translator for another package's curve types.
Save Expression		Saves the current expression as a text file to disk.
Clear Expression		Clears the current expression.
Clear Tab		Clears the current parameters from the tab.
Create Local Variable		Allows you to create a variable specific to that node. Use this when you want to drive one or more parameters off of other parameters. See Tutorial 4, "Working With Expressions," in the *Shake 4 Tutorials*.
Delete Local Variable		Deletes the local variable for the selected parameter.
Add Notes		A dedicated local variable in string format. Allows you to add notes to any node.

The Domain of Definition (DOD)

The Domain of Definition (DOD) is a rectangular zone that Shake uses to bind the significant pixels in an image in order to optimize rendering speed. Everything outside of the DOD is considered as background (black by default), and is therefore ignored in most computations. Proper handling of the DOD is an extremely powerful way to speed your render times.

To examine the efficiency of the DOD node:

1. Create an Image–*Text* node.
2. Ensure that the Display DOD button in the Viewer is on.

The green box that you see around the text in the Viewer is the DOD.

Because of the DOD, nodes attached to this image process in a fraction of the time it takes to calculate the same nodes with an image that fills the frame.

To test rendering times with DOD:

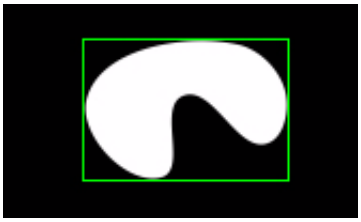
1. Attach a Filter–*RBlur* node to the Text node, and set the *RBlur* oRadius parameter to 360.
2. In a separate branch, create an Image–*Rand* node (to create an entire frame of pixels).
3. Attach a Color–*Monochrome* node to the *Rand* node to turn it into a two-channel image. The *Text* node creates a BWA (black and white, with alpha) image by default, so you must match the channels to compare rendering speeds.
4. Copy and paste the *RBlur1* node into the Node View, then attach the copied node (*RBlur2*) to the *Monochrome1* node.

There is a significant difference in rendering speed, even though both images are the same resolution.

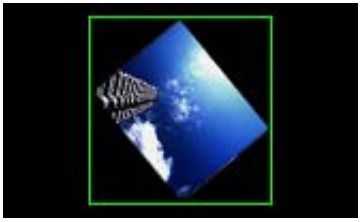
Assigning a DOD

All images from disk are automatically assigned a DOD that is equal to the resolution of the image. There are five ways to alter the DOD:

- Images generated in Shake have a DOD. For example, nodes from the Image tab such as *RGrad*, *Text*, and *RotoShape* automatically have an assigned DOD.

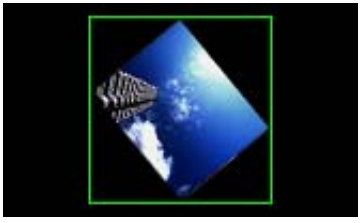


- The DOD of an image from disk that is transformed or filtered is automatically recalculated. For example, the following image is read in (imported) and scaled down and/or rotated with a *Move2D*.



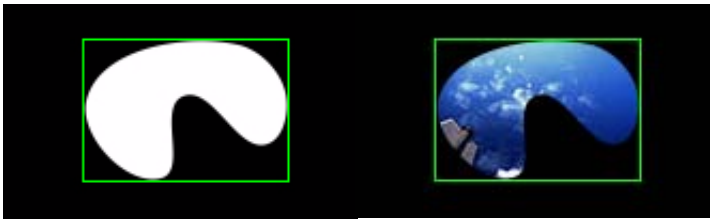
  Also, if an image is blurred, the DOD expands accordingly.

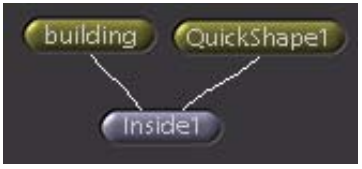
- A rendered .iff file from Shake is embedded with a DOD. When Shake writes an .iff file, it automatically saves the DOD information. Only the .iff format embeds the DOD. In the following example, the image that was written out in the previous (above) example is read back into Shake.



- The *SetDOD* node, located in the Transform tab, allows you to manually assign a DOD to an image. In the following illustration, a *SetDOD* node is attached to the building image to limit the effects to the tower.

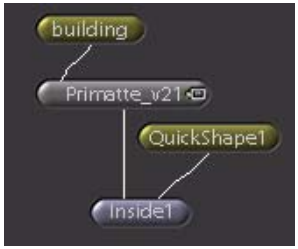
- You can combine multiple images using a DOD. When you combine two images, the DODs combine to form a larger rectangle. If, however, you use a node like *Inside* or *IMult*, that node takes the DOD of the second node. If the building image is placed *Inside* of the *QuickShape* image from above, it inherits the DOD of the *QuickShape* node.





Note: When using onscreen controls to edit a shape (for example, a *Rotoshape* or *QuickPaint* object) that has control points within the boundaries of the DOD, move the pointer over the shape inside of the DOD, and then drag to select the points.

Combining images with a DOD is an excellent way to optimize greenscreen or bluescreen images that need to be cropped via a garbage matte anyway, because it simultaneously removes the garbage areas and assigns an efficient DOD to the image.



The node tree above produces the folowing effect:

Building

QuickShape1

Primatte1

Inside1

With a good understanding of the role of the DOD, you can optimize the tree before and after the node in question. The above example not only optimizes any nodes you attach to *Inside1*, but executes the *Primatte* and reads in the part of the image that is inside of the DOD, reducing processing and I/O activity.

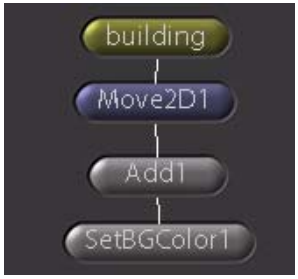
Keying, Color Correcting, and the Background Color

This section discusses the area outside of the DOD, which is called the Background Color (BGColor).

The two main keyers in Shake, *Keylight* and *Primatte*, recognize the background color, and have a toggle to key the background color in or out. By default, the keyer leaves the background area black in the alpha channel. To turn the background completely white, toggle BGColor on.

Shake processes color correction of the BGColor very quickly, as it recognizes there is a pure correction applied to previously black pixels. If the color correction does not change black, such as *Gamma* or *Mult*, it is ignored. If it does affect the black areas, as does *Add* or *Compress*, it processes these areas, but understands that they are still the result of a lookup process. Therefore, the DOD does not get reasserted to the resolution frame. This is the same process that is used when the Infinite Workspace kicks in. So, even though the pixels outside of the DOD are not visibly different from the pixels inside, the DOD remains in place. (For more information, see Chapter 7, "Using the Node View," on page 217.)

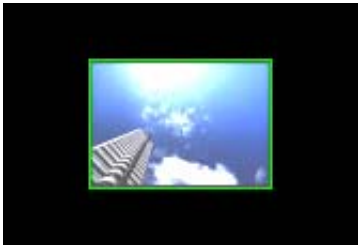
There may be cases, however, where you want to take advantage of the DOD for masking purposes. In this tree, an image is scaled down, and the brightness increased with an *Add* node. This, however, turns the area outside of the image a medium gray. Since this area is recognized as outside of the DOD, it can be returned to black with a Color–*SetBGColor* node, which sets the color for the area outside of the DOD.



Building

Move2D1

Add1



SetBGColor1

The Layer–*Constraint* node also limits a process. For more information on masking using the Constraint node, see Chapter 19, "Using Masks," on page 527.

The Time Bar

The Time Bar, at the bottom of the Shake window, displays the currently defined range of frames, the playback buttons, and the Info field, which provides a brief description of each control you move the pointer over.

The Time Bar is a display of the currently defined time range. It neither limits nor controls the actual parameters that are saved into the script. To set the frame range that renders via a *FileOut* or Flipbook operation, go to the Globals tab and enter the frame range in the timeRange parameter.

Setting a Script's Frame Range

The number in the field to the left of the bar is the start frame of the Time Bar, and the number in the field to the right is the end frame. In the above example, frames 1 to 21 are highlighted. This corresponds to an entry in the timeRange parameter of the globals of 1-21.

Time Bar Navigation Values

The Current Frame field indicates the position of the playhead, which is 49. The Increment parameter controls how far the playhead advances when you press the Left Arrow or Right Arrow key.

- The default value of 1 means that every frame is played back.
- A default value of 0.5 enables you to see each field when a video clip is loaded into the Viewer.
- A value of 2 or higher means that Shake skips frames. At 2, every other frame is skipped.

When you move the pointer within the Time Bar, the frame number that you'll jump to when you click to reposition the playhead is displayed over the pointer.

If you have already set the timeRange parameter in the Globals tab, click Home in the Time Bar controls to use the timeRange as the Time Bar frame range.

To change the current time and display that frame in the Viewer, do one of the following:

- Click or drag the playhead to interactively scrub across the current time range.
- To jump to a specific frame, type the frame number into the Current Frame field, and press Return. As with any value field, you can use the virtual sliders—press Control and drag the pointer left and right in the value field.

- To pan across the Time Bar, press the middle mouse button and drag; or Option-click or Alt-click and drag.
- To zoom into or out of the frame range displayed by the Time Bar, press Control and the middle mouse button; or Control-Option-click or Control-Alt-click, then drag.

Playback Controls

The controls illustrated below play through the script according to the Time Bar frame range, not the global timeRange.

- To play forward, click the forward arrow button.
- To play backward, click the backward arrow button.

  Note: Regardless of the speed of your computer, Viewer playback is now limited to the frame rate specified in the framesPerSecond parameter in the Format section of the Globals tab.

  Assuming your composition is cached so that real-time playback is possible, this playback rate is not exact, but may vary by around 10 percent.
- To stop playback, click the stop button, or click the left mouse button anywhere on the screen.
- Shift-click a playback button to render all frames in the current frame range and store the frames in memory. The sequence will play back much faster next time.
- Click the keyframe buttons to jump to the previous or next keyframe.

The following table lists additional keyboard shortcuts.

Keyboard	Description
Left Arrow key or Right Arrow key	Retreat/advance a frame based on the frame Increment setting (works in any window).
Up Arrow key or Down Arrow key	Jump to next/previous keyframe.
.	Play forward.
Shift-.	Begin cached playback.
Home	Fit the current time range into the Time Bar.
T	Toggle timecode/frame display.

For more information, see Chapter 8, "Using the Time View," on page 261.

Previewing Your Script Using the Flipbook

You can render a temporary Flipbook to preview your work. Once the Flipbook has rendered into RAM, use the playback buttons (described below) to play back the Flipbook. The Flipbook is available on Mac OS X and Linux systems.

To launch the Flipbook from the interface:

1. In the Globals tab, set the timeRange, for example, 1-50 or1-50x2.
2. Load the node that contains the image you want to preview into the main Viewer.
3. Click the Flipbook button in the Viewer shelf.

A Flipbook window appears, and the specified timeRange is rendered into RAM for playback.

4. When the render is finished, press the period or > key to play the result. When you're finished viewing the Flipbook, close the window and it disappears from memory.

On a Mac OS X system, you also have the option to create a disk-based QuickTime Flipbook. For more information on using both RAM and disk-based Flipbooks, see Chapter 11, "The Flipbook, Monitor Previews, and Color Calibration," on page 323.

Setting a Script's Global Parameters

2

This chapter covers how to set the global parameters within each script, tailoring your script's properties to fit your needs.

About Global Parameters

When you create a new script, you should customize its global parameters before starting work on your composite. The Globals tab contains parameters that are commonly found in the Project Properties window of other applications. These parameters include a script's time range, default frame width and height, aspect ratio, proxy settings, global motion-blur controls, bit depth, field rendering settings, and various ways to customize Shake's controls.

The global parameters also contain a group of guiControl parameters that let you customize how Shake works. Using the guiControl parameters, you can specify whether thumbnails are exposed, how many threads Shake uses on multi-processing computers, the colors used by shapes and noodles, and the sensitivity of shape controls in the Viewer.

Setting Global Variables From the Command Line

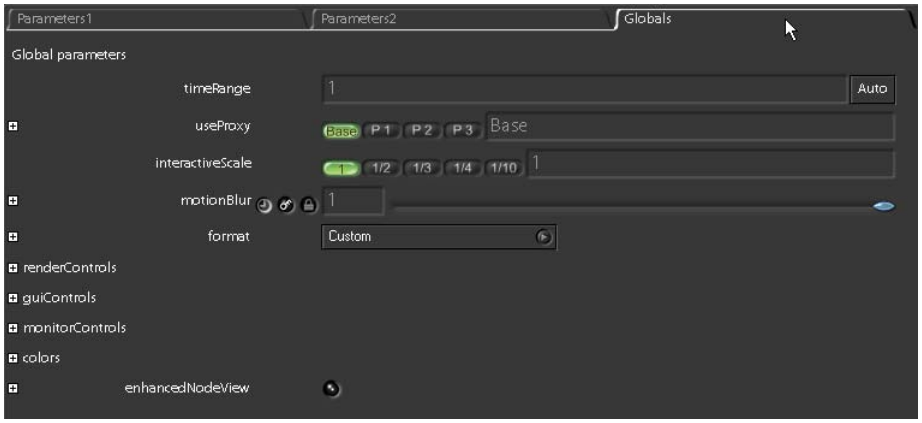
Many of the parameters described in this chapter can be set in the command line at the time you launch Shake, so you don't have to reset them each time you write out a script. For example, your timeRange may be 1-10, but you can modify that when you render on the command line with the *-t* option:

```
shake -exec my_script -t 1-240
```

To access the global parameters, do one of the following:

- Click the Globals tab.
- Double-click an empty area in the Node View.

Note: The global controls also appear in the Parameters1 tab when Shake is first started, or whenever you create a new script.



The global parameters that can be seen in the Globals tab of the Shake interface are divided into several groups.

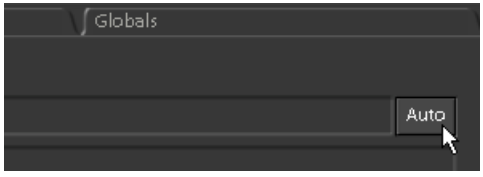
The Main Global Parameters

These parameters control the duration and format of the output from your script. While these parameters can be changed at any time, it's a good idea to set them to the proper values before you begin any serious compositing.

timeRange

This parameter defines the number of frames in your project. This parameter can be changed at any time. The timeRange is generally represented by a starting value and an ending value, separated by a dash. For example, a 10-second clip in a project that's set to a frame rate of 24 fps would have a timeRange parameter set to "1-240."

The fastest way to set the timeRange is to open the Globals tab and click Auto, which is located to the right of the timeRange field.



Clicking Auto automatically populates the timeRange parameter by calculating the duration from the earliest frame in any *FileIn* node to the last frame in any *FileIn* node.

The starting frame does not always have to be set to 1. For example, to quickly trim off the first 20 frames of your project, change the timeRange to "21-240." Doing this restricts the frame range displayed in the Time Bar and the processing and rendering of your script to only the frames you need.

Here are some more examples of frame ranges you can define in Shake.

Time Range	Number of Frames	Frames Rendered
1-100	100	1, 2, 3... 100
1-100x2	50	1, 3, 5... 99
1-100x20	5	1, 21, 41... 81
1-20, 30-40	31	1, 2, 3... 20, and 30, 31, 32... 40
1-10x2, 15, 18, 20-25	13	1, 3, 5... 9, 15, 18, 20, 21, 22 ... 25
100-1	100	100, 99, 98... 2

Several parameters and controls in Shake either inherit the timeRange parameter directly, or allow you to assign its value:

- The Home button in the playback controls
- The Render Parameters window
- The Time Range parameter in the Mixdown Options of the Audio Panel
- The Flipbook Render Parameters window

useProxy

A proxy is a lower-resolution image that can be temporarily substituted for the high-resolution plates in your script, allowing you to work and see rendered tests faster. Because the images are smaller, you drastically decrease the processing time, memory requirements, and the amount of time spent on reading and writing files as you work. Naturally, the trade-off is that the quality of the image displayed in the Viewer suffers as well, which is why proxies are generally used only when creating low-resolution comps or creating test previews. After assembling a script using proxies, you can return your script to the original, full resolution in order to render the final output.

These controls are linked to the proxy buttons found at the upper-right corner of the Shake interface, and allow you to switch among different resolutions to reap the aforementioned benefits. For more information on using proxies, see Chapter 4, "Using Proxies," on page 137.

interactiveScale

If the general processing speed for your operations is fine, but the interactivity of processor-intensive operations is slowing you down, you can turn on the InteractiveScale option in the Globals tab to use a proxy resolution only while you're adjusting parameters. This option does not affect your Flipbooks or *FileOut* renders. For more information, see "Using interactiveScale" on page 139.

motionBlur

In Shake, motion blur can be applied to animated transformation parameters. Each transform node has its own motion blur settings, so you can tune each one individually. The motionBlur parameters in the Globals tab either adjust or replace the existing values within each node in the script, depending on the parameter. You can also set the global motionBlur value to 0 to turn all motion blur within your project off. For more information on using motion blur, see "Creating Motion Blur in Shake" on page 778.

The Format Pop-Up Menu

The format pop-up menu provides a fast way of simultaneously setting all the format subparameters found within the format parameter subtree. The format pop-up menu contains many of the most popular media formats.

Name	default Width	default Height	default Aspect	default ViewerAspect	framesPerSecond
Academy	1828	1332	1	1	24
CinemaScope	1828	1556	.5	2	24
Full	2048	1556	1	1	24
1.85	1828	1332	1	1	24
HDTV1080i/p 30FPS	1920	1080	1	1	30
HDTV1080i/p 29.97 FPS ND	1920	1080	1	1	29.97
HDTV1080i/p 29.97 FPS DF	1920	1080	1	1	29.97
HDTV1080i/p 25 FPS	1920	1080	1	1	25
HDTV1080i/p 24 FPS	1920	1080	1	1	24
HDTV1080i/p 23.98 FPS	1920	1080	1	1	23.98
NTSC ND (D1 4:3)	720	486	1.1111	.9	29.97
NTSC DF (D1 4:3)	720	486	1.1111	.9	29.97
NTSC ND (16:9)	720	486	.83333	1.2	29.97
NTSC DF (16:9)	720	486	.83333	1.2	29.97

Name	default Width	default Height	default Aspect	default ViewerAspect	framesPerSecond
PAL (D1 4:3)	720	576	.9380	1.066	25
PAL (16:9)	720	576	.7032	1.422	25
PAL (square)	768	576	1	1	25

If the format you need is not in this list, you can always open up the format parameter subtree—by clicking the "+" (plus) icon to the left of the parameter name—and create your own custom format.

These settings are only for Shake-generated image nodes—they have no effect on the resolution or frame rate of media referenced by *FileIn* nodes. Shake generated nodes, such as *RotoShape*, *QuickPaint*, *Ramp*, and *Grad* inherit the global resolution.

Click the "+" (plus) icon to reveal the format parameter subtree, which contains the following subparameters:

framesPerSecond
This parameter limits the speed of playback from the Time Bar, and also sets the default playback rate of launched Flipbooks. Three buttons provide the three most common frame rates, Film at 24 fps, NTSC video at 29.97 fps, and PAL video at 25 fps. A value field allows you to enter a custom frame rate to accommodate any other format.

Note: To change the playback rate within the Flipbook, press + and – (on the numeric keypad). The current frame rate is displayed at the top of the Flipbook.

timecodeMode
Sets how timecode is calculated within your script, as 24 fps, 25 fps, 30 fps drop frame, or 30 fps non-drop frame. This parameter is unrelated to timecode that might be present in a QuickTime movie.

Note: Shake does not import timecode associated with QuickTime movies.

defaultWidth, defaultHeight
The width and height of the frame for Shake-generated images. See the above table for standard frame sizes.

defaultAspect
The pixel aspect ratio used for Shake-generated images. This should be set to match the format of the images you're reading into your script. For example, since most standard-definition video formats have nonsquare pixels, the aspect ratio of NTSC video is 1.111, while that of PAL video is .9380. Academy ratio film, which has square pixels, is simply 1. For more information on pixel aspect ratios, see "About Aspect Ratios and Nonsquare Pixels" on page 209.

defaultViewerAspectRatio
This value corrects the aspect ratio of the image displayed by the Viewer to account for images using nonsquare pixels. The defaultViewerAspectRatio parameter is for display only, and has no effect on rendered output.

Changing any format subparameter sets the format pop-up menu to Custom. If there's a particular custom format that you use frequently, you can add it to the Format pop-up list. For more information on adding entries to the format pop-up menu, see "Customizing the Format Pop-Up Menu" on page 96.

defaultBytes
Sets the default bit rate for Shake-generated images. The defaultBytes parameter has no effect on images that are read in via *FileIn* nodes, nor does it affect the rendered output from your script.

viewerZoom
The zoom level applied to the Viewer. This value has no effect on the output resolution of your script.

viewerAspectRatio
When set to formatDefault, this parameter scales the X axis of the Viewer by the defaultViewerAspectRatio parameter, located within the format subtree. When this parameter is set to custom, you can change it to whatever value you want. This is usually used to compensate for the nonsquare pixel ratios of video. For anamorphic film frames, you typically use the proxyRatio to scale down the Viewer's Y axis.

Customizing the Format Pop-Up Menu

You can create your own formats in a *startup.h* file. In *$HOME/nreal/include/startup*, add a line in the following format:

```
DefFormatType("Name", defaultWidth, defaultHeight, defaultAspect,
    defaultViewerAspectRatio, framesPerSecond, fieldRendering)
```

For example:

```
DefFormatType("NTSC (D1 4:3)", 720, 486, 1/.9f, 0.9f, 29.97,0);
```

renderControls

These parameters affect how Shake renders material processed by the currently open script.

fieldRendering
When fieldRendering is set to 0, progressive scan/full frames are rendered. When set to 1, the odd field takes precedence, meaning it is the first line at the top. For more information on setting up your script to render fields properly, see "The Basics of Processing Interlaced Video" on page 191.

quality
When this parameter is set to lo (0), anti-aliasing is disabled. This results in poorer image quality, but improved render speed.

maxThread
Set the maxThread to the number of available processors you want to use for rendering by Shake.

cacheMode
The cache is a directory or precalculated images with script information attached. When Shake evaluates a node tree at a given frame, it compares the tree to the cache to see if it has rendered that frame before. If it has, it calls up the cached image rather than recalculate the entire tree in order to save time. Shake keeps track of how many times each cached frame has been viewed, eliminating the least viewed frames first when the cache runs out of room.

You can set the cacheMode to one of four states:

- *none:* Cache data is neither read from nor written to.
- *read-only:* Preexisting cache data is read, but no new cache data is generated.
- *regular:* The cache is both read from and written to, but only nodes with non-animated values are cached.
- *aggressive:* The cache is both read from and written to, and nodes with animated *and* non-animated parameters are cached.

When setting the cacheMode, consider the following guidelines:

- In most circumstances, the regular cacheMode setting should be used.
- Consider setting the cacheMode to aggressive when you are constantly scrubbing back and forth between two or three frames (for example, when tweaking tracking or shape control points).
- You should only set cacheMode to none if you are using Shake on a system with extremely limited RAM and disk space. By setting the cacheMode to none, Shake is forced to re-compute each image that you select to view, which is the least efficient way to run.

For more information on Shake's caching system, see Chapter 13, "Image Caching," on page 343.

macroCheck
If you open or load a script on your system, and the script does not appear, the script may contain macros that are not on your system. If this is the problem, a message similar to the following appears in the Console tab:

```
line 43:  unknown function MissingMacro
    MissingMacro1=MissingMacro(Keylight_1v41);
```

To open or load a script that contains a missing macro:

1. Click the Globals tab.
2. Expand the renderControls subtree.
3. Set macroCheck to one of the following options:
   - *abort load:* does not load the script
   - *sub. with text:* substitutes a *Text* node in place of the missing macro
   - *sub. no text:* substitutes a *MissingMacro* node
4. Open/load the script.

To set the default macroCheck behavior to substitute a *MissingMacro* node, include the following in a .h file:

```
sys.useAltOnMissingFunc = 2
```

For more information on .h files, see "Creating and Saving .h Preference Files" on page 355.

guiControls

The parameters within the guiControls subtree allow you to customize the functionality and display of controls in Shake's graphical user interface. These settings are individually saved by each script you create.

displayThumbnails
Turns all thumbnails in the Node View on and off. For more information on customizing the thumbnail display, see "Customizing Thumbnail Display" on page 253. displayThumbnails has three subparameters—thumbSizeRelative, thumbSize, and thumbAlphaBlend.

thumbSizeRelative
Scales all thumbnails to the same size, or leaves them at different sizes relative to the original sizes of the images. By default, all thumbnails are displayed at the same width. To display thumbnails at their relative sizes, turn on thumbSizeRelative.

thumbSize
Lets you adjust the size of thumbnails in the Node View. If thumbSizeRelative is turned on, all nodes are resized relative to one another.

thumbAlphaBlend
Turns thumbnail transparency on and off. When thumbAlphaBlend is on, moving one thumbnail over another results in a fast look at how the nodes might appear when composited together in the Viewer. More usefully, it gives you an instant view of which images have transparency in them.

virtualSliderMode
When this parameter is turned off, dragging within any parameter's value field in Shake results in an edit bar appearing and the contents of that field being selected. When this parameter is turned on, dragging within a parameter's value field results in that parameter being modified as if you were dragging a slider. This mode is very useful when using Shake with a graphics tablet. You can also use these virtual sliders in the value fields simply by dragging with the mouse.

virtualSliderSpeed
Adjusts the speed of the virtual slider. When using a stylus, it is recommended you set this parameter to 0.

noodleTension
Lets you adjust how much "slack" there is in the way noodles are drawn from knot to knot. Higher values introduce more slack, and noodles are more curved. Lower values reduce the slack, and noodles are drawn in more of a straight line.

shapeControls
These subparameters allow you to customize the spline-based shape drawing and editing behaviors and transform controls in the Viewer. You can change these parameters to make it easier to use Shake's controls for your individual needs.

rotoAutoControlScale
An option which, when enabled, increases the size of the transform controls of shapes based on the vertical resolution of the image to which the shape is assigned. This makes it easier to manipulate a shape's transform control even when the image is scaled down by a large ratio.

rotoControlScale
A slider which allows you to change the default size of all transform controls in the Viewer when rotoAutoControlScale is turned on.

Note: You can also resize every transform control appearing in the Viewer by holding the Command key down while dragging the handles of any transform control in the Viewer.

rotoTransformIncrement
This parameter allows you to adjust the sensitivity of shape transform controls. When this parameter is set to lower values, transform handles move more slowly when dragged, allowing more detailed control. At higher values, transform handles move more quickly when dragged. A slider lets you choose from a range of 1-6. The default value is 5, which matches the transform control sensitivity of previous versions of Shake.

rotoPickRadius
This parameter provides the ability to select individual points on a shape that fall within a user-definable region around the pointer. This allows you to easily select points that are near the pointer which may be hard to select by clicking them directly. A slider allows you to define how far, in pixels, the pointer may be from a point to select it.

rotoTangentCreationRadius
This parameter lets you define the distance you must drag the pointer when drawing a shape point to turn it into a Bezier curve. Using this control, you can make it easier to create curves when drawing shapes of different sizes. For example, you could increase the distance you must drag, to avoid accidentally creating Bezier curves, or you can decrease the distance you must drag, to make it easier to create Bezier curves when drawing short shape segments.

gridWidth, gridHeight
Specifies, in pixels, how wide and tall each rectangle of the grid is. The gridHeight is locked to the gridWidth by default, although this expression can be changed. This default is 40 x 40 pixels.

gridEnabled
Lets you control the grid's effect on the nodes that you create. There are two settings: on and off. This parameter also toggles the background grid pattern in the Node View if gridVisible is turned on.

gridVisible
Displays the grid as a graphical background in the Node View. This graph is only displayed when gridEnabled is turned on.

layoutTightness
This parameter affects the Layout Arrangement commands described in "Arranging Nodes" on page 244. It lets you specify how closely nodes should be positioned to one another when they're newly created, or whenever you use one of the arrangement commands. This parameter's default is 40 pixels.

consoleLineLength
The maximum line length of information displayed in the Console tab. This defaults to 120 characters.

multiPlaneLocatorScale
Affects all *MultiPlane* nodes within the script. This parameter scales the depth of the virtual space used to distribute the locator points that are displayed in the Viewer (which represent 3D-tracking data clouds that are imported from .ma files). This parameter allows you to expand or compress the relative distance from the camera to the tracked background plate. Adjusting this parameter lets you more easily position layers in space when camera tracking data comes from a subject that's either very far away, or very close. This parameter is for reference only, and has no effect on the data itself. The default multiPlaneLocatorScale value is 50.

Monitor Controls

The monitorControls parameters affect the image that is output to a video monitor using a supported video output device. For more information about outputting to a second display, see "Viewing on an External Monitor" on page 330.

broadcastViewerAspectRatio
By default, this parameter is a link, to `script.defaultViewerAspectRatio`, which mirrors the setting in the format subtree of the Globals tab. When first launched, Shake looks at the system's monitor card and outputs the proper aspect ratio based on the format you select in the Globals tab. For example, if you have a D1 card and you select NTSC D1 from the format parameter, Shake displays nonsquare pixels in the Viewer and sends square pixels to the video monitor.

Note: If you change the value of the broadcastViewerAspectRatio using the slider or the value field, the link to defaultViewerAspectRatio is removed. As with all Shake parameters, you can enter another expression in the broadcastViewerAspectRatio parameter to reset it.

broadcastHighQuality
When broadcastHighQuality parameter is turned on, the image is fit to the size of the broadcast monitor in software mode (rather than hardware mode). The broadcastHighQuality parameter applies a scale node and a resize node, instead of using OpenGL. The broadcastHighQuality parameter is enabled by default.

broadcastGammaAdjust
Lets you adjust the gamma of your broadcast monitor to insure proper viewing (for example, if you are sending an SD NTSC signal to an HD monitor).

broadcastMonitorNumber
By default, Shake looks for the first available monitor with an SD or HD resolution to use as the external monitor. If you have more than one monitor card installed on your computer, this parameter lets you choose which monitor to use.

Note: The external display monitor doesn't have to be a broadcast display. If you have more than one computer display connected to your computer, the second one can be used as the external preview display.

Colors

These parameters allow you to customize the colors Shake uses for different Shake interface controls.

fontColor
This parameter lets you customize the color used by the text within the Shake interface. It simultaneously affects the text of settings in the Parameters tabs, the tab names, node names, and other text in the Node View, Curve Editor, and other Shake tabs.

shapeColors
These subparameters let you customize the colors used by splines generated by various nodes in the Viewer. A large collection of these parameters is used by the *Warper* and *Morpher* nodes to help you distinguish among the specific uses of each type of spline.

noodleColor
Lets you change the base color of noodles in the Node View. Noodles are white by default, but you can use the Color control to change this to anything you like.

noodleColorA, -BW, -BWA, -RGB, -RGBA, -Z, -AZ, -BWZ, -BWAZ, -RGBZ, -RGBAZ
When noodleColorCoding is turned on (in the enhancedNodeView subtree), noodles are color coded and stippled (see "enhancedNodeView" below), based on the bit depth and number of color channels being propagated by each noodle in your node tree. When turned off, noodles appear using the default NoodleColor.

Different combinations of color channels are represented by different colors, and this group of parameters lets you customize the color used for each representation. For more information on noodle color coding, see "Noodle Display Options" on page 224.

gridColor
When gridVisible is on, the grid in the Node View is drawn using this color.

enhancedNodeView

Unlike most other guiControls parameters, which have two toggle states (off and on), each of the parameters in the enhancedNodeView subtree—showTimeDependency, showExpressionLinks, showConcatentationLinks, and noodleColorCoding—has three states. They can be always off, always on, or set to follow the state of the enhancedNodeView parameter.

enhancedNodeView
This parameter allows you to toggle all four enhanced Node View parameters using a single button. This parameter can also be toggled using the Enhanced Node View command from the Node View shortcut menu (Control-E or Command-E). For more information on the enhancedNodeView parameters, see "Using the Enhanced Node View" on page 221.

showTimeDependency
This parameter, when turned on, draws a bluish glow around nodes that are animated. This includes nodes that consist of *FileIn* nodes that reference a QuickTime movie or multiple-image sequence, nodes containing keyframed parameters, or nodes utilizing expressions that change a parameter's value over time.

showExpressionLinks
Turning this parameter on draws a light purple line connecting a node that uses an expression to reference another node, and the node it references. An arrow pointing toward the referenced node indicates the relationship.

ShowConcatenationLinks
When this parameter is turned on, a green line connects a series of nodes that concatenate together. For example, three transform nodes that have been added to a node tree in sequence so that they concatenate appear linked with a green line connecting the left edge of each node. As a result, nodes that break concatenation are instantly noticeable.

Note: As is often repeated, node concatenation is a very good thing, and you are encouraged to group nodes that will concatenate together whenever possible to improve the performance and visual quality of your scripts.

noodleColorCoding
When noodleColorCoding is turned on, noodles are color coded and stippled based on the bit depth and number of color channels being propagated by each noodle in your node tree. When this parameter is turned off, noodles appear in the default NoodleColor.

stipple8Bit, stipple16Bit, stipple32Bit
Bit depths are represented by varying dotted (stippled) lines.These parameters let you customize the stippling used for each bit depth supported by Shake, choosing from five stippling patterns.

Application Environmental Variables

The default values of many of the global parameters can be customized via .h preference files. For example, if you consistently set one or more global parameters to a custom state whenever you create a new script, you can set custom defaults so that new scripts are created with your preferred settings.

Custom global values you set with .h files are only applied to newly created scripts. Once a script is saved, these values are saved within that script, and opening that script recalls the global settings saved within.

Custom Variable Loading Order

When you create .h files to customize Shake's functionality, there is an order of precedence used to load them.

- The default Shake settings found in the *nreal.h* and *nrui.h files* are loaded first.
- Settings in custom .h files load second, according to their own load order.
- Settings in .user files are loaded third.
- Finally, any variables found in a Shake script itself are loaded last, overwriting all previous settings.

Customizing Shake's Default Settings

Unlike many applications that control user customizable settings with a preferences window, Shake provides access to a wide variety of functionality using a system of user-created preference files. For more information on creating and using custom preference files, see Chapter 14, "Customizing Shake."

Script Environmental Variables

The following are variables with states that are not customized via .h files, and are only saved within Shake scripts.

Global Parameter	Type	Purpose
SetTimeRange(const char *range)	char	Range of frames displayed in the Time Bar.
SetTime(float /*frameNumber*/)	float	The frame number of the current position of the playhead.
SetFieldRendering(const char *mode)	char	Whether or not field rendering is turned on for rendered output.
SetFps(const char *fps)	char	The frame rate of the script.
SetMotionBlur(const char *mb, const char *st, const char *so)	char	Whether or not motion blur is turned on.
SetQuality(const char *quality)	char	The quality of the script, enables antialiasing for certain functions.
SetProxyScale(const char *proxyScale, const char *proxyRatio)	char	The default proxy scale of the script.

Global Parameter	Type	Purpose
SetUseProxy(const char *useProxy)	char	The default proxy setting.
SetProxyFilter(const char *proxyFilter)	char	The default filter used to scale proxies.
SetPixelScale(const char *pixelScale, const char *pixelRatio)	char	A temporary setting for the proxy resolution that is overwritten when useProxy is set.
SetUseProxyOnMissing(const char *useProxyOnMissing)	char	When active, substitutes a proxy generated from an associated proxy file when a missing image is encountered.
SetFormat(const char *s)	char	The format that's selected by default.
SetDefaultWidth(int width)	int	The width of Shake-generated images.
SetDefaultHeight(int height)	int	The height of Shake-generated images.
SetDefaultBytes(int bytes)	int	The bit depth of Shake-generated images.
SetDefaultAspect(float aspect)	float	The default aspect ratio of Shake-generated images.
SetDefaultViewerAspect(float aspect)	float	The value used to correct the image displayed in the Viewer to display nonsquare images.
SetTimecodeMode(const char* timecodeMode)	char	How Timecode is calculated in your script.
SetMacroCheck(int mc)	int	Sets macroCheck. 1 = abort load, 2 = substitute with text, and 3 = subsititute no text.
SetDisplayThumbnails(const char *displayThumbnails)	char	Whether or not to display thumbnails in the Node View.
SetThumbSize(const char *thumbSize)	char	Sets the size of thumbnails in the Node View.
SetThumbSizeRelative(const char* thumbSizeRelative)	char	Turns thumbSizeRelative off and on.
SetThumbAlphaBlend(const char *thumbAlphaBlend)	char	Turns thumbAlphaBlend off and on.

Adding Media, Retiming, and Remastering

3

This chapter covers adding media to your script using FileIn nodes, either as individual files, or as media from Final Cut Pro. Also discussed are the retiming and remastering functions available from within the FileIn node itself.

About Image Input

This section discusses importing images into a Shake script using the *FileIn* node. It also presents other procedures associated with the *FileIn* node, including associated file paths, temporary files and disk space, basic time shifting, and retiming footage. For more information on supported file formats and other issues regarding media, see Chapter 5, "Compatible File Formats and Image Resolutions," on page 167.

Adding Media to a Script

Usually, the first step in any composite is to add one or more *FileIn* nodes, which import or "read in" source media files on disk into Shake's node tree. Although there are many other image nodes that can be used to generate images directly within Shake (the *Checker*, *ColorWheel*, *Ramp*, and *RGrad* nodes are four examples), image sequences and QuickTime media files must be added to your script using the *FileIn* node. Each media element read into the node tree requires a separate *FileIn* node.

To add media to your script:

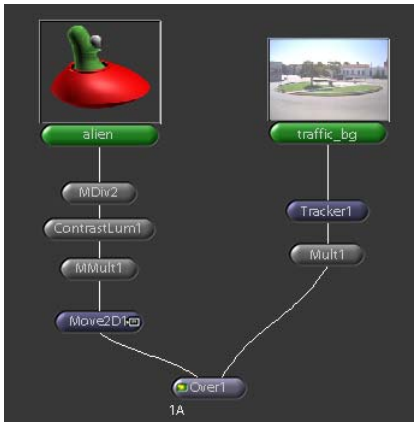
1. Create a *FileIn* node by doing one of the following:
   - Click the *FileIn* node in the Image tab.
   - Right-click in the Node View, and then choose *FileIn* from the Nodes > Image shortcut submenu.
   - On Mac OS X, choose Tools > Image > FileIn from the menu bar.
2. When the File Browser appears, select one or more files, then click OK.

The selected media appears in the Node View, represented by one or more *FileIn* nodes. For more information about finding and selecting files, see "The File Browser" on page 38.

Dragging and Dropping Media Into Your Script

If you're running Shake on Mac OS X, you can drag supported media types from the Finder directly into the Node View tab. This results in the creation of a *FileIn* node corresponding to each file you dragged in.

By default, *FileIn* nodes appear with a thumbnail of the first frame of the media they represent. In this example, the two highlighted *FileIn* nodes at the top of the node tree provide the source images that are modified and combined further down in the tree.



After you've finished creating the necessary effect in Shake, you export your finished shot by attaching a *FileOut* node to the section of the node tree that you want to write to disk. For more information on outputting images using the *FileOut* node, see Chapter 12, "Rendering With the FileOut Node."

Image Sequence Numbering

When referring to an image sequence, you can specify frame padding by adding special characters to the file name you enter:

- The # sign signifies a four-place padded number.
- The @ sign signifies an unpadded number.
- The %d characters signify either a padded or unpadded number, depending on the numbers placed between the two characters.

You can also use several @ signs to indicate padding to a different number. (For example, @@@ signifies 001.)

The following table lists some formatting examples.

Shake Format	Reads
image.#.iff	*image.0001.iff, image.0002.iff*
image.%04d.iff	*image.0001.iff, image.0002.iff*
image.@.iff	*image.1.iff, image.2.iff*
image.%d.iff	*image.1.iff, image.2.iff*
image.@@@.iff	*image.001.iff, image.002.iff*
image.%03d.iff	*image.001.iff, image.002.iff*

The above examples assume an exact relation between the current frame processed in Shake, and the frame read in. For example, at frame 1, *image.1* is read in. If you are reading the images in from the interface with Sequence Listing enabled in the File Browser, you see the actual sequence in the "File name" field. For example:

Image	Shake Syntax With Sequence Listing
image.4.iff, image.5.iff ... image.10.iff	*image.4-10@.iff*
image.4, image.5.iff, image.6.iff, image.10.iff	*image.4-6,10@.iff*

Unlike the previous examples, these offset the clip timing by placing *image.4.iff* at frame 1 and *image.5.iff* at frame 2. In the first example, *image.10.iff* is placed at frame 7. In the second example, *image.10.iff* is placed at frame 4. All sequence gaps are ignored. To offset or retime a clip, use the Timing subtree in the *FileIn* parameters or the Time View tab.

When reading in an image, the File Browser allows you to specify if the first sequence image (suppose the sequence starts at frame 20) is placed at frame 1, the start frame (for example, 20), or the current frame.

Referring to Media Using File Paths

Shake can read local or absolute file paths. For example, with a machine named "myMachine," suppose you had a directory structure like the following:

```
/shots/my_directory/my_image.iff
/shots/scr/my_script.shk
```

The script can access *my_image.iff* in the following ways:

```
FileIn1 = FileIn("../my_directory/my_image.iff");
FileIn2 = FileIn("/shots/my_directory/my_image.iff");
FileIn3 = FileIn("//myMachine/shots/my_directory/my_image.iff");
```

Note: Local file paths in a script are local to where the script is located, not from where Shake is started.

When Shake reads in an image, it converts the file path of the image to the UNC naming convention. This labels the machine name first, and then the file path. The third listing above is an example of this convention. This behavior can be turned off in a preferences file. For more information, see "Customizing File Path and Browser Controls" on page 371.

Shake looks for its files in the User directory ($HOME) when launched from the application icon, or the current directory if launched from the Terminal. This affects how manually entered paths for *FileIns* are read.

- *If the image paths are local* (for example, "ImagesDirectory/image.#.iff"), images are read relative to where the script is saved.
- *If paths are global* (for example, "//MyMachine/MyBigHardDisk/ImagesDirectory/image.#.iff"), then images have no relation to where the script is saved, and thus the script may be easily moved into different directories.

  If the script and the image are on different disks, you must specify the proper disk—local file paths do not work.
- *For a URL address*, place a *//* in front of the path. To read from another computer, write *//myMachine/Drive/Directory/etc*.

Using the FileIn (SFileIn) Node

The *FileIn* node is used to read in media (from disk for processing by your script.

Note: Although labeled "FileIn" in the Tool tab, this node actually represents the more advanced *SFileIn* function. The *SFileIn* node includes internal features not available in the older *FileIn* node used in previous versions of Shake. The enhanced functions include *FIBlend*, *FINearest*, *FIPullUpDown*, and *IRetime*. These functions are "internal" because they do not appear in the Shake interface, but are saved inside of each script. For the purposes of this manual, unless otherwise stated, "FileIn" refers to the enhanced "SFileIn" functionality.

The *FileIn* and *SFileIn* functions include:

- *FileIn*: A pre-v2.5 function, convenient for scripting, given its brevity, it can only be shifted in time with *IRetime*.
- *SFileIn:* From v2.5 and later, this node can be hooked up to have preset proxies, shifted with *IRetime*, or modified by *FIBlend*, *FINearest*, or *FIPullUpDown*.
- *FIBlend*: This invisible function does non-linear retiming of a sequence, blending frames together. It modifies *SFileIn* only.
- *FINearest*: This invisible function does non-linear retiming of a sequence with no frame blending. It modifies *SFileIn* only.
- *FIPullUpDown*: Does pullup or pulldown operations on an *SFileIn* node.
- *FISpeed*: Similar to *Blend*, except instead of a curve, you control speed with a slider, for example, 2x, .5 speed, and so on.

- *IRetime*: Sets the start/stop frame of a clip, can slip sync, and controls how the clip behaves for frames outside of the frame range. Works for *FileIn* and *SFileIn*.

FileIn Source Parameters

The parameters for each *FileIn* node are divided into subtabs: the Source tab and the Timing tab.

The Source tab, located on the left side of the Parameters tab, contains the following controls:

ImageName
Specifies the local or absolute path to the image or media file on disk.

Note: You can get away with supplying only the imageName and omitting the other path information within a script.

The ImageName subtree contains the following subparameters:

- *BaseFile:* Specifies the original, high-resolution image sequence or movie file associated with this node. Opening this subtree reveals one additional parameter:
  - *baseFileType:* A subparameter of BaseFile that tells Shake what the format is if the file suffix is ambiguous. Generally, you do not need to set this, as "Auto" automatically detects the format. If you have a problem reading an image, try setting the format to correct this.
- *ProxyXFile:* Additional parameters appear if you have created proxies to use in this script. For more information on using proxies, see Chapter 4, "Using Proxies," on page 137.

firstFrame
Lets you trim the beginning of a clip.

lastFrame
Lets you trim the end of a clip.

Note: QuickTime clips do not display either the firstFrame or lastFrame parameters.

increment
This parameter controls how frames in the referenced image sequence are advanced, providing an unsophisticated method for retiming the clip by either skipping or multiplying frames being read in from an image sequence.

- The default value of 1 means that every frame plays back, and the clip's duration in the script is identical to its duration on disk.
- A value of 2 or higher means that frames are skipped. At 2, every other frame is skipped, and the clip duration is halved.

Note: QuickTime clips do not display this parameter.

autoAlpha
If this parameter is on (1), then Shake creates a solid alpha channel for images not containing an alpha channel. If the image already has an alpha channel, this parameter is ignored.

deInterlacing
This parameter is to be used when importing interlaced images that you intend to render with fieldRendering on. When deInterlacing is on, Shake takes the odd or even field of the image (counted from the top) and copies it over the other remaining field. It then does the same thing half a frame forward. You are therefore left with two images the same height as your input image, but squeezed into the same time frame. For more information, see Chapter 5, "Compatible File Formats and Image Resolutions," on page 167.

force8Bit
This parameter appears for *FileIn* nodes reading in QuickTime media. When force8Bit is turned on, 10- and 16-bit media is downsampled to 8-bits per channel.

Paths

Both *FileIn* and *SFileIn* recognize local, absolute, variables, or URL paths:

- Absolute Path: */usr/people/myDirectory/myimage.iff*
- Local Path: *../myimage.iff*
- Environment Variables: *$myimages/myimage.iff*
- URL Address: *//wherever/usr/people/myDirectory/myimage.iff*

Note: For more information on variables, see "Environment Variables for Shake" on page 393.

Missing Frames in an Image Sequence

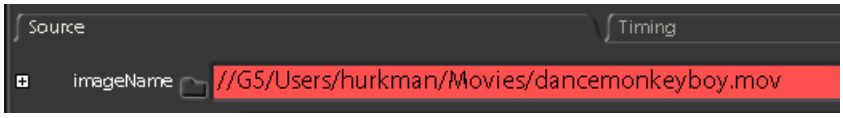
If one or more frames from an image sequence is missing, Shake handles this in one of two ways, depending on the file name format specified in the imageName parameter of the *FileIn* node.

- If the file name format is *filename.1-30#.tiff*, Shake expects an uninterrupted sequence of frames to exist on disk. If individual frames are accidentally deleted or moved from the specified path, each missing frame results in a gap in the image sequence in Shake. Each gap results in a black frame being displayed in the Viewer. For example, if frame 17 goes missing after a tiff sequence has been imported, moving the playhead to frame 17 in the Time Bar displays a black frame in the Viewer.
- If frame 17 is already missing on disk when you first select the image sequence, the File Browser will show a segmented frame range, and the resulting imported image sequence will appear as a continuous, unbroken frame sequence. For example, if frame 17 is missing in a selected tiff sequence, its name appears as *filename.1-16,18-30#.tiff*. Scrubbing to frame 17 in the Time Bar displays frame 18, instead, because the sequence name lets Shake know to skip that frame and close the gap.

Unlinked Files

If, for any reason, a *FileIn* node cannot find any of the media it was originally linked to, that node becomes unlinked. This happens with both image sequences and QuickTime files. Unlinked *FileIn* nodes are red in the Node Viewer.

In the Source tab of the Parameters tab, the imageName parameter field of an unlinked *FileIn* node also turns red. The original path name still appears.



FileIn nodes can become unlinked if the media they originally referenced has been moved to another directory or volume, renamed, or deleted. *FileIn* nodes can also become unlinked if the Shake script has been moved to another machine. In any case, *FileIn* nodes can be easily relinked to the original source media at any time.

To relink a *FileIn* node to the original files:

1 Load the *FileIn* node's parameters into the Parameters tab by double-clicking the node, or clicking on the right side of the node once.

2 Click the File Browser icon in the ImageName parameter.

3 Use the File Browser to find the original media files, then click OK.

Note: The File Name field displays the name of the file that was originally linked to that *FileIn* node.

FileIn Timing Parameters

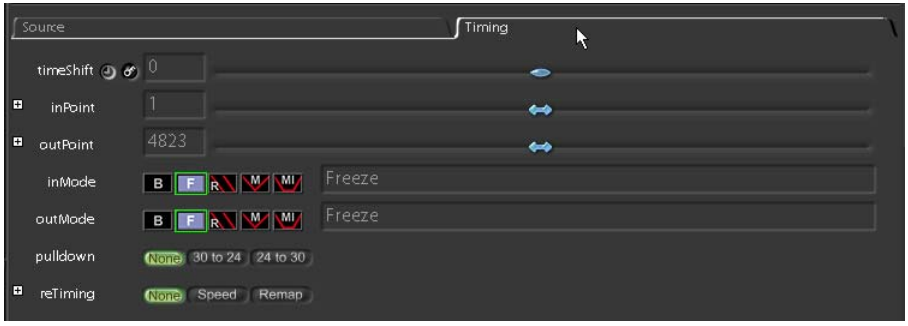
The second subtab under the *FileIn* Parameters tab is the Timing tab. The parameters in this tab control the timing of image sequences and movie files used in your Shake script. Many of these timing parameters, including timeShift, inPoint, and outPoint, can also be manipulated directly in the Time View.

The Timing tab also has options for retiming images, creating fast, slow, or variable-speed motion effects. These are covered in more detail in "Retiming" on page 117.

To access the timing parameters of a *FileIn* node:

1 Load the selected *FileIn* node's parameters into the Parameters tab by double-clicking it, or clicking the icon on the right side of the node.

2 Click the Parameters1 tab, and then click the Timing tab to reveal the timing parameters.



Parameters in the Timing Tab

The Timing tab has the following parameters:

timeShift
Slides the entire duration of the image sequence or movie forward or backward in time.

inPoint
Lets you extend the duration of a clip at its beginning, to loop or freeze the sequence.

outPoint
Lets you extend the duration of a clip at its end, to loop or freeze the sequence.

inMode
If media has been time-shifted or the In point changes so that there are blank frames prior to the first frame of the media in the Time View, this parameter lets you set how those empty frames should be filled.

outMode
This parameter lets you set how the empty frames after the last frame of the media in this *FileIn* node are filled.

Both the inMode and outMode parameters have the following options:

Icon	Name	Notes	Example (assumes a 5-frame sequence)
	Black	All frames before or after the image frames are black.	1, 2, 3, 4, 5, black, black, black...
	Freeze	The first and last frames are repeated before and after the clip.	1, 2, 3, 4, 5, 5, 5, 5...
	Repeat	The sequence is repeated, looping the image sequence from the first frame.	1, 2, 3, 4, 5, 1, 2, 3...
	Mirror	The sequence is repeated, looping the image sequence by flipping the order each time. The first and last frames are not doubled up.	1, 2, 3, 4, 5, 4, 3, 2...
	InclusiveMirror	The sequence is repeated, looping the image sequence by flipping the order each time. The first and last frames are shown twice in a row.	1, 2, 3, 4, 5, 5, 4, 3...

pulldown
Lets you introduce or remove 3:2 pulldown from the media referenced by this *FileIn* node.

reTiming
The reTiming parameter provides options for changing the speed of clips. By default, this is set to none, and the media remains at its original speed. For more information on retiming, see "Retiming" on page 117.

For more information on shifting clips, see "Adjusting Image Nodes in the Time View" on page 263.

Pulldown and Pullup

3:2 Pulldown is a technique to temporally convert the framerate of noninterlaced film footage to that of video, and back again. The pulldown parameter in the Timing tab of *SFileIn* allows you to manage your pulldown/pullup of a sequence. There are two options:

30 to 24

This option removes pulldown from a media file that has been telecined to 30 fps. Use this setting to return it to 24 fps for compositing in Shake.

24 to 30

This option converts 24 fps film footage to 30 fps—adding 3:2 pulldown.

dominance

When you select either option, an additional dominance parameter appears which lets you select the field dominance of the output.

More About 3:2 Pulldown

Film uses frames, running at 24 frames per second (fps). Video uses interlaced fields, with the frame rate of NTSC video running at 29.97 fps, and the frame rate of PAL video running at 25 fps. To convert a film sequence into a video sequence, you need to split the film frames into fields, and double up two out of every five frames in order to make 24 film frames fill the space of 30 video frames per second. To use the classic graph:

1/6 of a Second Equals					
4 Film Frames	A	B	C	D	
5 Video Frames	AA	BB	BC	CD	DD

The third and fourth frames have fields that blend to stretch time. It's called 3:2 because you have three solid frames and two mixed frames.

To fully reconstruct the original four film frames (in *time*, not *resolution*—the original resolution is already lost), you must extract the field data from the five video frames.

But there is usually a complication—when you receive your footage, it has probably already been edited. As a result, there is no guarantee that frames 3 and 4 are the mixed frames because all of the clips have been shifted in the edit. As a result, you need to determine what the first frame is.

To determine the first frame of an image sequence with 3:2 pulldown:

1. Double-click the *FileIn* node to load its image in the Viewer and its parameters into the Parameters1 tab.
2. In the Time Bar, move the playhead to the first frame in the sequence, and scrub through the first five frames while looking at the Viewer to determine the first frame that shows two fields from different frames that are blended together.

3 Choose the firstFrame value that corresponds to this frame number in the following chart:

First Frame With Field Blending	firstFrame Setting
1	BC
2	BB
3	AA
4	DD
5	CD

4 Click the Timing tab in the Parameters1 tab, and set the firstFrame parameter according to the table in step 3.

Note: If the first several frames in a shot are a solid color and you are unable to determine the first mixed frame, jump forward to a time range of frames that displays the blending, and start guessing what firstFrame is until the fields disappear.

Reintroducing 3:2 Pulldown After Transforming Media

Removing pulldown from an image prior to transforming it within a Shake script is simple. If you need to reintroduce it after the transformation, however, this must be performed in a second step with another script.

After removing the original 3:2 pulldown and performing all necessary transformations and other effects in the initial script, render out the result as a 24 fps media file. Afterwards, read the resulting media file into a second script using a *FileIn* node, adding 3:2 pulldown back to the shot by clicking the *24 to 30* option in the pulldown parameter of the *FileIn* Timing tab.

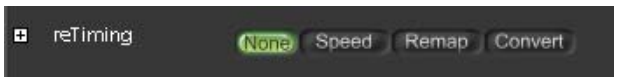
You can also simply process the output via a command-line render, using the following commands:

- `-pulldown <image> <field> [offset]`
- `-pullup <image> <field> [offset]`

For more information on processing media from the command line, see Chapter 3, "Adding Media, Retiming, and Remastering," on page 107.

Retiming

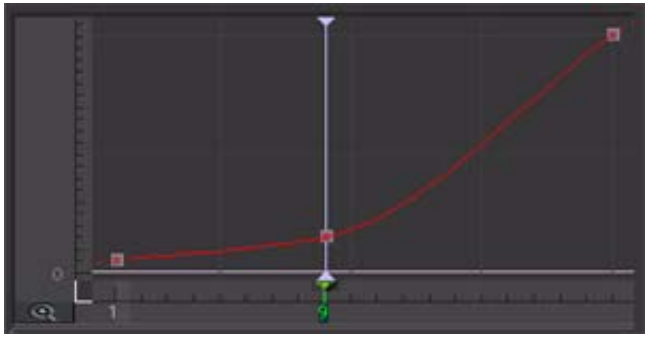
You can also squeeze, stretch, or nonlinearly retime your clip when you activate reTiming in *SFileIn* parameters. The reTiming Parameter can be set to Speed or Remap.



The reTiming parameter has four options:

- *None:* No retiming is applied, and the clip plays at its original speed.
- *Speed:* Lets you change the speed of a clip using a simple multiplier. For example, .5 returns a clip twice the length of the original, that plays in slow motion.
- *Remap:* Allows you to change the speed of a clip with a curve, with the X axis representing the input frame number and Y representing the frame it's remapped to.
- *Convert:* The Convert option of the reTiming parameter provides an advanced processing method for converting media to other formats. For more information on using the Convert option for format conversion, see "Remastering Media" on page 126.

The Remap option creates an ease-in effect, slowing the first part of a clip and accelerating it as the clip near the end.



Both the Speed and Remap options can smooth the effect of slow-motion strobing by blending frames using either the Blend or Nearest option in the retimeMode pop-up menu. Blend averages frames together and Nearest takes the frame right below the specified frame. For example, if frame 5.7 is needed, frame 5 is used.

The Convert option provides a high-quality method of processing speed changes.

Speed Parameters

If you select the Speed option, the following parameters appear:

Speed

The speed of the clip is multiplied by this value. For example, a value of .5 slows the clip to 50 percent. A value of 2 speeds up the clip by 200 percent.

retimeMode
By default, you are given three options for frame blending:

Nearest
No frame blending is applied, and Shake simply uses a copy of the nearest available frame to fill the new in-between frames.

Blend
Averages neighboring frames together to create in-between frames that are a combination of both to soften the strobing effect that can result from slow motion effects. If the retimeMode parameter has been set to Blend, three additional parameters appear underneath.

- *retimeBytes:* Affects frame blending. When multiple frames are blended together, setting retimeBytes to a higher bit depth will result in a more accurately calculated image by doing the math at a higher bit depth. For example, if you have a blending mode that is blending together many frames, and two of them happen to be at 50%, in 8-bit, you get .504 or .496. In a 16-bit calculation, you can get much closer to exactly .5.
- *weight:* A gamma curve is applied to source frames that are blended together in order to create the in-between frame. A weight of 0 means each source frame needed to create the in-between frame contributes equally, while a higher gain, such as 2, causes the center frame to give the greatest contribution and frames farther away proportionately less.
- *range:* Controls how many frames are blended together to create the final result. For example, if you want to extend a source clip of 20 frames to 40 frames, each source frame is applied to two output frames. With a range of 2, it is applied in four output frames, resulting in more blending. If you apply only this value with no other modifications, Shake inserts repetitions of neighboring frames to help you with degraining.

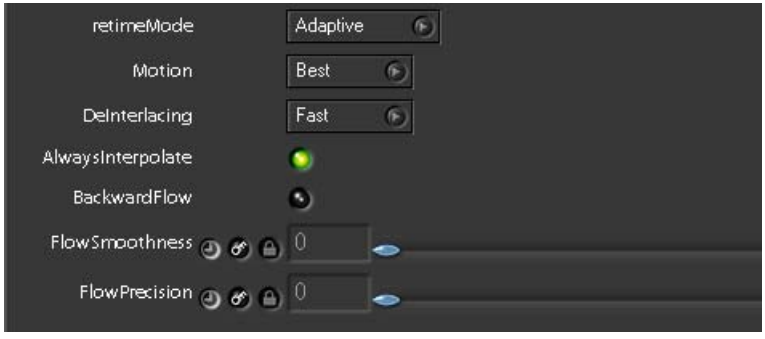
  Note: The *FileIn* node has been written so that it's possible to create a custom frame-blending algorithm, if you happen to have a spare developer.

Adaptive
This option in the retimeMode pop-up menu uses advanced image analysis to generate new in-between frames, creating seamless slow and fast-motion effects. Selecting Adaptive reveals additional parameters.

Fast versus Best Settings for Adaptive Retiming

When setting up an adaptive timing operation, you might be tempted to simply choose Best across the board for every parameter. This would probably be a mistake—producing dramatically longer render times in exchange for a potentially undetectable increase in quality, especially at higher resolutions.

That said, every clip's requirements are different. One group of settings is unlikely to produce equal results for shots with widely different subjects and movement. You're encouraged to do some limited tests prior to committing yourself to a particular group of retiming settings. As you experiment with different settings, be sure you always compare output from the Fast settings to that from the Better or Best settings, to make sure that it's worth committing yourself to the more computationally intensive Best settings.



- *Motion:* Two options determine the trade-off between image quality and processing time—Fast and Best. Fast makes motion interpolations using a mesh warp, and works well in most situations. Best, the most processor-intensive mode, uses motion vectors to track each pixel from frame to frame, interpolating new frames.
- *DeInterlacing:* Similar to the Motion parameter, two options let you determine the trade-off between image quality and processing time—Fast and Good. These settings represent two levels of mesh warping used to interpolate data from both fields of each frame. Here are some tips for using the DeInterlacing parameter:
  - Good is actually a better setting for frames without any moving subjects.
  - If you're working with standard definition video, there is no difference between using Fast or Good.
  - The DeInterlacing operation can also be used to improve clips that were poorly deinterlaced in other applications prior to importing into Shake.

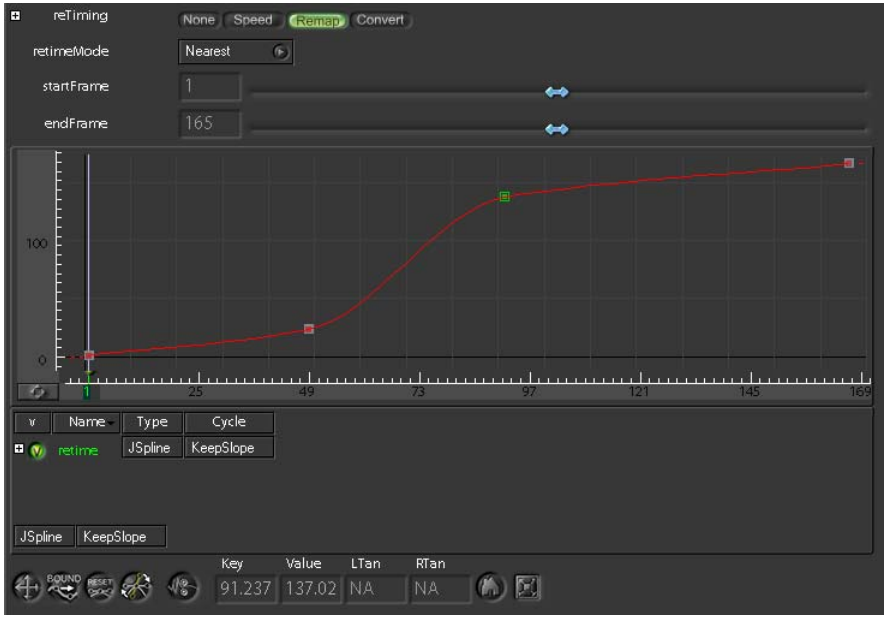
- *AlwaysInterpolate:* With AlwaysInterpolate turned off, the final result of a retiming operation is a mix of original, unprocessed frames, and interpolated frames. For example, setting the speed to 0.5 to slow an image sequence down by 50 percent results in an alternating series of original frames and interpolated frames. In cases where there is a significant visual difference (softening, for example) between the unprocessed and reprocessed frames resulting from a retiming operation, the image sequence may appear to flicker. Turning on AlwaysInterpolate forces Shake to process every single frame in a retimed sequence, resulting in a series of frames with consistent visual quality.

If the Motion parameter is set to Best, three additional parameters become available:

- *BackwardFlow:* Turning on BackwardFlow evaluates the flow of images in both directions in order to generate interpolated in-between frames. This mode is usually visually superior, but significantly slower.
- *FlowSmoothness:* Higher or lower values may improve the quality of interpolated frames, depending on the shape of the subject in the image.
  - Use *low* values to improve the quality of subjects in the frame that change shape, like a person or animal that's running or jumping.
  - Use *high* values to improve the quality of static objects that don't change shape, such as trees, buildings, or cars.
- *FlowPrecision:* This is the last parameter you should adjust, after obtaining as much quality as possible with all of the above settings. Increasing the value of this parameter increases the overall precision of the Adaptive retiming operation by increasing the resolution at which the optical flow is estimated. A value of 0 is fine for most situations.

Remap Parameters

If you select the Remap button in the reTiming parameter, the following additional parameters appear:



- *retimeMode:* By default, you are given two options for frame blending:
  - *Nearest:* No frame blending is applied, and Shake simply uses a copy of the nearest available frame to fill the new in-between frames.
  - *Blend:* Averages neighboring frames together to create in-between frames that are a combination of both to soften the strobing effect that can result from slow motion effects.
  - *Adaptive:* An interpolation method for retiming that uses advanced image processing to generate new in-between frames. This is the highest-quality method for many images, and is the most processor-intensive. With the retimeMode parameter set to Adaptive, two additional parameters appear to let you adjust the trade-off between quality and processing time. For more information, see the section on Adaptive parameters on page 120.

  If the retimeMode parameter has been set to Blend, two additional parameters appear underneath.

  - *weight:* A gamma curve is applied to the mixture of source frames that are blended together in order to create the in-between frame. A weight of 0 means that each source frame needed to create the destination contributes equally, while a higher gain, such as 2, causes the center frame to give the greatest contribution and frames farther away proportionately less.

- *range:* Controls how many frames should be blended together to create the final result. For example, if you want to extend a source clip of 20 frames to 40 frames, each source frame contributes to two output frames. With a range of 2, each source frame contributes to four output frames, resulting in more blending. If you only apply this value with no other modifications, you get repetitions of neighboring frames to help you with degraining.

  Note: The *FileIn* node has been written so that it's possible to create a custom frame-blending algorithm, if you happen to have a spare developer sitting around.

- *retimeBytes:* This parameter affects frame blending. When multiple frames are blended together, setting retimeBytes to a higher bit depth will result in a more accurately calculated image by doing the math at a higher bit depth. For example, if you have a blending mode that is blending together many frames, and two of them happen to be at 50 percent, in 8-bit, you get .504 or .496. In a 16-bit calculation, you can get much closer to exactly .5.
- *startFrame, endFrame:* Specifies the frame range used for retiming calculations.
- *Curve Graph:* The *retime* parameter appears in a graph within the Parameters tab. The X axis represents the input frame number, and the Y axis represents the frame it is remapped to.

Understanding the Retiming Parameters

If you are having difficulty understanding the multitude of Retiming parameters, copy this string, paste it into the Node View, and then render out 100 frames.

```
Text1 = Text(300, 100, 1, "%f", "Courier", 44.3, xFontScale, 1,
     Hermite(0,[10,76.08,76.08]@1,[261,76.08,76.08]@100), Hermite(0,[90,-
     34.69,
     -34.69]@1,[30,-34.69,-34.69]@100), 0, 2, 2, 1, 1, 1, 1, 0, 0, 0, 45,
     0, 1);
```

Then, read in the rendered clip and test the retiming.

The TimeX Node

You can also retime a clip using the *TimeX* node, located in the Other tab. This node lets you use mathematical rules to change timing on the input clip.

By default, the value of the newTime parameter is the expression *time*—which is the frame at the current location of the playhead in the Time Bar. Using the *time* expression makes no change. Typically, you'll modify this expression in order to remap frames from their original position in the clip, to create new timings.

Parameters

The *TimeX* node has one parameter in the Parameters tab:

newTime

This parameter defaults to a spline that maps every input frame to a corresponding frame in time, such that the clip plays forward normally at 100 percent speed. Typically, you'll enter a new expression into this field, using the expression *time*, to remap the frames of the image sequence or movie file to create new timing effects.

Similar to *Lookup* and *ColorX*, you can duplicate most other Time functions with *TimeX*. These other functions are simply macros that include *TimeX*. They are included because *TimeX* can be counter-intuitive.

Timing Expression	Explanation
time-10	Shifts the clip 10 frames forward. While processing frame 50, it reads input frame 40.
101-time	Assumes frame 100 is the last frame. At frame 1, 100 used.
time%10+1	Loops every 10 frames. Takes the remainder of time/10, and adds 1 (otherwise frame 10 = 0).
time>10?10:time	A conditional expression. Freezes the clip at frame 10. Any frame before that is processed normally.
time	Do nothing.
100 (or any integer)	Picks one frame. In this example, at every frame, the node returns 100, so only input frame 100 is used.
time*2	Double the rate. In this expression, at frame 10, frame 20 is used.
"CSpline(0, 1@1, 30@25, 40@50, 90@75, 100@100)"	Using a curve to speed the clip up and down.The arbitrary curve shown here returns different frame values. You can use any spline type, with as many keyframes as you want.

A more complex example is to animate 360 3D frames with an animated light. The light is from the right at frame 0, from the top at frame 90, from the left at frame 180, and so on. You then position a fake light source in the composite. By figuring out the angle of the light to the 3D element (using trigonometry), you can pick one of the 360 input frames to fake the lighting change.

Multiple Branches

You can only have one branch traced up to the *FileIn* with a *TimeX* in it. To get multiple time shifts on the same clip, copy the *FileIn*. Note also that *FileIn* has timing controls of its own that you may find easier to use.

Manual Manipulation of Time

This section explains the notation Shake uses for a *FileIn* node, and the available *FileIn* options. It also discusses the notation for the timeRange parameter in the Globals tab, or the *-t* option on the command line. For a discussion of the interactive controls of time, see Chapter 8, "Using the Time View," on page 261, and "Using the FileIn (SFileIn) Node" on page 110.

Time Notation for a FileIn

This section focuses on manual manipulation of time. For most interactive time manipulation, Shake relies on the Time View and its associated timing subtree in the *FileIn* parameters. You can manipulate time in other ways, specifically on the command line.

When Shake reads in a clip, it inserts the start and end frame of the clip in the clip name, and gives an indication of the padding style, denoted here with the number sign #:

image.1-50#.iff

In the above example, this notation indicates that only frames 1 through 50 are loaded, even though there may be more files. The other frames are black when read in with the default settings.

Shake puts the start of the range at frame 1. If you have:

image.20-50#.iff

at frame 1, *image.0020.iff* is read.

You can also shift the clip to frame 20 in the Time View of the interface.

Shake can recognize a series of frames when reading in a file without using the clip range. When looking at a sequential series of files, use a placeholder in the file name to represent the frame number. This placeholder is typically either a # sign (padded images, *image.0001.iff*, *image.0002.iff*, and so on.) or an @ sign (unpadded images, *image.1.iff*, *image.2.iff*, and so on). If your numbers are padded to a number other than 4, you can substitute multiple @ signs.

You can also use the %d placeholder, which specifies the number of decimal spaces applied to padded images. For example, *image.%03d.iff* produces padding of three decimal places—*image.001.iff, image.002.iff,* and so on).

The following are some examples of frame number placeholders:

Shake Format	Reads/Writes
image.#.iff	*image.0001.iff, image.0002.iff*
image.%04d.iff	*image.0001.iff, image.0002.iff*
image.@.iff	*image.1.iff, image.2.iff*

Shake Format	Reads/Writes
image.%d.iff	*image.1.iff, image.2.iff*
image.@@@.iff	*image.001.iff, image.002.iff*
image.%03d.iff	*image.001.iff, image.002.iff*

Time Notation Setting the Script Range

The script range can be set in the timeRange field of the Globals tab, or on the batch command line with the *-t* option, which overrides the script.

The range description is extremely flexible. The following are some examples:

Time Range	Number of Frames	Frames Rendered
1-100	100	1, 2, 3... 100
1-100x2	50	1, 3, 5... 99
1-100x20	5	1, 21, 41... 81
1-20, 30-40	31	1, 2, 3... 20, and 30, 31, 32... 40
1-10x2, 15, 18, 20-25	13	1, 3, 5... 9, 15, 18, 20, 21, 22 ... 25
100-1	100	100, 99, 98... 2

To set time range in the command line when rendering a script, use the *-t* option:

```
shake -exec my_script.shk -t 50-60 -v
```

For command line examples of time manipulation, see "Frequently Used Functions" on page 1023 of Appendix B, "The Shake Command-Line Manual."

For more information on using the Time View, see Chapter 8, "Using the Time View."

Remastering Media

The Convert option of the reTiming parameter provides a method for converting media from one format to another using advanced image processing to rescale and retime the incoming media. For example, if you have a high definition image sequence that you want to convert into a standard definition image sequence, or a PAL clip that you need to change to NTSC, the Convert option provides the tools to do so.

Choosing Convert reveals a series of parameters within the *FileIn* node that allow you to change the frame rate, resize the output resolution, anti-alias and sharpen the resulting image, and deinterlace the media being referenced by that *FileIn*. These options provide the highest-quality means of resizing and deinterlacing available in Shake, with results that are superior to the transform nodes that are available from the Tool tabs. These options are only available within the *FileIn* node.

You can use these options to convert individual shots that you're compositing within Shake, or you can read in an edited sequence from an application like Final Cut Pro for format conversion using Shake.

Important: If you're converting a clip from a video frame rate to that of film with the intention of adding 3:2 pulldown back to the video (to achieve a film look for video), render the 24 fps conversion first. Add 3:2 pulldown to the shot in another operation by processing it in a second script, or by adding 3:2 pulldown with a command-line render. For more information, see "Reintroducing 3:2 Pulldown After Transforming Media" on page 117.

Fast versus Best In Remastering Parameters

When setting up a remastering operation, you might be tempted to simply choose Best across the board for every parameter. This would probably be a mistake—producing dramatically longer render times in exchange for a visually undetectable increase in quality, especially at higher resolutions.

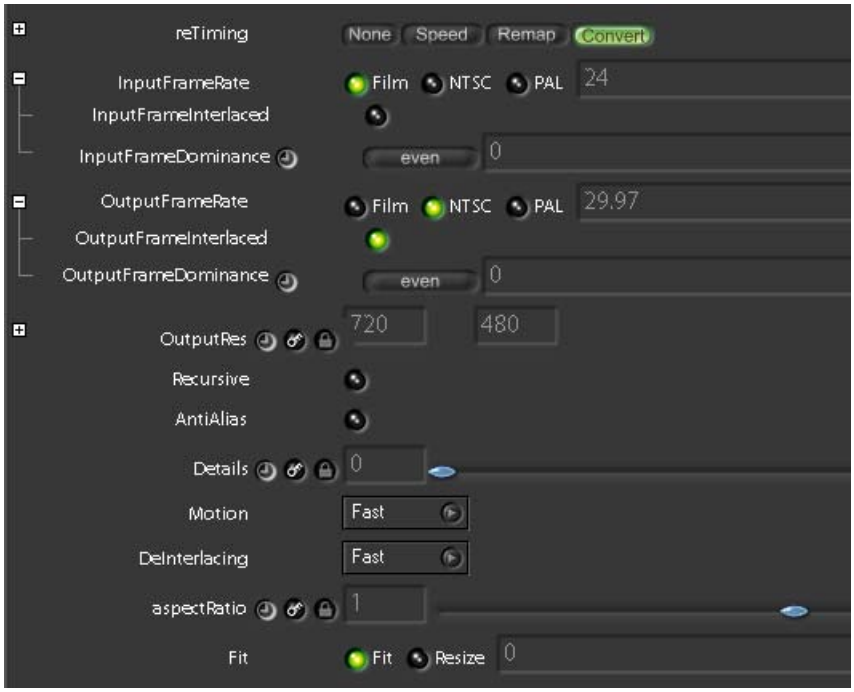
That said, every clip's requirements are different. One group of settings is unlikely to produce equal results for shots with widely different exposures, grain, and camera movement. You're encouraged to do some limited tests prior to committing yourself to a particular group of mastering settings. As you experiment with different settings, be sure you always compare the output from the Fast settings to that of the Better or Best settings, to make sure that it's worth committing yourself to the more computationally intensive Best settings.

Automatic Scene Detection for Multiple Shots

If you're reading in a sequence of pre-assembled shots, the remastering operators in Shake use automatic scene detection to eliminate artifacts at the frame boundaries between shots. This edge detection works well for cuts and dissolves. but other types of transitions may produce unwanted artifacts.

Convert Parameters

The Convert mode has the following parameters:



InputFrameRate

Specify the original frame rate of the input media here. This parameter is also a subtree with two additional subparameters.

InputFrameInterlaced

If the input media is video, enable this parameter if it's interlaced.

InputFrameDominance

If the input media is interlaced, specify the field dominance here.

OutputFrameRate

Specify the output frame rate here for format conversion. This parameter is a subtree with two additional subparameters. For advanced retiming options to produce slow and fast motion, see "Retiming" on page 117.

OutputFrameInterlaced

If you want interlaced video output, turn this parameter on. Leaving it off results in Shake outputting progressive-scan (non-interlaced) media.

OutputFrameDominance
If OutputFrameInterlaced is turned on, specify the field dominance of the output image here.

OutputRes
Two fields where you enter the horizontal and vertical output resolution you want to convert the media to, to scale it up or down. Scaling an image sequence using the OutputRes parameter of the Convert options results in higher-quality output than using Shake's *Transform* nodes.

Recursive
Turning this parameter on enables a different resizing method, which can be sharper when enlarging some kinds of images. Try it on one or more representative frames to see if it helps.

Note: The recursive setting may also enhance unwanted noise, depending on the image.

AntiAlias
Turning this parameter on improves the quality of conversions when you're scaling media up. For example, when converting standard definition video to high definition, turning on AntiAlias smooths out jagged edges that might appear in the image.

Details
A built-in sharpening control that lets you add detail back to an image being enlarged. Unlike other sharpening operations, the details setting is able to distinguish between noise and feature details, and generally doesn't increase unwanted grain. Increasing this parameter may introduce jagged edges, however, which can be eliminated by turning on the AntiAlias parameter.

Motion
Two options determine the trade-off between image quality and processing time. Fast makes motion interpolations using a mesh warp, and is generally the only setting necessary for purposes of remastering.

Note: Best, the most processor-intensive mode, uses motion vectors to track each pixel from frame to frame to interpolate new frames, and should only be necessary when doing retiming. For more information on the parameters available for the Best setting, see page 120.

DeInterlacing
Similarly to the Motion parameter, two options let you determine the trade-off between image quality and processing time—Fast and Good. These settings represent two levels of mesh warping used to interpolate data from both fields of each frame.

AspectRatio
This parameter is a multiplier that allows you to convert pixels of one aspect ratio into another aspect ratio—for example, from NTSC to PAL, or from high definition (square) to NTSC. The default value of 1 makes no change to the output. The following table contains common conversion values:

Operation	Conversion Value
Square to NTSC (4:3)	0.9140, or 1 if Fit is set to Resize
Square to PAL (4:3)	1.1111, or 1 if Fit is set to Resize
NTSC (4:3) to Square (4:3)	1.1111, or 1 if Fit is set to Resize
NTSC (4:3) to PAL (4:3)	1, set Fit to Resize
PAL (4:3) to Square	.9375, or 1 if Fit is set to Resize
PAL (4:3) to NTSC (4:3)	1, set Fit to Resize

Note: In some conversion cases, AspectRatio may be left at 1 if the Fit parameter is set to Resize, which rescales the image horizontally and vertically to match the new OutputRes.

Fit
The Fit parameter determines how an image fits into the new frame that's determined by the OutputRes parameter if the aspect ratio of the *FileIn* image is different than that of the final resolution set by the OutputRes parameter. There are two options:

- *Fit:* Enlarges the image until either the vertical or horizontal resolution of the image matches that of the outputRes, depending on which is greater. This option maintains the original aspect ratio of the image, enlarging it as much as possible and leaving black to fill in the resulting gaps at the sides, or the top and bottom.
- *Resize:* Rescales the vertical and horizontal resolution of the image so that it fits within the entire frame of the OutputRes. The image is stretched as necessary to fit the new resolution.

Working With Extremely High-Resolution Images

These guidelines are specifically for high-resolution images of 4K and 6K. The following discussion is based on the premise that you have a massive amount of RAM for your interactive workstation (at least 1 GB).

Although Shake works with any resolution, there is a default crop on Viewers in the interface of 4096 x 4096 pixels. This protects the user in case a zoom of 1000 is applied to a 2K plate. Instead of trying to render an enormous image, only the lower-left corner up to 4096 pixels is rendered in the interface. This is fine for normal HD or film production, but the cropping takes effect if you read in 6K IMAX plates. This limitation is only in the interface—images rendered to disk are at the uncropped full resolution.

There are two ways you can get around this safety feature.

Using Proxies

The first is to use proxies with a proxyScale of less than 1. For example, at a proxyScale of .5, you can potentially look at images up to 8K x 8K resolution.

Changing the Viewer Limits

The other workaround is to change the default Viewer limits by customizing a *ui* preference file. Add the following lines:

```
gui.viewer.maxWidth = 4096;
gui.viewer.maxHeight = 4096;
```

These lines set the maximum resolution to 4K. If you want a larger resolution, enter it here.

For more information on creating and editing these preference files, see Chapter 14, "Customizing Shake."

Adjusting the Cache for High-Resolution Images

When working with high-resolution images, it's also necessary to adjust the cache settings. By default, only images under 2K resolution are cached. By not automatically caching large files, Shake conserves cache capacity, enabling you to add more files. You can override this default with the following two lines, which adjust the default values. The first line sets the maximum size by listing the X resolution, Y resolution, number of channels, and amount of bytes. The second line sets the maximum amount of disk space for the cache directory. You can assume that if you are working on 6K plates, you can allow for more than 512 MB of disk space for your cache. These lines go in your *startup* preference files. You modify the numbers to suit your production situation:

```
diskCache.cacheMaxFileSize = 2048*2048*4*2;
diskCache.cacheSize = 512;
```

Keep in mind that if you set your maximum file size to 6K x 6K x 4 channels in float, you are saving massive files. The return you have on swapping this in and out of cache is extremely limited, at best. It is recommended you use proxies when interactively working with 4K and 6K images.

If you need to work at full resolution, try putting a *Crop* at the end of the chain to focus on an area of interest, or using the Viewer DOD. This retains full pixel resolution, but keeps your image resolution within the framework of your computer.

Tuning the Amount of RAM Shake Uses

Finally, you need to tune the amount of RAM used by Shake. By default, 96 MB are assigned to the nodes and 64 MB to the images themselves. You need to increase the second setting. It is recommended that you allocate one-third of your memory to each of the two following settings, to reserve memory for other applications and Flipbooks. However, the first setting rarely needs to exceed 96 MB. For example, if you have 1 GB of RAM, you might want to have memory settings like the following:

```
cache.cacheMemory = 96;
diskCache.cacheMemory = 500;
```

The first line is associated with nodes, and does not affect image resolution. The second setting is associated with the images themselves, so you want to increase it as your images get larger. The default setting is 64 MB—not useful for large resolutions. These settings also go in your *startup* preference file.

For more information about resolution, see Chapter 5, "Compatible File Formats and Image Resolutions." For more information about caching, see Chapter 13, "Image Caching," on page 343.

Using Shake With Final Cut Pro

A new command in Final Cut Pro, *Send to Shake,* provides an automated way to move media back and forth between both applications. Using the Send to Shake command in Final Cut Pro exports one or more selected clips into a Shake script, opening it immediately in Shake while Final Cut Pro is running. When you do this, a placeholder is created in the originating Final Cut Pro project file that automatically corresponds to the media that will be output from Shake.

Note: Each exported clip from Final Cut Pro is brought into the Shake script using individual *FileIn* nodes. This is true even if two or more clips originate from the same master clip in the original Final Cut Pro project.

For example, you can use Final Cut Pro to superimpose a group of clips that you want to turn into a single composite using Shake. Final Cut Pro makes it easy to set the In and Out points of each clip, and how they overlap. You can then send the media to Shake along with each shot's edit decision information, freeing you from having to reconstruct the media arrangement within Shake.

You can also move an entire sequence of clips into a Shake script. For example, you might do this to add operations to each individual clip in that scene to perform color correction, or keying.

Once you're finished in Shake, you can render the *FileIn* node that was automatically created when you used the *Send* command from Final Cut Pro, and easily relink the resulting media in the original Final Cut Pro project.

How Sent Clips Are Arranged in Shake

Regardless of how you move Final Cut Pro clips into Shake, how they're assembled in the newly created Shake script depends on whether they were sequentially arranged within a single video track, or vertically superimposed using several video tracks. Imported Final Cut Pro clips are arranged within the node tree using *Select* and *MultiLayer* nodes:

- Clips edited sequentially on the same video track in Final Cut Pro are connected to a single *Select* node when exported to Shake. The *Select* node switches between clips at their In and Out points, reflecting the editing decisions made on the track in Final Cut Pro. If the clips were originally superimposed across multiple video tracks, each video track that contains a clip results in a corresponding *Select* node being created in the Shake script. All clips that were edited into the same video track are connected to the same *Select* node.

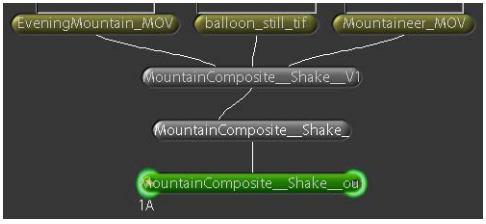
  Note: The actual edit points for each *FileIn* node attached to the *Select* node are stored within the branch parameter. The data stored within this parameter is *not* intended to be editable; any attempt to do so will disrupt the edit points of the affected nodes.

- All the *Select* nodes are connected to a single *MultiLayer* node, which determines which clips are in the foreground of the composition, and which are in the background. Their arrangement reflects the arrangement of video tracks in the original Final Cut Pro sequence.

For example, if you used the Send to Shake command on the following three sequentially edited clips:

Sequentially edited clips in Final Cut Pro

The result would be the following Shake script, with one *Select* node and one *MultiLayer* node.

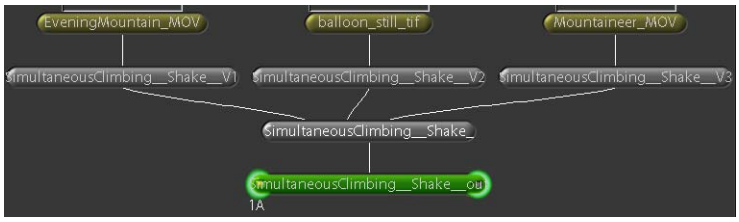


Resulting arrangement in Shake

If you used the Send to Shake command on the following superimposed clips:

Sequentially edited clips in Final Cut Pro

The result would be the following Shake script, with three *Select* nodes, and one *MultiLayer* node:



Resulting arrangement in Shake

While it is possible to slide footage within edits by adjusting the placement of imported clips in Shake's Time View, you are better off making these adjustments in Final Cut Pro and re-sending the media to Shake. Shake's Time View makes it difficult to determine whether there is sufficient underlying footage to prevent gaps in the sequence.

Warning: Audio clips and tracks from the original QuickTime files are not imported into Shake. Any timing changes you make in Shake will result in the adjusted clips going out of sync with the audio in the originating Final Cut Pro project file.

Unsupported Media and Effects

Since QuickTime is the file format used for all media exchange between Final Cut Pro and Shake, the following media and settings are not imported into Shake from Final Cut Pro:

- QuickTime audio tracks
- Standalone audio files
- Still image files
- Generators
- Composite modes
- Transformations (referred to in Final Cut Pro as motion effects)
- Filters

Sending Clips From Final Cut Pro

If you want to send one or more selected clips (or a single sequence), from Final Cut Pro to Shake, you should use the Send to Shake command in Final Cut Pro.

To send one or more clips from Final Cut Pro to Shake:

1 Clean up your project timeline, so that you are able to select only the clips you intend to send.

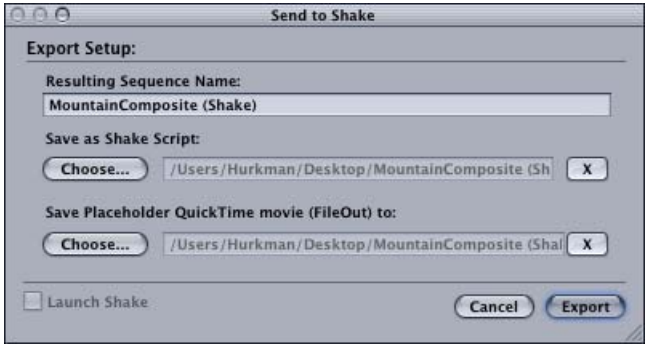
2 Do one of the following:

- Select one or more clips in the Timeline or Browser.
- Select a sequence in the Browser.

3 Do one of the following:

- Choose File > Send > Send to Shake.
- Control-click (or right-click) the selected clips or sequence, then choose Send > To Shake from the shortcut menu.

4 When the Send to Shake dialog appears, select the appropriate options:



- *Resulting Sequence Name:* Type a name for the sequence that you've selected, prior to sending the media to Shake.
- *Save as Shake Script:* Type a name for the Shake script to be created, then click Choose to pick a location on disk to save it to.
- *Save Placeholder QuickTime movie (FileOut) to:* Type a name for the placeholder QuickTime movie that will correspond to the *FileOut* node in the newly created Shake script, then click Choose to pick a location on disk to save it to.

5 Check the Launch Shake box if you want to automatically open the newly created Shake script and start working on it.

6 Click Export.

When you click Export, four things happen:

- A duplicate sequence appears in your Final Cut Pro project, containing duplicates of the selected media.
- A Shake project is created on disk.
- A placeholder QuickTime file is created on disk.
- The placeholder QuickTime file appears in a new video track that is created as the topmost track in your sequence (the original media remains where it was).

The placeholder QuickTime clip in your Final Cut Pro project corresponds to the media that will eventually be rendered out of Shake—specifically, from the *FileOut* node appearing at the end of the generated Shake script.

The TimeRange of Scripts Generated From Final Cut Pro

The timeRange Global parameter in the Shake script that's created by the Send to Shake command is automatically set with the appropriate range of frames for the media it references.

Important: Clicking the Auto button to update the timeRange is not recommended. This can result in many more frames being referenced than expected, depending on the total duration of the source media files that are referenced.

Sending Media Back to Final Cut Pro

When you're finished working in the Shake script that was generated from Final Cut Pro, all you have to do is render the originally created *FileOut* node. The newly rendered media file takes the place of the original placeholder QuickTime file, ready for use by the original Final Cut Pro project.

When you reopen the originating Final Cut Pro project file containing the original placeholder QuickTime file, you'll need to use the Reconnect Media command to relink the clip in your Timeline to the media that was rendered out of Shake.

Using Proxies

4

Shake has a sophisticated proxy system that lets you dynamically adjust the resolution of the images to speed your workflow. This chapter covers how to tailor Shake's proxy system to suit your needs.

Using Proxies

This section discusses how to use proxies to speed up your workflow. This includes using Shake's interactive scale setting, creating and assigning proxies to footage in your script, creating custom proxy settings, working with offline full-resolution elements, and pre-generating proxies.

What Are Proxies?

A proxy is a lower-resolution image that can be temporarily substituted for the high-resolution plates in your script, thereby enabling you to work and see rendered tests faster. Because the images are smaller, you drastically decrease the processing time, memory requirements, and amount of time spent reading and writing files as you work. Naturally, the trade-off is that the quality of the image displayed in the Viewer suffers, which is why proxies are generally used only when creating low-resolution comps, and creating test previews. After assembling a script using proxies, you can return your script to the original, full resolution in order to render the final output.

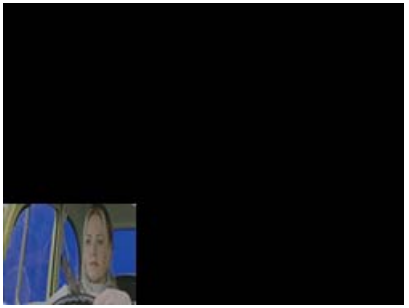
You can also use proxies to temporarily view anamorphic images in flattened space. For more information, see "About Aspect Ratios and Nonsquare Pixels" on page 209.

Proxies and Final Low-Resolution Output Renders

When you work with film plates and you need to generate a high-quality video output, you have better (but slower) results if you render your plates at full resolution and then size them down, instead of using the proxies to generate low-resolution images. The proxies can be used to generate lower-resolution output files, but the quality is not as high as with full-resolution rendering.

The following example shows a full-resolution image compared to a 1/3 scale proxy image. You can see that the proxy uses 1/9th of the space, which potentially requires 11 percent of the processing time, memory, and I/O activity.

Full resolution



1/3 proxy

Images from *The Saint* provided courtesy of Framestore CFC and Paramount British

As a result, there is a dramatic difference in quality when the clip is viewed at the same resolution, as you can see by the softening of the image to the right.

Full resolution

1/3 proxy, enlarged

Shake automatically adjusts the values of pixel-based parameters behind the scenes in order to compensate for the lower resolution of any proxies being used. In other words, when using a 1/3 proxy, a Pan parameter set to 100 pixels is actually calculated by Shake to be 33.333 pixels. The actual Pan parameter used in the interactive value field is not modified, and continues to reflect the actual size of the original media.

Shake's Three Proxy Methods

There are three basic approaches to using proxies that are controlled via parameters in the Globals tab.

Enabling a useProxy setting

If processing is slow overall, and you need to speed things up while you're working, you can enable one of the proxy settings without needing to pre-render a set of proxy files. This is a good option if you don't anticipate working on the project for very long.

Turning on interactiveScale

If the general processing speed for your operations is fine, but the interactivity of controls that correspond to processor-intensive operations is slowing you down, you can turn on the InteractiveScale option in the Globals tab. This sets Shake to use a proxy resolution only while you're adjusting parameters. This option does not affect your Flipbooks or *FileOut* renders.

Enabling useProxy and pre-rendering sets of proxy files

If you're working with very high-resolution footage, or you're using footage that's stored remotely on networked machines, you may find it best to pre-render a set of proxy files to your local machine. This is also a good option if the entire project is extremely processor-intensive, and you're doing a lot of Flipbook tests.

Important: If you decide to pre-render proxy files for your script within the Shake interface, make sure that you set the proxySet parameter in the useProxy subtree of the Globals tab *before* following the procedures outlined in "Pre-Generating Your Own Proxies" on page 150.

Note: The pixelScale and pixelRatio parameters are generally obsolete due to the proxy functions in *SFileIn* that were introduced in Shake 2.5. These parameters have been retained for general compatibility, but you probably won't ever use them.

Using interactiveScale

The interactiveScale setting, in the Globals tab, is designed to speed up Shake's interactivity whenever you make adjustments to parameters.

It works by temporarily dropping the image-processing resolution to the proxy resolution that's selected in the interactiveScale parameter whenever you adjust a parameter's controls. While you make adjustments, the image displayed in the Viewer is low-resolution, but is updated much more quickly. As soon as you release the mouse button, the image is rendered at its full resolution (or the current proxy resolution, if you've enabled useProxy). The interactiveScale setting has no effect on your rendered output or Flipbooks.

You can combine this setting with the useProxy setting if the script you're creating is exceptionally slow to render. For example, setting useProxy to P1 lowers the overall processing resolution to 1/2 by default. If you set the interactiveScale setting to 1/4, parameter interactivity will be very fast, and you won't have so long to wait when you release the parameter control to let the image render at the current proxy setting.

The following is an example of how to use the interactiveScale parameter:

1. To follow along, use a *FileIn* node to read in the *saint_fg.1-5#* file, located in the *$HOME/nreal/Tutorial_Media/Tutorial_05/images* directory.
2. Apply a Filter–*RBlur* (not *Blur* or *IBlur*).
3. Set your oRadius to approximately 300 (note the very slow *RBlur*).

4. Open the Globals tab, then set the interactiveScale to 1/3.

5. In the Viewer, drag the center control around. Notice that the image drops to a temporary lower resolution while you make this adjustment. When you release the mouse button, the Viewer image returns to its original resolution.

Modify a parameter...

...Release the mouse button.

Using Temporary Proxies

Unless you specifically do otherwise, Shake generates temporary proxies (also called on-the-fly proxies) that are created only for frames that are displayed, as needed, and that are discarded once your computer's disk cache is full.

Whenever you set the useProxy parameter to something other then Base, Shake scales down the resolution of frames at the position of the playhead as you view them, in order to accelerate your script's performance. Unlike the interactiveScale setting, the image is left at the proxy resolution until you return the useProxy parameter to the Base resolution.

Important: The useProxy parameter affects both Flipbooks and *FileOut* nodes.

Temporary proxy images are written first into memory, and then to the disk cache as memory runs out. Shake keeps track of how many times each cached frame has been viewed, and eliminates the least viewed frames first when the cache runs out of room.

To change to a lower proxy using the default proxy resolutions, do one of the following:

- In the Globals tab, switch useProxy from Base to P1, P2, or P3.

- Use the pull-down button menu in the title bar.

When activated, the proxy button in the title bar is highlighted with the selected proxy.

The default proxy settings are:

Proxy Setting	proxyScale	proxyRatio
Base	1	1
P1	1/2 (.5)	1
P2	1/4 (.25)	1
P3	1/10 (.1)	1

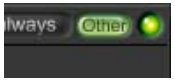
By default, you can select from the predefined proxy sets in the useProxy subtree of the Globals tab. These are common proxy settings, but you can also use your own.

To temporarily set a custom proxy:

1. In the Globals tab, open the useProxy subtree.
2. Change the proxyScale and proxyRatio parameters to the desired settings.

Note: You can also enter the desired proxyScale and proxyRatio, separated by a comma, directly into the useProxy value field.

As soon as the proxyScale or proxyRatio is modified, the useProxy Base/P1/P2/P3 buttons turn to Other, since there is no longer a correspondence to any of the preset proxy settings.



In the following example, the proxyRatio is set to .5. This setting has the added benefit of correcting the anamorphic distortion of the image while simultaneously reducing its resolution.

Full Resolution

.5 proxyRatio

To return to a preset proxy setting:

- Click Base/P1/P2/P3 (or Other in the upper-right corner of the Shake window).

Changing the Aspect Ratio to 0.5

To ensure that your nodes understand the aspect ratio as 0.5 (for example, *Rotate*), open the Globals tab and set the defaultAspectRatio format to 0.5. This does not modify the image, but only affects nodes such as *Twirl* and *Move2D* that are created after you set this ratio. For more information, see "About Aspect Ratios and Nonsquare Pixels" on page 209.

Customizing P1/P2/P3 for a Script or Session

If you consistently require a different group of proxy settings for your project, customized useProxy subparameters can be saved within that script.

To customize the proxy settings for an individual script:

1. In the Globals tab, open the useProxy parameter to reveal its subparameters.
2. Open one of the proxyXDefaultFile subparameters (where X is 1, 2, 3, or 4). In this example, the proxy1DefaultFile and proxy2DefaultFile subparameters are being customized.

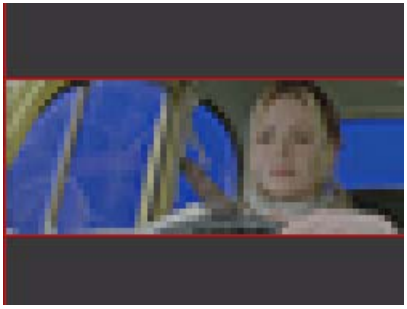
3 Modify the proxy1DefaultScale and proxy1DefaultRatio parameters.

- For example, suppose you want to create a proxy setting that lowers the resolution of an anamorphic image by resizing the image vertically to correct the anamorphic distortion. You want to set P1 to be the same width as the base file (the full resolution image) but flattened, and P2 to be 1/4 scale and also flattened. To do this, use the following values:
  - proxy1DefaultScale 1
  - proxy1DefaultRatio .5
  - proxy2DefaultScale 1/4
  - proxy2DefaultRatio .5

Note: You can now close the useProxy subtree to clean up the Globals tab.

Click P1 or P2 to toggle to the flattened versions of the full-resolution image.

Modified P1



Modified P2

Note: Do not use the ratio parameter to change your aspect ratio on the fly if working with pre-rendered proxies. For more information, see "Anamorphic Images and Pre-Generated Proxies" on page 155.

Permanently Customizing Shake's Proxy Settings

The previous section described saving custom proxy settings into a script. However, if you always use the same custom settings for all new scripts, you can create a new set of default P1/P2/P3 settings by creating a .h file in your *startup* directory.

When an *SFileIn* node is created, three pieces of information are taken from the File Browser:

- The file name
- The proxy level that corresponds to this file name (Base, P1, P2, or P3)
- The set of images to use for that proxy level

The chosen file name appears in the *FileIn* node's proxy1File parameter, and the settings for the selected proxy level from the selected proxy set are used to set the other subparameters of the fields.

Next, the remaining proxy set paths defined in the proxy set are filled into the remaining proxyNWhatever fields in the *SFileIn* node, with path modifications and substitutions made as defined in the DefProxyPath or DefBasePath statements for the proxy set.

The *SFileIn* node's parameters are set via the values, modifications, and substitutions defined in the proxy set, in much the same way that a *Grad* node takes its initial width and height parameter values from the current defaultWidth and defaultHeight values in the format subtree of the Globals tab.

.h File Syntax for Custom Proxy Sets

The syntax for defining proxy sets is as follows:

```
DefProxyGroup("proxySet",
  DefBasePath(
     "baseDefaultFile",
     "baseDefaultFileType",
     defauktAlwaysAdd,
     "baseDefaultReplace"),
  DefineProxyPath(
     const char *proxyPath="../proxy.50/<base>.<format>",
     float scale=.5,
     float aspect=1,
     int bytes=GetDefaultBytes(),
     const char *fileFormat="Auto",
     int render=0,
     int alwaysAdd = 1,
     int index =1,
     string substitionString
);
```

Variable Definitions

This section explains the declarations made in the above script.

proxyPath

Defines the default location for pre-generated proxies. (See the example below.) Note that you can use variables to grab strings from the baseName:

- <base> = image name + frame range
- <format> = format extension
- <name> = image name (no frame range)
- <range> = the frame range
- <dir1>, <dir2>, etc. = the name of the parent directory, two directories up, and so on.

scale

The proxy scale.

aspect

The aspect ratio (the Y-axis scale).

bytes

The default bytes for pre-generated proxies. This does not affect on-the-fly generation of proxies, so you maintain your bit depth if you do not pre-render your proxies.

fileFormat

The file format for pre-rendered proxies. (See below.)

render

Turns on or off the render lights on the Render Proxies menu. When on (1), the files are rendered with the Render Proxies menu. (See below.)

alwaysAdd

When set to 1, an entry is added to an *SFileIn* node at the time of its creation.

index

Sets whether you are P1, P2, or P3 (1, 2, 3). This replaces Shake's default settings, unless you set the index to 0, in which case it is appended.

substitutionString

When the first string is found in the base file name, it substitutes the second string; for example, from the first line, "4096x3112" is substituted by "2048x1556." This string is not always necessary—the proxyPath may already be taking care of differentiating the proxy files if all of the names of the files are the same except for a root directory stating the size of the proxies.

Example

This example sets a proxy of .25 with an aspect ratio of .5. It takes the default bytes setting, turns on the render light for the Render Proxies menu, adds an entry into an *SFileIn*, and is set as P1:

```
DefineProxyPath("../proxy.25.5/<base>.<format>", .25, .5,
     GetDefaultBytes(), "Auto", 1,1,1, "substitutionStrings");
```

You can also create and use predefined proxy sets in the useProxy subtree (in the Globals tab), where you can choose the proxyScale values for P1, P2, and P3. The following example assumes file names such as "4096x3112/name_4k.#.cin,""2048x1556/name_2k.#.cin,""1024x778/name_1k.#.cin," and "410x366/name_sm.#.cin." To set a group, use the following code in a *startup* .h file:

```
DefProxyGroup("4K Fullap",
  DefBasePath("../4096x3112/<base>.<format>",
     "Auto",
     1,
     "2k|4k;1k|4k;sm|4k"),
  DefProxyPath("../2048x1556/.",
     .50,
     1.,
     GetDefaultBytes(),
     "Auto",
     0,
     1,
     0,
     "4k|2k;1k|2k;sm|2k"),
  DefProxyPath("../1024x778/." ,
     .25,
     1.,
     GetDefaultBytes(),
     "Auto",
     0,
     1,
     0,
     "4k|1k;2k|1k;sm|1k"),
  DefProxyPath("../410x366/." ,
     .10,
     1.,
     GetDefaultBytes(),
     "Auto",
     0,
     1,
     0,
     "4k|sm;2k|sm;1k|sm")
);
//This sets the default set at startup, so obviously you only set it once.
script.proxySet = "4k Fullap"
```

The first line names the group as "4k Fullap." The next line describes the base file name. The next three lines that begin with `DefProxyPath` describe the subproxies using the DefineProxyPath definition, except the substitutionStrings. When the first string is found in the base file name, it substitutes the second string. For example, from the first line, "4096x3112" is substituted with "2048x1556."

Note: Because the syntax is somewhat intricate, you may find it easier to copy an example from the *<ShakeDir>/shake.app/Contents/Resources/nreal.h* file and to paste it into a new file.

Using Pre-Generated Proxy Files Created Outside of Shake

It is sometimes convenient to begin working using only pre-rendered proxy-resolution media files (which are typically generated during the scanning process), and to substitute the full-resolution media files later. Shake's proxy structure allows you to do this.

Note: In the following example, it is assumed that all users in your production pipeline are using a standardized naming convention.

To read pre-generated proxies into a script:

1. Open the useProxy subtree in the Globals tab.
2. Do one of the following:
   - Choose an option from the proxySet pop-up menu as described in the previous section. This automatically enters names for the file path names for the base file and the proxy sets.
   - Create custom proxyNDefaultFile settings as appropriate to match the resolution of the pre-generated proxy files you'll be using.
3. If you want to substitute a string in the file name, you can use the defaultReplace parameters in the Globals tab. These subparameters are located within the useProxy subtree, inside of each DefaultFile subtree.

These parameters allow you to replace text in the original DefaultFile paths.

For example, suppose that your P1 proxies are already available on the computer "MyMachine," but your full-resolution elements are not. You start compositing with the proxies, intending to work on the full-resolution elements later. When the full-resolution elements become available, they're located on "Server1." The easiest way to substitute the proxies with the full-resolution media is to use the defaultReplace parameters.

If the proxy was named:

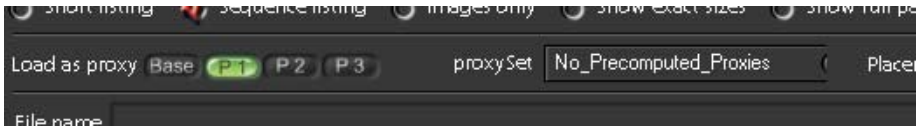
```
//MyMachine/project1/shot1/plate1/proxy1/myfile_proxy1
```

and the full resolution elements are:

```
//Server1/project1/shot1/plate1/full/myfile_full
```

you enter *MyMachine|Server1; proxy1|full* as your baseDefaultReplace. Note the semicolon to split the entries.

4 Create a *FileIn* node in the Node View to read the pre-rendered proxies into your script. When the File Browser appears, choose the proxy files and specify the proxy setting it corresponds to in the "Load as proxy" setting at the bottom of the Browser. In this example, P1 is the proxy setting at which the selected files are read in.



When you import proxy files without corresponding base files, the imageName parameter in the Parameters tab appears olive green.

Later, when you're ready to start using the full-resolution media that corresponds to the proxy files you've been using, you can read in this media using the baseFile subparameter of the *FileIn* node's imageName parameter.

When the full-resolution elements are brought online, the imageName field goes back to its normal gray appearance, and you may toggle to the Base full-resolution mode to display the media at its highest resolution.

Important: Use caution when working with anamorphic elements. The proxy ratio parameter determines the relationship of the proxy to the base file. Therefore, if you load in low-resolution flattened images, ensure that your ratio reflects the proper ratio of the height to the width in the base files. See "Anamorphic Images and Pre-Generated Proxies" on page 155.

Pre-Generating Your Own Proxies

Ordinarily, if you set useProxy to P1, P2, or P3, the proxies created for each frame of the composition are written on the fly to memory. Eventually, the computer's memory fills up, and these temporary proxy images are written to disk—into the cache.

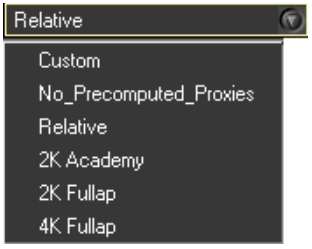
The cache is a closed set of files that are viewable only by Shake. Additionally, remote renders do not recognize the cache directory if not explicitly specified to do so. Finally, you may have so many files that the cache mechanism becomes overworked. In these cases, it makes sense to pre-generate your proxies when you start the project with an initial rendering process. The proxy files are then pulled from these precalculated images rather than generated on the fly.

Important: If you decide to pre-render proxy files for your script within the Shake interface, make sure that you set the proxySet parameter in the useProxy subtree of the Globals tab *before* you generate your proxies.

You can either pre-generate the files inside the interface, or you can load them after they are created by an external process.

To quickly pre-generate files using Shake's default settings:

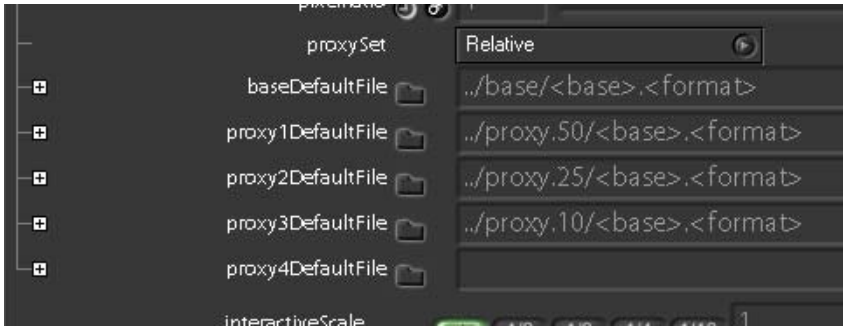
1. Open the Globals tab and open the useProxy subtree.
2. Choose the type of proxies you want to generate from the proxySet pop-up menu. There are six options in this menu:



- *Custom:* This option is automatically selected whenever you choose your own resolutions, names, and format.
- *No_Precomputed_Proxies*: No proxies are generated.
- *Relative:* Three sets of proxies are generated, with defaultScale values that are calculated relative to the original size of the image files:
  - 1/2
  - 1/4
  - 1/10

- *2K Academy:* This option is suitable if your original image files have a resolution of 1828 x 1556. Three sets of proxies are generated, with the following absolute defaultScale values:
  - 914 x 778
  - 457 x 389
  - 183 x 156
- *2K Fullap:* This option is suitable if your original image files have a resolution of 2048 x 1556. Three sets of proxies are generated, with the following absolute defaultScale values:
  - 024x778
  - 512x389
  - 205x156
- *4K Fullap:* This option is suitable if your original image files have a resolution of 4096x3112. Three sets of proxies are generated, with the following absolute defaultScale values:
  - 2048x1556
  - 1024x778
  - 410x366

When you choose an option, the proxyXDefaultFile fields are automatically populated with the appropriate path names. For example, if you choose Relative from the proxySet pop-up menu, the proxyXDefaultFile fields are populated with the following:



By default, proxies are stored in new directories that are created in the same location as the source media on disk. If necessary, you can change the path where the proxies are saved, the directory into which they're saved, the names of the generated files, and the format of the proxies that are rendered. For more information, see "How Proxy Paths Are Defined" on page 155.

3 As an optional step, select the *FileIns* in the Node View that you want to generate as proxies.

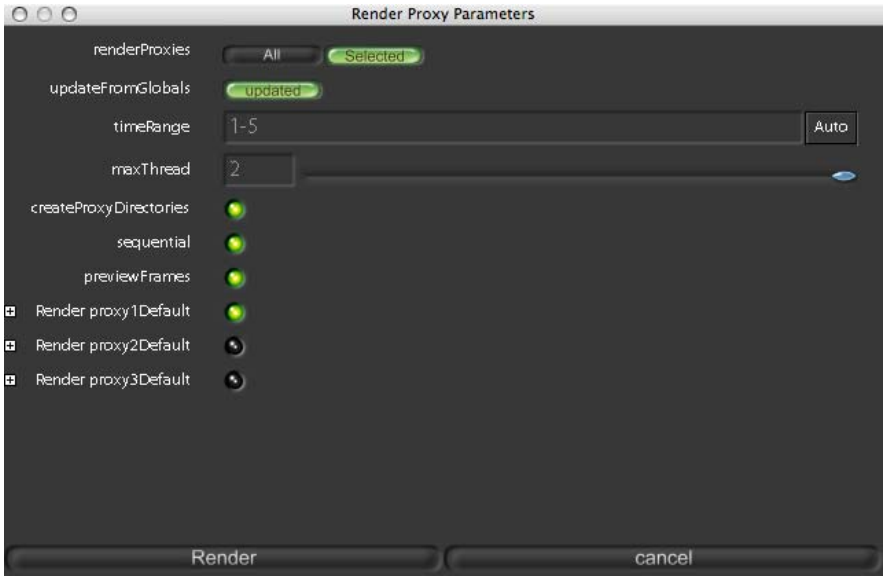
Note: Even if proxies have already been rendered, they will be rendered again. Shake has no way of checking to see if proxy files are already present and/or valid.

4 Choose Render > Render Proxies.

The Render Proxy Parameters window appears.

5 Turn on the proxies you want to generate. In the following screenshot, only the proxy1Default proxy set will be rendered, because the other two sets are turned off.

The Render Proxy Parameters window has the following parameters:



renderProxies
Specifies whether all *FileIn* nodes in the currently open script are rendered as proxies, or just the selected ones.

timeRange
Sets the range of frames to render as proxies. When Auto is clicked, the entire frame range for each clip is rendered. Note that proxies are not generated beyond a clip's frame range.

maxThread
The number of processors used for rendering on multiprocessor systems.

createProxyDirectories
Creates appropriate directories for the new proxy image files.

sequential
When generating many files, it may be more efficient to process each file individually, one after the other, rather than simultaneously. This is equivalent to the *-sequential* flag on the command line.

previewFrames
Displays the thumbnails of the new proxy frames as they're rendered.

Render proxy Defaults
Each proxy set you want to be rendered must be enabled. You can also open each proxyXDefault's subtree to modify any parameters. If you create your own custom proxy setting with a .h file (see above), you can specify if this button is on or off by default.

6 When you're finished changing the settings, click Render.

When Shake finishes rendering, the proxies are ready to be used in your script. Activating the P1, P2, or P3 settings of the useProxy parameter results in Shake loading the pre-generated proxy files you've just created, rather than generating them on the fly.

Rendering Proxies on the Command Line

If you're rendering your proxies from the command line, there are three additional parameters you can specify.

`-renderproxies`

Only renders the proxy subsets of the *FileIn* nodes. If no subsets are specified, what is saved in the script is rendered. Otherwise, you can use *-renderproxies p1 p2 p3*.

`-proxyscale`

You can specify numerical scale and ratio parameters, or use the keywords *Base, p1, p2, p3*.

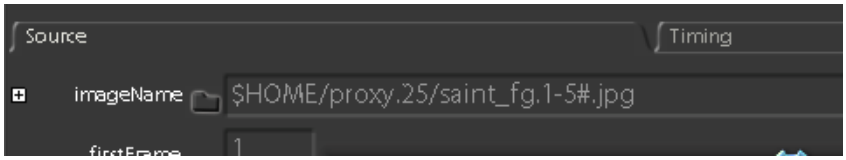
> ***Warning:*** Currently, the command-line proxy keywords referenced above do not correspond to the proxy keywords in the useProxy parameter of the Shake graphical interface. As a result, p1 (command line) = Base (graphical interface), p2 (command line) = p1 (graphical interface), and p3 (command line) = p2 (graphical interface).

`-createdirs`

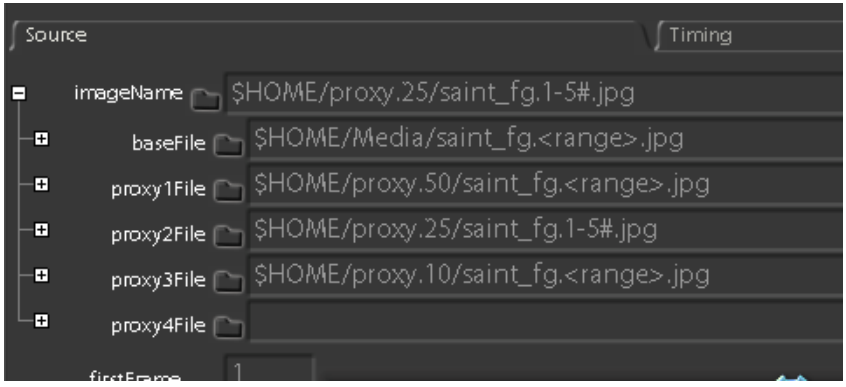
Creates directories for the *-renderproxies* command. Does nothing for normal *FileOut* nodes.

Pre-Generated Proxy File References in FileIn Nodes

When you open a *FileIn* node's parameters in the Parameters tab, the imageName parameter shows which proxy image files are currently being used. For example, generating a set of proxies for an image sequence located in a directory named "Media" in the $HOME directory results in the following path appearing when you set useProxy to P2:

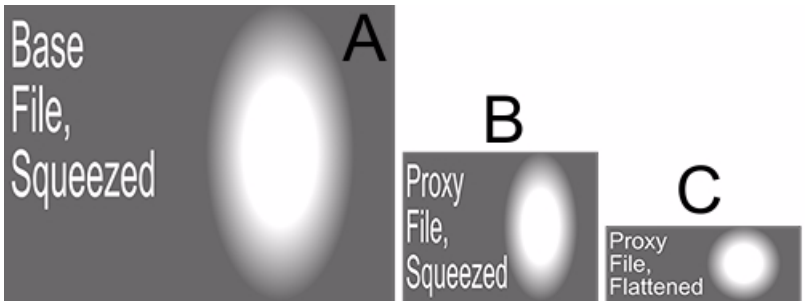


If you open the imageName subtree in the Parameters tab, four proxyNFile fields display information about the media files referenced for each proxy setting.



Anamorphic Images and Pre-Generated Proxies

Do not use the proxyRatio parameter to change your aspect ratio on the fly if working with pre-rendered proxies. This parameter dictates the relationship of the height-to-width ratio between the proxy file and the base file as they exist on disk. Therefore, either ensure you are using flattened pre-generated proxies (that can be a second proxy set), or use the global parameter viewerAspectRatio to flatten the anamorphic proxies. In the following example, image A is the full-resolution anamorphic frame, so it looks squeezed. Image B is a half-size anamorphic-resolution proxy. Image C is a half-size, flattened image. This is how the raw images appear on disk.



Therefore, your proxy settings should be:

- *P1:* Image B, scale of .5, ratio of 1; as the ratio of the height to the width is the same as in image A.
- *P2:* Image C, scale of .5, ratio of .5. The width is the same as image B, the ratio is half of the ratio of the height-to-width of image A.

How Proxy Paths Are Defined

The default paths for proxies generated by Shake use variables that reference the name and format of the original source media files. If the proxySet pop-up menu is set to Relative, then the path of the proxy1DefaultFile is:

```
../proxy.50/<base>.<format>
```

This path translates into the following:

- ../ references the directory level of the directory containing the original media files.
- <base> represents the *image name* and the *frame range* of the original media files.
- <format> represents the format extension of the original media files.

Therefore, if the original media files being referenced are named *MyAmazingFootage.1-100#.cin*:

- <image name> = *MyAmazingFootage*
- <frame range> = *1-100#*
- <base> = *MyAmazingFootage.1-100#*
- <format> = *cin*

For example, suppose the source media of an image sequence using the file name *plate.#* is referenced by the following path:

```
<MyLongPath>/MyHiResImages/plate.#.cin
```

By default, proxy image files are rendered and saved into new directories, which are created within the directory referenced by <MyLongPath> using the following names:

```
<MyLongPath>/proxy.50/plate.#.cin
<MyLongPath>/proxy.25/plate.#.cin
<MyLongPath>/proxy.10/plate.#.cin
```

You can always change the name of the directories that are created to hold the proxies generated by Shake. For example, you can use the variable that defines the file name, plus a suffix, such as:

```
../<name>_half/<base>.<format>
```

You can also change the image format you want the proxy files to be written to by changing <format> to a specific file suffix. For example, if you want to write the proxy files as a .tiff sequence, you simply type:

```
<MyLongPath>/proxy.50/plate.#.tiff
```

Proxies of YUV Files

Use caution when making proxies of YUV files, as they are always at a set resolution. Be sure to toggle your FileType to a different format. By default, the proxies are switched to .iff format.

Organizing Proxy Files

When specifying proxy directories, there are two ways you can organize the proxy files that are generated.

You can place all the images at the same directory level, for example:

```
images/bluescreen1/bs1.#.cin,
images/bluescreen2/bs2.#.cin,
images/cg_plate/cg_plate.#.iff
```

In the following case, the default path puts all proxies into the same subdirectory:

```
images/bluescreen1/bs1.#.cin,
images/bluescreen2/bs2.#.cin,
images/cg_plate/cg_plate.#.iff
images/proxy.50/bs1.#.cin
images/proxy.50/bs2.#.cin
images/proxy.50/cg_plate.#.iff
```

If you have many plates and a high frame count, you may want to put the images for each proxy resolution into separate directories. For example, you can provide a file path such as:

```
../<name>_p.50/<base>.<format>
```

This approach keeps the file count down in each directory, but increases the overall number of directories referenced by your script. Examples of this are:

```
images/bluescreen1/bs1.#.cin,
images/bs1_p.50/bs1.#.cin
images/bluescreen2/bs2.#.cin,
images/bs2_p.50/bs2.#.cin
images/cg_plate/cg_plate.#.iff
images/cg_plate_p.50/cg_plate.#.iff
```

All of your images are in subdirectories based on resolution, for example:

```
images/bluescreen1/2048x1556/bs1.#.cin
images/bluescreen2/2048x1556/bs2.#.cin
images/cg_plate/2048x1556/cg_plate.#.iff
```

In this case, the default naming scheme works fine:

```
../proxy.50/<base>.<format>
```

This gives you:

```
images/bluescreen1/2048x1556/bs1.#.cin
images/bluescreen1/proxy.50/bs1.#.cin
images/bluescreen2/2048x1556/bs2.#.cin
images/bluescreen2/proxy.50/bs2.#.cin
images/cg_plate/2048x1556/cg_plate.#.iff
images/cg_plate/proxy.50/cg_plate.#.iff
```

You can of course use the following as your path to return numerical values for a half-proxy:

```
../1024x778/<base>.<format>
```

You can also use the <*dirN*> variable to access the name of any parent directory. For example:

```
<dir2>/1024x778/<dir2>.<range>.<format>
```

creates:

```
images/bluescreen1/1024x778/bluescreen1.#.cin
```

Full-Resolution Proxies and Network Rendering

If your script references media that resides on a remote network machine, it can sometimes be convenient to create full-resolution duplicates of this media on your local machine.

Using local files can speed your compositing work by eliminating the need for your computer to access media over the network. As an added benefit, using local files speeds up renders on your machine. On the other hand, having your script reference the original files over the network can speed up network renders by preventing networked machines from having to access media on your computer.

You can create a local set of media files by setting the proxyScale of one of the proxy settings (typically P1) to 1. This creates full-resolution duplicates of the files from the network server on your local machine when you use the Render Proxies command.

Doing this allows you to switch back and forth between your local media, and the original network media. When you switch the proxyMode to P1, the local copies of the media files are used. When proxyMode is switched back to Base, the media referenced from the network is used.

To specify this on the command line, use the following commands:

To use the original files:

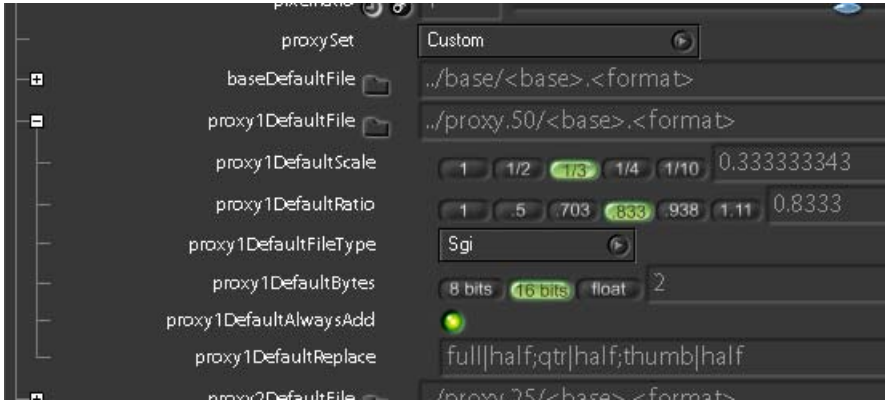
```
shake -proxyscale Base
```

To use the local full-resolution copies:

```
shake -proxyscale 1 1
```

Customizing the Format of Pre-Generated Proxies

Instead of using the default proxy settings, you can open any of the proxyNDefaultFile subtrees and change the scale, ratio, format, and bit-depth parameters for proxies generated using that setting. If you customize these subparameters, the proxySet parameter automatically changes to Custom.



Note: Changing the subparameter settings within a proxyNDefaultFile set does not automatically rename the directory that will be created to hold the proxies that are generated.

The following example uses one of the tutorial clips to illustrate how you can create custom proxy settings to create half-height proxies for anamorphic footage. Do not read the tutorial images in right away.

To pre-generate customized proxies from the Shake interface:

1. In the Globals tab, open the useProxy subtree.
2. Set your proxy1DefaultFile to:

   `/TEMP/saint_p.1x.5`
3. Set your proxy2DefaultFile to:

   `/TEMP/saint_p.25x.5`

   Note: This example requires you to use the tutorial images, and it is likely (if you are at a large facility) that you do not have permissions to create files and directories in the install directory. Therefore, set the proxy directories to be in an open user area. (Typically, you do not save your proxies into the TEMP directory.)
4. Open the proxy1 subtree, and set scale to 1, ratio to .5, format to .iff, and turn proxy1DefaultAlwaysAdd on.
5. Open the proxy2 subtree, and set proxy 2, to 25 and .5, format to .iff, and turn proxy2DefaultAlwaysAdd on.
6. Open proxy3 and disable AlwaysAdd (in this example, you only need two proxy sets).

This group of parameters should now look like this:

7 Now, create a *FileIn* node, and read in the *saint_fg.1-5#* and *saint_bg.1-5#* files from the *$HOME/nreal/Tutorial_Media/Tutorial_05/images* directory.

8 Choose Render > Render Proxies.

9 In the Render Proxy Parameters window, set the frame range to 1-5.

10 Enable Render Proxy1Default and Render Proxy2Default.

11 Click Render.

Your proxies are rendered and available for use. When you click P1, you switch to the half-height images. When you click P2, you switch to the quarter-resolution, half-height images. When you click P3, you are using 1/10 resolution images, but these are generated on the fly, as they were not pre-rendered.

Pre-Generating Proxies Outside of the User Interface

You can also pre-generate proxies for a script from the command line. There are two methods to generate proxies outside of the interface.

Pre-Generating Proxies From the Command Line—Method One

If the base-resolution images are already loaded into a script and you have checked that the proxy paths are correct (see above), you can launch a proxy-only render on the command line with the *-renderproxies* command:

```
shake -exec myscript.shk -renderproxies p1 p2 p3 -t 1-100 -v -
    createdirs
```

This automatically creates the appropriate subdirectories when *-createdirs* is activated. There is no checking for file status, so all images are still re-rendered, even if they already exist. Also, each sequence is only rendered for its actual length, so a five-frame sequence is not rendered out to 100 frames.

The command to specify your proxies looks like the following example, and can be entered into a *startup* .h file or in a script. Its format is identical to what is listed above:

```
DefineProxyPath("../proxy.25.5/<base>.<format>", .25, .5,
    GetDefaultBytes(), "Auto", 1,1,1);
```

Now, no further work is needed to load these proxies back into the user interface, as all paths are determined by the default proxy settings that were saved into the script.

Pre-Generating Proxies From the Command Line—Method Two

If you only have the raw image files, you can use the *-z* or *-zoom* functions to render your images:

```
shake fullres.#.cin -z .5 -fo halfres.#.cin -t 1-100 -v
```

```
shake fullres.#.cin -zoom .5 .5 -fo halfres_halfheight.#.cin -t 1-100
    -v
```

```
shake fullres.#.cin -zoom .5 .5 -bytes 1 -fo
    halfres_halfheight_8bit.#.sgi -t 1-100 -v
```

The first command renders half-resolution Cineon files. The second renders half-resolution flat files (to squeeze scope images). The third command does the same, but writes out 8-bit SGI files.

Using Pre-Generated Proxies in Your Script

After pre-generating your proxies, you need to set up your script so that it references both the original media and the proxies you've created.

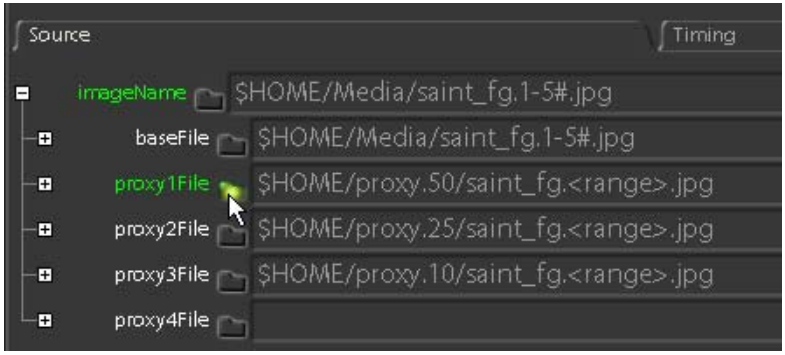
To use pre-generated proxies in a script via the user interface:

1 Read the full-resolution images into a script with a *FileIn* node. The Load as proxy parameter at the bottom of the Browser lets you choose whether the resolution of the image corresponds to the Base, P1, P2, or P3 proxy resolution.

Keeping High-Resolution Elements Offline

If you have not yet loaded your full-resolution images onto a disk and are loading a proxy into the interface, click Cancel to close the File Browser. Then, in the Globals tab, set the useProxy parameter to Base. This helps to automatically calculate the proxy size. Next, return to the *FileIn* and indicate the proxy set of your image with the Browser. Later, when the full-resolution elements go online, you can load in your elements with the *FileIn* node. If your layer does not have a full-resolution element the same size as the option chosen in the format pop-up menu in the Globals tab, you must manually adjust the scale and ratio parameters for the proxy set of that *FileIn*.

2 In the *FileIn* parameters, expand the imageName subtree. The baseName is already supplied, but you'll notice that proxy1File probably has an incorrect default name. Click the folder to browse to the correct location of the proxy media files.



After you load the appropriate files for each proxyNFile parameter, the scale and ratio parameters in the proxyNFile subtrees should be automatically set relative to the full-resolution baseFile image. Otherwise, they're calculated according to the defaultWidth and defaultHeight parameters in the format subtree of the Globals tab.

Note: When you toggle the useProxy parameter from Base to P1, P2, or P3, you do not necessarily load a *FileIn* node's corresponding proxy1File, proxy2File, or proxy3File media. The proxy mechanism loads the set that is closest to the global settings, and does a further scale based on that set. For example, if you haven't loaded a 10-percent pre-generated proxy and you set useProxy to P3, a 10-percent file is generated on the fly from the generated P2 proxy.

When Not to Use Proxies

Proxies are fine for gauging color and relative position. However, proxies do not work well for pixel-sensitive operations such as *DilateErode* (where you may chew just one pixel in or out), or for tracking nodes, due to round-off errors. This is not true for all filters. *Blur*, for example, works fine at a proxy resolution.

If you're performing an operation where it's not advisable to use proxies, an easy way to increase Shake's refresh speed while working on high-resolution images is to activate the Viewer DOD button.

For more information, see "The Domain of Definition (DOD)" on page 82.

Do Not Use Proxies for Tracking

This is worth repeating. Because proxies eliminate detail by lowering an image's resolution, proxies will introduce rounding errors with Shake's motion tracking nodes—including the *MatchMove*, *Stabilize*, and *Tracker* nodes.

Proxies and Z-Depth

Proxies also interfere with Z-depth compositing, as there is no anti-aliasing with the Z channel. The left image is a full-resolution Z composite. The right image is the same composite at a proxy resolution. The anti-aliasing of the depth channel significantly alters the image quality.

Full resolution

Proxy image

This quality loss can be somewhat minimized by using a Box filter for your proxies, but then the rest of your image quality suffers as well.

Proxy Parameters

The following tables list proxy parameters everywhere they appear in Shake, in the Globals tab, and in each *FileIn* node's Source tab.

In the Globals Tab

The useProxy subtree in the Globals tab has the following parameters that let you customize how Shake handles proxies for the media used by your script:

useProxy
Specifies the proxy set to be used. The sizes are determined by opening proxy1File, Proxy2File, and so on, and setting the scale/ratio parameters. The two numbers are the scale and ratio of the proxy.

useProxyOnMissingFile
When active, substitutes a proxy generated from an associated proxy file when a missing image is encountered.

proxyScale
A temporary setting (though it is saved into a script) for the proxy resolution that is overwritten when useProxy is set. If setting this parameter matches up with a proxy set, the proxy set is automatically activated.

proxyRatio
A temporary setting (though it is saved into a script) for the squeeze on the Y axis to compensate for nonsquare pixel images that are overwritten when useProxy is set. If setting this parameter matches up with a proxy set, the proxy set is automatically activated.

proxyFilter
The filter used in the sampled images. The default is generally fine, although you may want to switch to Box when working with Z-depth files.

pixelScale
An obsolete function to scale all pixel values.

pixelRatio
An obsolete function to scale all pixel values.

proxySet
A pop-up menu with pre-defined sets of proxy resolutions.

Adding Your Own Entry to the proxySet Pop-Up Menu

You can define your own proxy set to appear in this menu via a .h file. This automatically sets the paths and sizes for each set. You can also declare a proxy set for a specific *FileIn* during browsing.

A predefined proxy set looks like this:

```
DefProxyGroup("4K Fullap",
DefBasePath( "../4096x3112/<base>.<format>"),
 DefProxyPath("../2048x1556/.", .50, 1., GetDefaultBytes(), "Auto", 0, 1,
     0, "4096x3112|2048x1556"), DefProxyPath("../1024x778/." , .25, 1.,
     GetDefaultBytes(), "Auto", 0, 1, 0, "4096x3112|1024x778"),
DefProxyPath("../410x366/." , .10, 1., GetDefaultBytes(), "Auto", 0, 1, 0,
     "4096x3112|410x366")
 );
```

baseDefaultFile

This is used when you bring in pre-rendered proxies before loading in the full-resolution elements. It is assumed you are using a standardized naming convention. Therefore, by using this naming convention, Shake can establish the name of the full resolution elements, based on the name of the proxy you supplied. Opening the baseDefaultFile subtree reveals three more parameters:

- *baseDefaultFileType:* The anticipated file type of the full resolution element.
- *baseDefaultAlwaysAdd:* Whether to add this into the *FileIn*.
- *baseDefaultReplace:* Lets you replace strings in the first loaded proxy set to be replaced by a second string, taking the format: *source|replace;source|replace*. For example, if you always have *_lr* at the end of a low-resolution file name and *_hr* at the end of a high-resolution file name, you could use *_hr|_lr* to automatically change *myfile_hr* to *myfile_lr* for that proxy set.

proxyNDefaultFile

The default file path for each of the four proxy sets you can specify. Relative paths are relative to the image read in with a *FileIn*. The proxyNDefaultFile subtree has six additional parameters:

- *proxyNDefaultScale:* The setting for that proxy set, also setting P1, P2, and so on. For example, proxy1 is P1.
- *proxyNDefaultRatio:* The Y scaling for that proxy set. If working with pre-rendered elements and with anamorphic elements, be sure that this setting reflects the height-to-width relationship between the proxy and base files as they actually exist on disk. See "Anamorphic Images and Pre-Generated Proxies" on page 155.
- *proxyNDefaultFileType:* When pre-rendering your proxy files, they are stored in this format. This has no effect with on-the-fly proxies.

- *proxyNDefaultBytes:* The bit depth for pre-rendered proxies. This has no effect with on-the-fly proxies.
- *proxyNDefaultAlwaysAdd:* When enabled, this proxy set is added to a *FileIn* node when created.
- *proxyNDefaultReplace:* See *baseDefaultReplace*.

textureProxy
The proxy level at which texture-rendered images that are used by the MultiPlane's hardware-rendering mode are displayed in the Viewer. This is similar to the interactiveScale setting, in that the proxy level that's set here is used to generate on-the-fly Viewer images.

interactiveScale
Sets a temporary proxy setting that is active only when modifying a parameter. Fully compatible with the other proxy methods.

FileIn

Each *FileIn* node also contains parameters for defining proxy usage.

BaseFile
The original, high-resolution image sequence or movie file associated with this node. Opening this subtree reveals one additional parameter:
- *baseFileType:* Tells Shake what the format is if the file suffix is ambiguous. Generally, you do not need to set this, as "Auto" automatically detects the format. If you have a problem reading an image, try setting the format to correct this.

ProxyXFile
The directory for pre-generated proxies. This is ignored for on-the-fly proxies.
- *proxyNScale:* The scaling factor for that proxy set.
- *proxyNRatio:* The Y scaling factor for that proxy set, used for anamorphic files.
- *proxyNFileType:* The file type for pre-generated proxies, ignored for on-the-fly proxies.

Compatible File Formats and Image Resolutions

5

The first part of this chapter covers the many file formats with which Shake is compatible. The second chapter covers how to control image resolution.

File Formats

The *FileIn* node can read in two kinds of media—image sequences and QuickTime files. Image sequences are simply collections of image files, where each frame of film or video corresponds to one image file. QuickTime files, on the other hand, contain every frame of media inside of a single file. Which media format is more useful for you depends on your production pipeline.

Note: QuickTime files can only be read and written by Shake on Mac OS X.

Shake Does Not Support HDV

Shake does not support long-GOP formats, including HDV, MPEG-2, or MPEG-1. If you want to use HDV media that was captured with Final Cut Pro or Final Cut Express HD in Shake, recompress it with the Apple Intermediate Codec first.

Image Sequences

The individual image frames in an image sequence can be saved in a wide variety of formats. Interlaced frames for video contain both fields within the image file for that frame.

Each frame of an image sequence has the frame number saved as part of its file name. These frame numbers can contain padding to keep the length of the file names constant (Image0001, Image0002, Image0003, and so on) or padding can be left off (Image1, Image2, Image3, and so on).

When creating an image sequence for use by Shake, it is good practice to include the file extension (for example, .iff, .cin, .tif, and so on), but Shake does not necessarily need it. In general, you use the extension to define the input or output format.

Shake is a hybrid renderer—it adapts its rendering from either scanlines or a group of tiles. This means it never has to load the entire image, just a single piece of the image, making a much smaller memory footprint than other compositors. Sometimes you cannot load just a single line, for example, when using a *Rotate* node, in which case Shake internally breaks the image down into small tiles to work with more manageable bits.

QuickTime Files

Shake on Mac OS X supports the reading and writing of QuickTime files. QuickTime support in Shake is limited, however. Embedded Flash and SMIL content is ignored, as are all of QuickTime's interactive features.

Audio tracks are also ignored by the *FileIn* node. If you read in a QuickTime file, you must import its audio via the Audio Panel in a separate process. For more information on using the Audio Panel, see Chapter 9, "Using the Audio Panel," on page 277.

If you reference a QuickTime movie with more than one video track, Shake only reads the first video track into your script; all others are ignored.

Note: You cannot read out (export) a QuickTime movie with a dynamically varying frame size. The resulting file will be unusable.

Which Codec Is Best?

Compositing with Shake is best accomplished using source media files with little or no compression. Ideally, you should then capture or create QuickTime movies for use with Shake using codecs that apply the least amount of compression possible. QuickTime includes the Uncompressed 8 and 10-bit 4:2:2 QuickTime codecs, as well as other codecs for various formats of standard- and high-definition video. There are also a variety of third-party codecs available that provide other bit depths and compression ratios.

Note: Always check with the developer regarding the compatibility of third-party codecs with Shake.

Image Formats That Support tTmp Files

Shake creates temporary files (*tmp* files) when writing certain formats of images, or when running out of memory. These temporary files are written like swap files, but are used before memory-intensive activity occurs to avoid the slowdown of swapping.

Normally, Shake reads in only the portion of an image that it can fit into memory—either a group of scan lines or a tile of the image. This means that any given image is accessed not just once, but many times, with only the necessary portion of it being read each time in order to save memory and processing time.

The ideal format to support this behavior is the native Shake .iff format (this format is also licensed to Alias/Wavefront for their Maya software), but several other image formats support this behavior as well.

There are some formats that do not support the ability to efficiently read a random portion of the image. As a result, these images can take significantly longer to load, and may require more memory.

Note: QuickTime files do not support the creation of *tmp* files.

Create Temporary Files	Do Not Create Temporary Files
Alias	AVI
BMP (depending on orientation)	DXP
Cineon (depending on orientation)	GIF
JPEG	IFF
PBM	Mental Images
Softimage	.mov/QuickTime
Targa (depending on orientation)	OpenEXR
TIFF (depending on orientation)	PNG
YUV	RLA
	SGI
	Side FX

To calculate the maximum disk space needed to accomodate *tmp* files for each *FileIn* node during I/O, use the following formula:

*tmp file = width * height * number of channels * bytes*

where bytes = 1 for 8 bit, 2 for 10 or 16 bits, 4 for float.

Nodes That Create tmp Files

Certain nodes also create *tmp* files of their own during the processing of a script. This is required for nodes that drastically change the X, Y position of an image's pixels during rendering. For example, if you rotate an image 90 degrees, the pixel in the lower right now moves to the upper right. In order to process this, Shake creates a *tmp* file that includes as much information as necessary to calculate the image. A tiling system is used, so these *tmp* files are typically much smaller than those created during I/O. The nodes that create *tmp* files include the following:

- *Move2D*
- *Move3D*
- *Rotate*
- *Orient*
- *Flip*
- All Warps (*Warper, Morpher, WarpX*, *DisplaceX*, *Turbulate*, and so on)

By default, temporary files are written to: `/var/tmp.`

To relocate the temporary directory, set the environment variable TMPDIR:

```
setenv TMPDIR /more_disk_space/tmp
```

Table of Supported File Formats

The table in this section outlines all of the image formats that Shake supports, with columns for the file extension, image format, supported input and output channels, compression options, bit depth, and *tmp* file support of each format.

Shake supports several combinations of the following input channels:

- *BW:* Black and White
- *RGB:* Red, Green, and Blue
- *A:* Alpha Channel
- *Z:* Z channel, for depth

For example, BW[A] is either BW or BWA. BW[A][Z] is any combination of BW, alpha, and Z. RGB[A] and RGB[A,Z] are optional additions of alpha or Z channels.

Note: Targa and SGI have different input/output options for channels. When you write a BWA image, it is converted to RGBA. Also, many options must be explicitly stated when in command-line mode. For example, Cineon and JPEG files always write in RGB unless you specify every argument found in the *FileOut* node for Cineon or JPEG in the Shake interface.

Compression Controls

Compression controls indicate any special compression techniques. Note that Cineon and YUV have no compression.

An asterisk indicates additional format notes (following the table).

Extension	Image Format	Input Channels	Output Channels	Compression	Bit Depth	tmp Files
.iff* (or no extension)	Shake native	BW[A, Z], RGB[A, Z]	Same		8, 16, float	No
.nri	Shake icon (only for interface icons)	RGB[A]	Same		8	No
.iff*	Alias/ Wavefront Maya (licensed from Apple)	RGB[A, Z]	Same		8, 16, float	No
.als, .alias, (pix)	Alias/ Wavefront Alias	RGB	Same		8	Yes
.alsz	Alias/ Wavefront Alias Z buffer	Z, BW[A, Z], RGB[A, Z]	Same		8,16, float	Yes
.avi*	Microsoft video file format	RGBA	Same	Lossy, from 0 to 1, 1 = high quality		
.bmp, .dib	BMP	RGB	Same		8	
.ct, .ct16, .mray	Mental Ray	RGBA	Same		8,16, float	No
.cin*	Kodak Cineon	RGB[A]	Same	None	16 (10 on disk)	Yes
.dpx	DPX reader courtesy of Michael Jonas, Das Werk Gmbh, modified by Apple	RGBA	Same	N/A	8, 16	Yes
.exr	OpenEXR	RGB[A, Z] (supports any number of additional channels)	Same	Options for both lossless and lossy compression ratios from 2:1 to 3:1	16-bit float, 32-bit float, (32-bit int supported in Z channel)	No
.gif (read only)	GIF	RGB	N/A	N/A	8	No

Extension	Image Format	Input Channels	Output Channels	Compression	Bit Depth	tmp Files
.jpeg, .jpg, .jfif*	JPEG	BW, RGB	Same	Lossy, from 0 to 100%. 100 = high quality	8	Yes
.pbm, .ppm, .pnm, .pgm	PBM	BW, RGB	Same		8	Yes
.pic	Softimage	RGB[A]	Same		8	Yes
.png	PNG	RGB[A], BW [A]	Same		8, 16	No
.psd*	Adobe Photoshop	RGB[A]	RGBA		8, 16	
.mov, .avi (QuickTime*)	Apple video file format, multiple codecs supported	RGB[A]	Same	Lossy, from 0 to 1, 1 = high quality	8, 16	
rgb, sgi, bw, raw, sgiraw*	SGI	BW[A], RGB[A]	RGB[A]	Lossless RLE	8, 16	No
.rla*	Alias/ Wavefront RLA (supports Z buffer)	BW[A,Z], RGB[A,Z]	Same		8, 16, float	No
.rpf*	RLA Rich Pixel Format. Use this type when saving RLA files with Z depth to be read into Adobe After Effects. Make sure the file extension is still .rla, but set the format to .rpf.	BW[A,Z], RGB[A,Z]	Same		8, 16, float	No
.tdi	Alias/ Wavefront Explore format (identical to .iff)	BW[A, Z], RGB[A,Z]	Same		8, 16, float	No

Extension	Image Format	Input Channels	Output Channels	Compression	Bit Depth	tmp Files
.tdx	Alias/ Wavefront Explore Tiled Texture Map	BW[A, Z], RGB[A,Z]	Same		8, 16, float	No
.tga	Targa	RGB[A]	RGB[A]	On/Off	8	Yes
.tif, .tiff	TIFF	BW[A], RGB[A]	Same	4 options, see below.	8, 16, float	Yes
.xpm	XPM	RGB[A]	Same		8	
.yuv, .qnt, .qtl, .pal*	YUV/Abekas/ Quantel	RGB	Same	Uncom- pressed files with YUV encoding	8	Yes
.yuv (10-bit)	Same	RGB	Same	Uncom- pressed files with YUV encoding	16 (10)	Yes

Format Descriptions

The following section discusses some of the more useful image formats (those indicated with an asterisk in the table above) in greater detail.

IFF

The Shake IFF (.iff) format is not the same as the Amiga format with the same extension, although they share certain structural similarities. The IFF format is licensed to Alias/Wavefront for use with Maya, so Shake is ideally suited to work with Maya. Since Shake deals with this format internally, you get the best performance by maintaining your intermediate images in this format as well. The IFF format can accomodate 8, 16, or 32 bits per channel, as well as maintain logarithmic information, alpha, and Z channels. Currently, not many packages explicitly support this format, but if the package supports the old TDI (.tdi) format, it works with IFF as well (for example, with Interactive Effects' Amazon 3D Paint).

CIN

Shake works with images bottom-up, meaning 0,0 is at the bottom-left corner. The Cineon and TIFF formats allow you to write the files either bottom-up or top-down. Because of Shake's bottom-up nature, the I/O time (actual render time remains the same) is four times greater when dealing with top-down Cineon or TIFF files. You can set how Shake writes the images—reading either way is no problem, except for the speed hit. This information is placed in a *startup.h* file.

To set Shake to write images in top-down mode:

- Add the following lines to a .h file in your *startup* directory:

```
script.cineonTopDown = 1;
script.tiffTopDown = 1;
```

You can also set environment variables in your .cshrc (or .tcshrc or whatever):

```
setenv NR_CINEON_TOPDOWN
setenv NR_TIFF_TOPDOWN
```

(For more information on setting up your own .h files, see Chapter 14, "Customizing Shake.")

By default, Cineon and TIFFs are set to the slower top-down mode, since many other software products do not recognize bottom-up images. If you write a bottom-up image and it appears upside down in another software package, you have four choices:

- Reset the TopDown switch/environment variable, and save the image again in Shake.
- Flip the image in Shake before saving it.
- Flip the image in the other software.
- Call the other vendor and request that they properly support the file formats in question.

DPX

The reading and writing of DPX images has been improved for greater compatibility with more film recorders.

When you read in a DPX image, its header data is passed down through the node tree. If you read in a DPX image, process it with single input nodes, such as color, filter, or transformation nodes, and then render (with a *FileOut* node) the result as another DPX file, the header data is passed through the node tree and written out to the resulting file. For more information about Shake's support for custom file header metadata, see "Support for Custom File Header Metadata" on page 178.

When rendering a DPX file with a *FileOut* node, an additional parameter allows you to specify the orientation of the output image as either Top to Bottom (default), or Bottom to Top.

OpenEXR

OpenEXR (.exr) is an extremely flexible, cross-platform file format developed and maintained by Industrial Light & Magic. Key features of the OpenEXR format include support for the efficient storage of high dynamic-range image data using the 16-bit float "half" format, and support for auxiliary image data channels. OpenEXR 16-bit float and 32-bit float data channels can be read directly into Shake's RGBAZ data channels. In addition, OpenEXR 32-bit unsigned integer channel data can be read into Shake's Z data channel, although Shake's image processing nodes cannot process this data.

Note: 32-bit unsigned integer channel data will only be useful to custom plug-ins with built-in logic capable of processing the data within the Z channel.

A major feature of the OpenEXR format is its ability to support an extremely wide dynamic range. Thanks to its floating-point support, a contrast range of up to 30 f-stops can be supported with no loss of precision. Color resolution in 16-bit float ("half") files is 1024 steps per f-stop.

Another advantage of the OpenEXR format is support for any number of additional auxillary data channels, in addition to the standard RGBAZ channels. For example, additional channels can be written to store luminance, surface normal direction channels, velocity channels, and even individual lighting passes written from a 3D rendering application.

Shake Support for Auxiliary OpenEXR Data Channels

Internally, Shake only supports the processing of RGBAZ channels down the processing tree. However, the *FileIn* node provides channel remapping options in the Image tab of a *FileIn* node's parameters. Each channel that Shake supports has a corresponding pop-up menu. Each menu presents a list of every compatible channel within the referenced OpenEXR file. Color channels that correspond to Shake's supported channels are mapped by default.

FileIn channel pop-up menus for two files with different image channels

To remap any image channel:

1. Load a *FileIn* node that references an OpenEXR file into the Preferences tab.
2. Choose a new channel to map to from any color channel pop-up menu.

Channel remapping has the following restrictions:

- Any 16-bit or 32-bit float channel can be remapped within the RGBAZ channels.
- 32-bit integer channels can only be remapped to the Z channel.

  Note: If you want to access multiple 32-bit integer channels within a script, duplicate a number of *FileIn* nodes equal to the number of channels you want to access, then remap each 32-bit integer channel to a Z channel of one of the duplicate *FileIn* nodes.

Support for Data Compression

The OpenEXR format supports several codecs, with options for either lossless or lossy compression. Compression ratios range from 2:1 to 3:1.

Note: By default, *FileOut* nodes set to output OpenEXR images default to the Piz codec.

The following codec information appears courtesy of Industrial Light & Magic:

- *none:* No compression is applied.
- *RLE:* (Lossless) Differences between horizontally adjacent pixels are run-length encoded. This method is fast, and works well for images with large flat areas. But for photographic images, the compressed file size is usually between 60 and 75 percent of the uncompressed size.
- *ZIP:* (Lossless) Differences between horizontally adjacent pixels are compressed using the open source zlib library. ZIP decompression is faster than PIZ decompression, but ZIP compression is significantly slower. Photographic images tend to shrink to between 45 and 55 percent of their uncompressed size.
- *PXR 24:* (Lossy) After reducing 32-bit floating-point data to 24 bits by rounding, differences between horizontally adjacent pixels are compressed with zlib, similar to ZIP. PXR24 compression preserves image channels of type HALF and UINT exactly, but the relative error of FLOAT data increases to about 3°—10-5.This compression method works well for depth buffers and similar images, where the possible range of values is very large, but where full 32-bit floating-point accuracy is not necessary. Rounding improves compression significantly by eliminating the pixels' eightleast significant bits, which tend to be very noisy, and difficult to compress.
- *Piz:* (Lossless): This is the default compression method used by Shake. A wavelet transform is applied to the pixel data, and the result is Huffman-encoded.This scheme tends to provide the best compression ratio for typical film images. Files are compressed and decompressed at roughly the same speed. For photographic images with film grain, the files are reduced to between 35 and 55 percent of their uncompressed size.

OpenEXR Proxy Handling

Shake can read both tiled and scanline OpenEXR images. Scanline files contain a single image at a set resolution, but tiled files hold several versions of the same image at a variety of resolutions, for use as proxies in supporting applications.

Shake's proxy mechanism does not take advantage of tiled images. As a result, Shake defaults to reading in the highest available tiled resolution.

For More Information

More information about the OpenEXR format can be found at http://www.openexr.com.

JPEG

In the *FileOut* node you can set the quality level of these image formats (.jpeg, .jpg, .jfif) and determine which channels are present in the file.

MOV, AVI

The QuickTime format (.mov, .avi) is available on Macintosh systems only. AVI files are written through QuickTime. If you select either as your output format, you have three additional options:

- *codec:* A pop-up menu with a list of all available compressors with Animation (RLE compression) as the default codec.
- *compressQuality:* 0-1. A high value sets high quality and a larger file size.
- *framesPerSecond:* This setting is embedded in the file.

When QuickTime files are rendered from the interface, the Shake Viewer displays the thumbnail. You must close this window before the QuickTime file is actually completed. For this same reason, you cannot write over the file on disk while it is still being viewed.

Important: When using the *FileOut* node to render uncompressed QuickTime movies, use the Apple Uncompressed 8- or 10-bit 4:2:2 codecs to obtain the highest quality.

PSD (Photoshop)

There are two ways to import Photoshop files. First, you can use a *FileIn* node to import a .psd file and select either the merged layers or a single layer. These controls are located in the *FileIn* parameters.

The second way to import a Photoshop file is to use the File > Import Photoshop Script command. Each layer is imported as a separate file and fed into a *MultiLayer* node. For information on the Photoshop layering modes, see "Importing Photoshop Files" on page 473.

RGB, SGI, BW, RAW, SGIRAW

With each of these image formats you have the option to set the channels saved into the output file.

RLA, RPF

Adobe After Effects and Autodesk 3ds max do not properly support the original Wavefront RLA file specifications for the Z channel. Therefore, you have to write the image in a specific format—rpf (Rich Pixel Format) in the *FileOut* node with the .rla file extension in the file name, or else these packages do not recognize the extension.

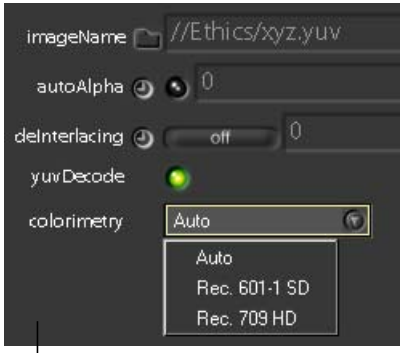
YUV, QNT, QTL, PAL

You have the choice to write out these image formats in NTSC, PAL, or 1920 x 1080 4:2:2 8 bit. You can also write them out as10-bit YUV files.

Note: The .qnt and .qtl options do not appear in the fileFormat list in the *FileOut* parameters, and must be entered manually when setting your *FileOut* name.

When yuvFormat is set to Auto, the resolution is automatically determined by the resolution of the *FileIn* node. The selected resolution is the smallest possible to fit the entire image. For example, if the image is smaller than NTSC, it is NTSC. If it is between NTSC and PAL, it is PAL; otherwise it is HD. You can also manually select the resolution. The *script.videoResolution* is no longer used for this purpose.

Note: The YUV file reader and writer supports Rec. 709 and Rec. 601-1 colorimetry coefficients, used primarily in HD YCbCr (HD-SDI).



YUV *FileIn* parameters

YUV *FileOut* parameters

Support for Custom File Header Metadata

Internal support for blind data allows for the preservation of metadata from custom file formats for facilities using special file translators.

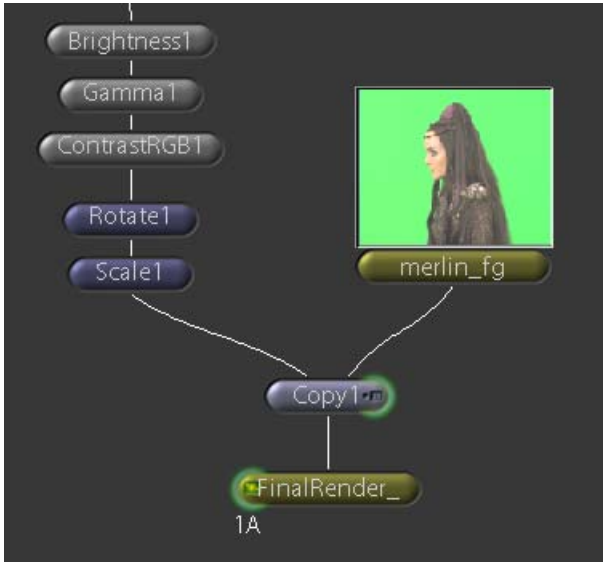
If you design a file translator that places a file's header metadata into Shake's blind data container, it will be passed down through the node tree. For example, if you read in an image with custom metadata using such a file translator, process it with a series of single input nodes, then render out the result into the same format with a *FileOut* node, the blind header data is passed through the node tree and written to the resulting file.

If two images with metadata in the blind data container are combined in a node, such as *Over*, *Outside*, or *Multilayer*, the data from the image connected to the node's left-most input (frequently labeled the foreground input knot) is propagated down the tree.

If, at some point in the node tree, you wish to assign the blind header data from one image to another, use the *Copy* node. This can be useful if you have a complex node tree that uses many layer nodes combining several images, yet you want the final file to render out with specific header data taken from one of the *FileIn* nodes.

To assign blind header data from one image to another:

1 Add a *Copy* node to the node tree, so that the nodes providing the RGBA data you want to use are connected to the Foreground input knot.



2 Attach a second *FileIn* node containing the blind header information you want to use to the *Copy* node's background input knot.

3 In the *Copy* node's Parameters tab, turn on the copyBData parameter.

The resulting output from the Copy node contains the blind header data from the second *FileIn* node. This operation replaces any header data that was originally in the image.

Table of File Sizes

In the following table, all sizes are for 3-channel images. Note that many images support optional alpha or Z channels, which add to the file size. A single channel image is typically one-third the size. The two sizes listed in each cell are for a *Ramp* (an example of extreme compression), and a completely random image, each in MB. Normal plates tend to be in between, usually closer to the higher value. This can give you a very wide variation in an image. For example, an .iff goes from 2.5 MB for a 2K 8-bit ramp to 9.1 MB for the same size/depth with a random image. If only one entry is listed, it is an uncompressed file.

Extension	NTSC, 8 Bits	NTSC, 16 Bits	NTSC, Float	2K, 8 Bits	2K, 16 Bits	2K, Float
.cin (10 bits)		1.3		12.2		
.iff	.74 | 1	1.7 | 2	2.9 | 3.9	2.5 | 9.1	11.5 | 18.2	22.5 | 35.8
.jpg (100 %)	.02 | .48		1.4 | 4.3			
.mray	1.3	2.7	5.3	12.2	24.3	48.6
.pic	.9 | 1		1.5 | 9.1			
.rla	.8 | .92	1.8 | 2	4	2.3 | 9.2	11.5 | 18.4	36.5
.sgi	.74 | 1	1.8 | 2		2.3 | 9.3	16.5 | 18.5	
.tif	.04 | 1.4	.07 | 1.6	3 | 3.7	.25 | 12.5	.2 | 17.8	27.4 | 33.3
.xpm	.68 | 1.2		6.1 | 10.6			
.yuv	.68		4 (HD)			

Controlling Image Resolution

Shake has no formal working resolution to set. Setting the defaultWidth and defaultHeight in the format subtree of the Globals tab only affects the size of newly created Shake-generated images, including those created by the *Checker, Color, ColorWheel, Grad, QuickPaint, QuickShape, Ramp, Rand, RGrad, RotoShape,* and *Text* nodes in the Image tab. The actual resolution of your composition is primarily determined by those of the input images, and can be modified by a variety of operations as you work down your node tree. For example, if you read in a D1 resolution image, your project starts out at D1 resolution.

Note: While it may seem that choosing a new setting from the format pop-up menu in the Globals tab may be changing the size of the image in the Viewer, this is not true. What you're really seeing is the change being made to the defaultViewerAspectRatio parameter, which changes how the image is scaled in the Viewer in order to compensate for images using nonsquare pixels (like standard definition NTSC and PAL video). This parameter has *no* actual effect on the resolution of your final image.

Combining Images of Differing Resolution

When you composite images with different resolutions using one of the *Layer* nodes, you can select which image defines the output resolution of this operation using the clipMode parameter. This parameter allows you to select either the foreground (first image) or background (second image) resolution to use for the final image.

For example, if you read in 50 2048 x 1556 images, you are working at 2048 x 1556. If you composite all of the images over a 720 x 486 background, and you choose the background resolution of that composite, then the foreground elements are cropped to the video resolution, or you can choose to remain at the 2048 x 1556 resolution.

Because of the Infinite Workspace, you never have to crop a lower-resolution element to match a higher-resolution element when compositing or applying transformations.

Note: You can view the resolution of an element in the title bar of the Viewer, or position the pointer over the node and look at the help text in the lower-right window.

Changing Resolution

You can change the resolution of an element in your composition in several ways. In the following examples, a 640 x 480 foreground element is composited over a 720 x 486 NTSC D1 element. The dark gray area designates space outside of the image frame.

Foreground, 640 x 480

Background, 720 x 486

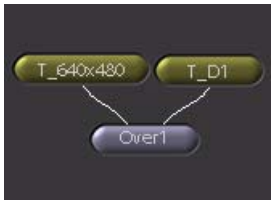
Changing Resolution by Layering

One of the simplest ways you can handle this case is by using layering nodes to crop the frame.

To set resolution by compositing two images of different resolutions:

- Select the background or foreground resolution in the clipMode parameter in the layering node.

Note: This method works even when compositing a pure black plate generated with the *Color* node.

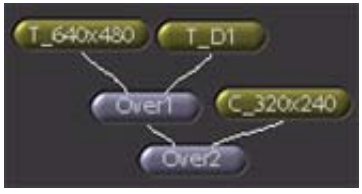


Tree

Background, 720 x 486

Foreground, 640 x 480

In the following example, a 320 x 240 black frame is created with an Image–*Color* node. The resolution of the foreground and background elements is set to 320 x 240 by assigning the background clipMode in the second *Over* node.

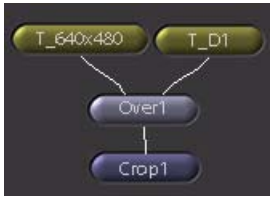


Crop the Frame using the Crop, Window, or Viewport Nodes

The following three nodes, located in the Transform tab, crop the frame without scaling and filtering the pixels.

- *Crop*: Arbitrarily supplies the lower-left and upper-right corners, and cuts into the frame at those coordinates. Also turns off the Infinite Workspace.
- *Window*: Supplies a lower-left coordinate, and then the image's resolution. Cuts off the Infinite Workspace.
- *Viewport*: Same as *Crop*, except it maintains the Infinite Workspace. This is useful for setting resolutions in preparation for later use of onscreen controls, as the controls are always fitted to the current resolution.

In the following example, a *Crop* node is attached, with the *Crop* parameters set manually to 244, 92, 700, and 430. These settings return a 456 x 338 resolution (this is completely arbitrary). Notice the onscreen control you can use to adjust the resolution manually.



Using the Resize, Fit, or Zoom Node to Scale the Frame

The following three nodes change the resolution by scaling the pixels.

- *Resize*: You set the output resolution of the node, and the image is squeezed into that resolution. This usually causes a change in aspect ratio.

- *Fit*: Like *Resize*, except it pads the horizontal or vertical axis with black to maintain the same aspect ratio.

- *Zoom*: Same as *Resize*, except that you are supplied with scaling factors, so a zoom of 1, 1 is the same resolution; 2, 2, is twice the size; .5, .5 is half the size, and so on.

For more information on the individual transform nodes that can be used to change resolution, and tables listing the differences between the scaling functions, see Chapter 26, "Transformations, Motion Blur, and AutoAlign," on page 763.

Nodes That Affect Image Resolution

All of the nodes in this section can be used to modify image resolution.

Fit

The *Fit* node changes the image resolution, resizing the image to fit inside of a frame. *Fit* does not stretch the image in either axis; it zooms X and Y by the same amount until either value fits in the new resolution (that you specify). For example, if you have a 100 x 200 image, and fit it into a 250 x 250 resolution, it zooms the image by 25 percent (250/200 = 1.25), and pads black pixels on the left and right edges.

Note: The *Fit* node allows you to use different scaling filters to scale the image horizontally and vertically.

Parameters

This node displays the following controls in the Parameters tab:

xSize, ySize

The new horizontal and vertical resolution. This parameter defaults to the *width* expression.

xFilter, yFilter

The methods used to scale the image horizontally and vertically. Choosing *Default* uses the *sinc* filter when scaling the image down, and the *mitchell* filter when scaling the image up. For more information, see Chapter 28, "Filters."

preCrop

Turns off the Infinite Workspace so that the letterbox area remains black.

Resize

The *Resize* node resizes the image to a given resolution.

Parameters

This node displays the following controls in the Parameters tab:

xSize, ySize

The new horizontal and vertical resolution. This parameter defaults to the *width* expression.

filter

The method to use to scale the image. Choosing *Default* uses the *sinc* filter when scaling the image down, and the *mitchell* filter when scaling the image up. For more information, see Chapter 28, "Filters."

subPixel
Turns on quality control.
- 0 = low quality
- 1 = high quality

If the new width or height is not an integer (either because it was set that way, or because of a proxy scale), you have a choice to snap to the closest integer (subpixel off) or generate transparent pixels for the last row and column (subpixel on).

Zoom

The *Zoom* node resizes the image to a given resolution.

Parameters

This node displays the following controls in the Parameters tab:

xScale, yScale
The scaling factor, for example, .5 = half resolution, 2 = twice the resolution.

filter
The method to use to scale the image. For more information, see Chapter 28, "Filters."

subPixel
Turns on quality control.
- 0 = low quality
- 1 = high quality

If the new width or height is not an integer (either because it was set that way, or because of a proxy scale), you have a choice to snap to the closest integer (subpixel off) or generate transparent pixels for the last row and column (subpixel on).

Remastering to a Different Resolution With Proxies

You can remaster your scene's resolution using proxy images. As an example, you can load NTSC and PAL images simultaneously, with one set as a proxy image. With the proper scaling factors, the scene can be reset to the other resolution by switching the proxy set.

For more information on working with proxies and high-resolution images, see Chapter 3, "Adding Media, Retiming, and Remastering."

Cropping Functions

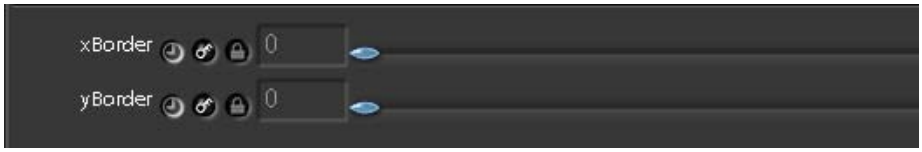
This section describes several nodes you can use to crop your images. *Window*, *Viewport*, and *Crop* are located in the Transform tab, and *AddBorders* is located in the Other tab.

AddBorders

The *AddBorders* node is similar to a *Crop*, except it adds an equal amount of space to the left and right sides, or to the top and bottom sides. It is located in the Other tab because of its infrequent use.

Parameters

This node displays the following controls in the Parameters tab:



xBorder

The number of added pixels to the left and right border.

yBorder

The number of added pixels to the bottom and top border.

Crop

This node crops an image by defining the lower-left corner and the upper-right corner of the crop. Since the numbers can be greater or less than the boundaries of the image, you can make the image smaller, or expand the image with a black border. A *Crop* cuts elements beyond the frame area, so if you later use another transform node (for example, *Pan*, *Move2D*, and so on) to move the image, black is brought in. This is how *Crop* differs from *Viewport*—*Viewport* does not cut off the image. *Window* is also the same command, but you set the lower-left corner, and then the X and Y resolution.

Crop is particularly helpful in that it sets a new frame area. Shake only processes nodes within the frame, so to speed up an operator you can limit its area with a *Crop*. This is essentially what is done with the *Constraint* node.

Parameters

This node displays the following controls in the Parameters tab:

cropLeft

The number of pixels to crop from the left of the image. This parameter defaults to 0, the leftmost pixel of the image.

cropBottom

The number of pixels to crop from the bottom of the image. This parameter defaults to 0, the bottommost pixel of the image.

cropRight

The number of pixels to crop from the right of the image. This parameter defaults to the *width* expression.

cropTop

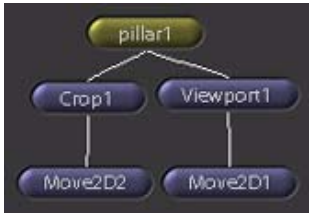
The number of pixels to crop from the top of the image. This parameter defaults to the *height* expression.

Viewport

The *Viewport* node is exactly like the *Crop* node, but it keeps the image information outside of the frame so you can perform later transformations.

Viewport Node Example

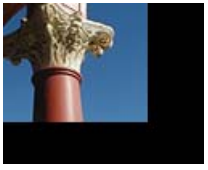
The following tree has a large input image (scaled down in the illustration) which is piped into a *Crop* node and a *Viewport* node, each with the same values. Both nodes are then piped into *Move2D* nodes with the same xPan values. The *Crop* result has black on the right edge after the pan; the *Viewport* result does not.



Node Tree

FileIn1

Crop1



Viewport1

Move2D1

Move2D2

Parameters

This node displays the following controls in the Parameters tab:

cropLeft
The number of pixels to crop from the left of the image. This parameter defaults to 0, the leftmost pixel of the image.

cropBottom
The number of pixels to crop from the bottom of the image. This parameter defaults to 0, the bottommost pixel of the image.

cropRight
The number of pixels to crop from the right of the image. This parameter defaults to the *width* expression.

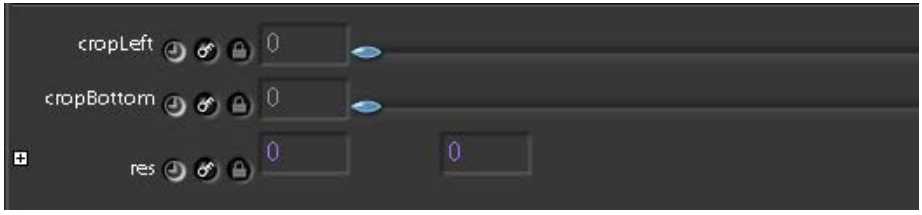
cropTop
The number of pixels to crop from the top of the image. This parameter defaults to the *height* expression.

Window

The *Window* node is exactly like the *Crop* node, but you enter the lower-left corner, and then the X and Y resolution of the image.

Parameters

This node displays the following controls in the Parameters tab:



cropLeft
The number of pixels to crop from the left of the image. This parameter defaults to 0, the leftmost pixel of the image.

cropBottom
The number of pixels to crop from the bottom of the image. This parameter defaults to 0, the bottommost pixel of the image.

xRes
The number of horizontal pixels that equals the horizontal resolution of the image. This parameter defaults to the *width* expression.

yRes
The number of vertical pixels that equals the vertical resolution of the image. This parameter defaults to the *height* expression.

Importing Video and Anamorphic Film

6

Shake provides support for nearly any video or anamorphic film format in use. This chapter covers the parameters that must be set up—and the special considerations given—for these formats.

The Basics of Processing Interlaced Video

Although Shake's origins lie in film compositing—the processing of high-resolution, *non-interlaced* (otherwise referred to as *progressive-scan*) images, Shake can also create composites using video images from nearly any format, standard definition or high definition. The individual frames of image sequences may be interlaced as well as progressive-scan, enabling Shake users on any platform to work with video clips. On Mac OS X, Shake supports QuickTime, which allows for an even wider variety of video clips to be imported from applications such as Final Cut Pro.

When you read interlaced clips into Shake, there are a variety of parameters that you must set to ensure that the image data in each field of every frame is properly processed. If these parameters are incorrectly set, the result may be undesirable motion artifacts that are not apparent on your computer screen, but that leap out at you when the exported composite is played back on a broadcast video monitor.

Preserving, Eliminating, and Creating Interlacing

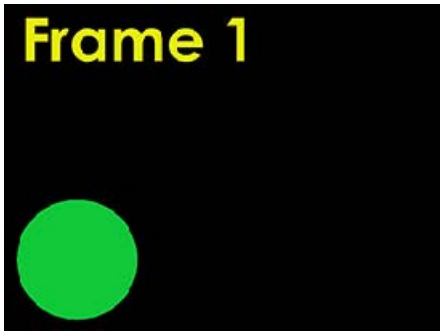
When you process interlaced video in Shake, you have the option of either preserving both fields from every frame for interlaced output, or eliminating the interlacing altogether and outputting progressive-scan clips. Additionally, you have the option of taking non-interlaced source media and turning it into an interlaced clip for use in an interlaced project.

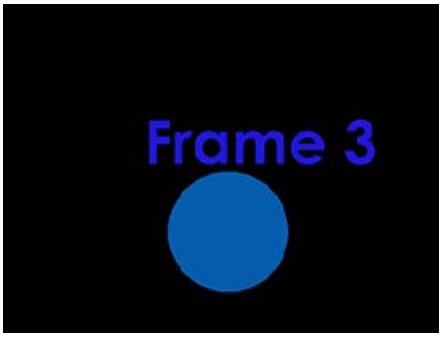
Each of these processes requires you to set up parameters within each *FileIn* node, as well as those found within your script's Globals tab, in specific ways to insure proper rendering, and to maximize the quality of the processed result.

Understanding Video Interlacing

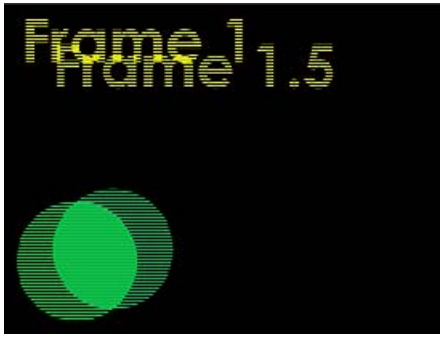
Dividing each frame of video into two fields is a technique originally developed for television broadcasting to solve a number of technical difficulties with early TV equipment. In essence, interlacing reduces the perceived strobing of 30 images playing every second on a television screen. Interlacing divides each frame into two fields, each of which contains half of the image information. Consequently, the television screen displays 60 fields each second, resulting in smoother motion.

The following example depicts an animation sequence showing frames 1 and 3 of a moving image. These are non-interlaced, full frames—each frame contains the entire image at that instant in time.





The images below depict the same frames up to (but not including) frame 3 as they appear when they're interlaced. The image on the left shows some information from frame 1, field 1 and from frame 1, field 2 (unconventionally labeled here as frame 1.5). The image on the right shows frame 2, field 1 and frame 2, field 2 (unconventionally labeled frame 2.5 for illustration purposes).



As you can see, each field contains only half of the horizontal lines of the image. If you've ever seen a still photograph of an interlaced source clip, you've probably already seen this type of image: the moving subject appears to be in two places at once, blurred because each image contains only half the scan lines.

This effect occurs because video fields are recorded one after the other, just like frames. When a moving subject is recorded using an interlaced video format, the subject is in one position when the first field is recorded, and in another position when the second field is recorded. When the video is played back, and each field is played in the correct order, the image appears normal. However, looking at both fields together as a single frame, the change in position can be seen to occur within the one frame.

The following images show field 1 and field 2 of a single frame as separate images. Notice the black lines across the images—these are the lines that are filled by the opposing field. This example clearly illustrates that each field contains only half of the total image data in a given frame.

The following illustration depicts a close-up of the interlaced frame, with both fields combined. Because of the subject's change in position from one field to the next, the fields appear offset.

Because each interlaced frame of video consists of two fields that contain half the information for that frame, each field can be thought of as a half-height version of the originating frame. Because, during playback, the television displays these images quickly, one after the other, the human eye is fooled into perceiving the image as having a higher resolution than each individual field actually possesses. To sum up, each field sacrifices quality in terms of vertical resolution (perceived as image sharpness) for the benefit of improved temporal quality (perceived as smoothness of motion).

Common Issues When Compositing Interlaced Images

In order to avoid pitfalls when compositing interlaced media, it's important to understand the peculiarities of the format. This section describes the compositing operations in Shake that are affected by improperly setting a script's interlacing parameters.

Parameter Animation Across Fields

The first problem occurs when you animate any parameter. The animation must be understood and applied to every field of video—at half-frame intervals. If you read in an interlaced clip and apply a static *Gamma*, no problems occur because both fields receive the same correction. If, however, you animate the gamma correction, you must enable field rendering in order to apply the correct gamma value to each of the two fields, in the correct order.

Transforms Applied to Fields

The second, and trickier, issue arises when you apply spatial effects with a node such as the *Blur* or a *Move2D* node. For example, panning an image up by 1 pixel in the Y axis has the inadvertent effect of reversing the two fields within that frame, because the even lines are moved to the odd field, and the odd lines are moved to the even field. The resulting motion artifacts will appear as a jittering effect because the fields are playing backwards even though the frames are still playing forwards. This would be the same as if you inverted a standard film frame sequence 1, 2, 3, 4, 5, 6 to play as 2, 1, 4, 3, 6, 5.

Another issue arises when you apply image rotation and scaling to an interlaced clip. In the following images, a rotation effect has been applied to two images, one with field rendering and one without field rendering. The right image will appear correct when played back on a broadcast monitor, because the interlaced lines are properly arrayed. The lack of clearly defined fields in the left image will cause undesirable artifacts during video playback.

No field rendering

Field rendering

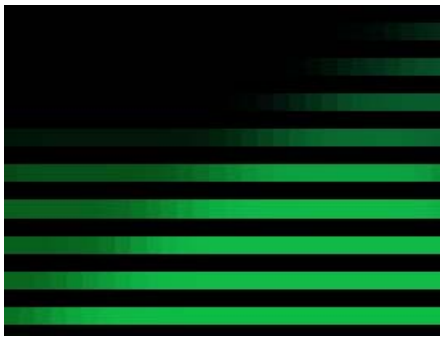
Filters and Fields

Field rendering doesn't just affect spatial transformations. The examples below depict close-ups of frame 1 (from above), without and with a *Blur* node applied. Look at the second image. When the blur effect is applied uniformly to both fields, the result is an actual loss of image data because the individual fields are improperly combined.

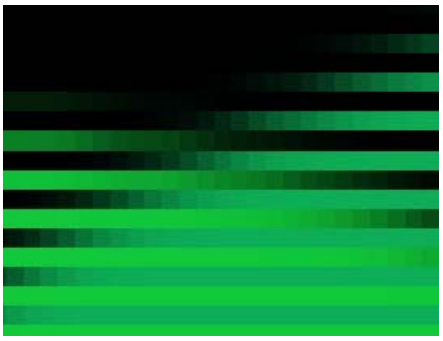
Original image

Blurred without field rendering

To illustrate what happens when fields are improperly combined, we've removed one field from the image on the left below. Notice that information from both fields intermingles due to the blur, as pixels from a different moment in time bleed into the current field. Turning on field rendering gives you the correct image, shown on the right. No image information bleeds between the two fields.



One field of improperly blurred image



Properly blurred image with field rendering

Setting Up Your Script to Use Interlaced Images

To make sure that interlaced footage is rendered properly in your script, you need to make sure that your script is set up to correctly process and view the fields and frames in your composition. This procedure involves four steps:

Step 1: Set the deInterlacing parameter of each FileIn node

In the Parameters tab of each *FileIn* node in your project, set the deInterlacing parameter (located in the Source tab) to match the field dominance of that media.

Step 2: Set the reTiming parameters of each FileIn node

In the Parameters tab of each *FileIn* node, set the reTiming parameter (located in the Timing tab) to Convert. Next, adjust the reTiming subparameters to the settings appropriate to your desired output, whether to preserve or eliminate interlacing. For more information, see "Setting the reTiming Parameters of Each FileIn Node" on page 198.

Step 3: Set the Inc (Increment) parameter in the Time Bar to 0.5

Setting the Inc parameter in the Time Bar to 0.5 allows you to view each field independently while you work on your composition. This allows you to eliminate the blurring caused by overlapping fields.

Step 4: Set the OutputFrameInterlaced and fieldRendering parameters when you're finished compositing

Once you've completed your composite and you're ready to render, turn on the OutputFrameInterlaced parameter within each *FileIn* node's Timing tab, then set the fieldRendering parameter in the renderControls subtree of the Globals tab to the same field dominance used in your other clips.

Important: Make sure you leave fieldRendering set to off until you're ready to render. Otherwise, the Viewer will display both fields of every frame together, eliminating your ability to see each individual field.

Setting the deInterlacing Parameter of Each FileIn Node

When you create a *FileIn* node to read interlaced media into your project, the first thing you should always do is to set the deInterlacing parameter within the Source tab to the correct field dominance for that media. By default, deInterlacing is turned off.

There are three field rendering settings:

- *off/0:* The image is not deinterlaced. This is the appropriate setting for progressive-frame footage, including progressive-frame video formats and scanned film frames.
- *odd/1:* Use this setting for video media with a field dominance of odd (counting from the top) first. This is generally the setting for PAL images. In Final Cut Pro, this setting is referred to as Lower (Even).
- *even/2:* Use this setting for video media with a field dominance of even. This is generally the setting for NTSC images. In Final Cut Pro, this setting is referred to as Lower (Even).

When the deInterlacing parameter of a *FileIn* node is set to either odd or even, Shake separates the two fields within each frame, placing field 1 at frame 1, and field 2 at frame 1.5. This effectively doubles the number of frames processed within your script, but keeps them within the same duration of the Time Bar. Turning on deInterlacing for each *FileIn* node ensures that all animation, transforms, and filters are properly handled by Shake's field rendering.

How to Determine Field Dominance

If you're not sure of the field dominance of your media, scrub through the first few frames in the Time Bar until you find a range of frames with an obvious interlacing artifact (generally found in frames with a fast-moving subject). Choose a field rendering setting in the Parameters tab, set the frame Inc setting in the Time Bar to 0.5, and then step through that group of frames to see if the motion looks correct. If you perceive any stuttering, that's an immediate sign that the field dominance you selected is incorrect (which results in every two fields playing backwards) and should be reversed.

If the motion of the clip is subtle and you're still not sure, it may be a good idea to render a test clip that you can output to an interlaced broadcast video monitor. Any motion artifacts should be immediately visible.

Setting the reTiming Parameters of Each FileIn Node

After you've set a *FileIn* node's deInterlace parameter to the appropriate field dominance, an optional step is to open that *FileIn* node's Timing tab and set the reTiming parameters.

Preserving Interlacing

This step is not strictly necessary if you're not removing, reversing, or adding interlacing to the source media, but will improve the processing of clips in Shake when you're making transformations to interlaced footage—even when you're preserving the original interlacing for video output.

To preserve the interlacing from the original source media:

1. Click the right side of the *FileIn* node to load its parameters into the Parameters tab.
2. Open the Timing tab.
3. Set the reTiming parameter to Convert.

   Additional parameters appear below.
4. Set the InputFrameRate to match the format of the original interlaced video media (NTSC or PAL).
5. Open the InputFrameRate subtree, and turn on InputFrameInterlaced.

   The InputFrameDominance defaults to the same setting as the deInterlacing parameter in the Source tab.

6 Set the OutputFrameRate to match the InputFrameRate parameter.

7 While you're working in Shake, leave the OutputFrameInterlaced parameter in the OutputFrameRate subtree turned off.

Once you've finished and you're ready to render the resulting composite from your script as an interlaced image sequence, turn OutputFrameInterlaced on.

Removing Interlacing

If you're reading in interlaced media with the intention of rendering out a non-interlaced result, or if you're adding an interlaced shot to other, non-interlaced media, you can permanently remove the interlacing, recreating progressive frames from the original field information.

Note: This is a much higher-quality method than the *DeInterlace* node found in previous versions of Shake.

To remove interlacing from the original source media:

1 With a *FileIn* node's parameters loaded into the Parameters tab, open the Timing tab.

2 Set the reTiming parameter to Convert.

Additional parameters appear below.

3 Set the InputFrameRate parameter to match the format of the original interlaced video media (NTSC or PAL).

4 Open the InputFrameRate subtree, and turn on InputFrameInterlaced.

The InputFrameDominance defaults to the same setting as the deInterlacing parameter in the Source tab.

5 Set the OutputFrameRate to the desired output frame rate, either to match the media's original frame rate, or to convert the media to another format.

In this example, the media is being converted to 24p (progressive-scan).

6 In the OutputFrameRate subtree, turn off the OutputFrameInterlaced button.

Creating Interlacing From Non-Interlaced Source Media

You can also use the reTiming parameters to interlace previously non-interlaced media.

To create interlaced output from non-interlaced source media:

1 With a *FileIn* node's parameters loaded, open the Timing tab.

2 Set the reTiming parameter to Convert.

Additional parameters appear below.

3 Set the InputFrameRate parameter to match the format of the original media.

4 Open the InputFrameRate subtree, and make sure that InputFrameInterlaced is turned off.

5 Set the OutputFrameRate to the desired video format (NTSC or PAL).

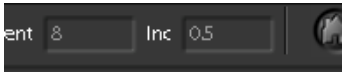
6 Open the OutputFrameRate subtree and do the following:

a Turn on the OutputFrameInterlaced parameter.

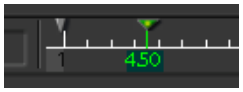
b Set the OutputFrameDominance parameter to the desired field dominance.

Displaying Individual Fields in the Viewer

In order to be able to see each individual field for purposes of paint or rotoscoping, you must set the Inc (Increment) parameter in the Time Bar to 0.5.



With Inc set to 0.5, the playhead moves in half-frame increments as you scrub through the Time Bar. When you use the arrow keys to move back and forth in the Time Bar, you'll actually be moving from field to field. The first field is displayed when the playhead is on whole frames, and the second field is displayed at every frame and a half.

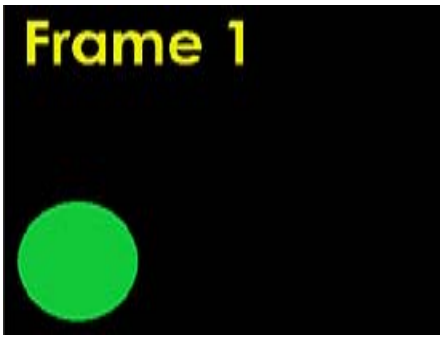


When Shake deinterlaces the media in your script, the even and odd fields displayed in the Viewer are actually interpolated to fill in the gaps where the lines of the opposing field were previously pulled out. This makes it easier for you to work with the individual fields, and also improves Shake's overall processing quality. For example, if you only displayed the actual lines found in each video field, the images would look something like this:

Field 1 appears in frame 1.

Field 2 appears in frame 1.5.

Setting the deInterlacing parameter for each *FileIn* node not only separates each field internally to Shake, but it sets the Viewer to display each field with interpolated lines of resolution added, so that each field appears as a complete image. The default interpolation quality is somewhat low, but is fast to work with. To improve the display and processing quality of individual fields, see "Setting the reTiming Parameters of Each FileIn Node" on page 198.



De-interlaced field 1

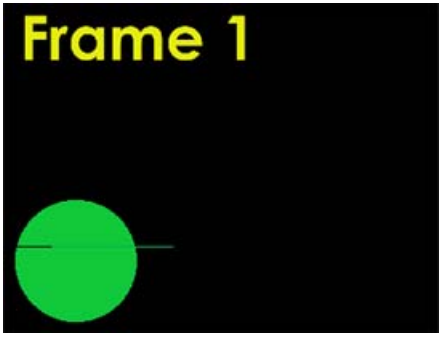
De-interlaced field 2

Important: While you're working on your composition, you should leave the fieldRendering parameter in the renderControls subtree of the Globals tab turned off. Otherwise, the Viewer simultaneously displays both fields for each frame regardless of the Inc setting in the Time Bar. Later, when you're ready to render your composition, you must then turn fieldRendering on to output interlaced media.

Zooming In on Interlaced Images in the Viewer

Zooming in on interlaced images in the Viewer can result in some undesirable artifacts. These artifacts will only appear on your computer screen, but will not appear in the final rendered output. These artifacts disappear when you set the Viewer back to a 1:1 viewing ratio (move the pointer over the Viewer area, and press Home).

Note: You can also click the Home button in the Viewer to reset the ratio to 1:1.



Viewer artifact example

Exporting Field Interlaced Footage

If you're working on media that will be output to an interlaced video format, you have to set one additional global parameter. Once you've finished your composite, you need to set the fieldRendering parameter in the renderControls subtree of the Globals tab to the appropriate field dominance.

Field Rendering Settings

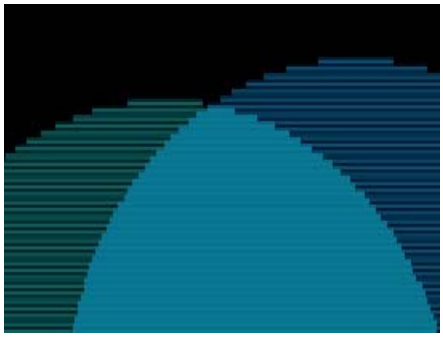
By default, fieldRendering is set to *off*.

There are three field rendering settings:

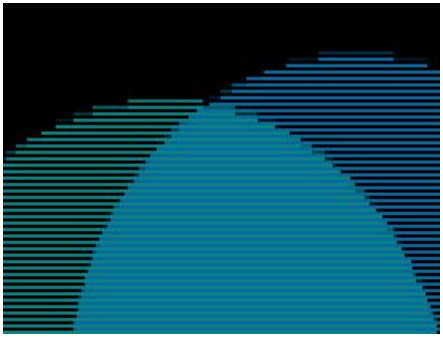
- *off/0:* Field rendering is turned off. This is the appropriate setting for progressive-frame media, including progressive-scan video formats and scanned film frames.
- *odd/1:* Field rendering with the odd field (counting from the top) first. This is generally the setting for PAL images.
- *even/2:* Field rendering with the even field first. This is generally the setting for NTSC images.

When fieldRendering is set to odd or even, Shake re-interlaces both fields of each frame when rendering out the final media.

In the following example, the image has been resized from 640 x 480 to 720 x 486. The image on the left has field rendering off, while the image on the right has field rendering on. In the left example, resizing the image without removing interlacing first has resulted in undesirable inter-field bleeding (in other words, the lines in the alternating fields have been enlarged and distorted, and no longer line up). The right example benefits from having been de-interlaced *before* image resizing. By reapplying interlacing *after* the resize transformation, field-to-field continuity has been preserved (in other words, the field lines line up properly).



Resize without field rendering



Resize with field rendering

JPEGs and Fields

Using the JPEG output for field rendering is not recommended. The compression bleeds information from one field into the other, which results in unwanted artifacts.

Integrating Interlaced and Non-Interlaced Footage

If you need to integrate interlaced and non-interlaced media within the same script, the best strategy is to decide which format the video needs to be in—interlaced or non-interlaced—and convert all media in your script to that format.

You can use the Convert setting of the reTiming parameter in the Timing tab of each *FileIn* node's parameters to do this. For more information, see "Setting the reTiming Parameters of Each FileIn Node" on page 198.

Video Functions

Shake has several other video-oriented functions. When using these features, make sure that field rendering is off, because field-rendering options may interfere with the desired result. These functions include the following:

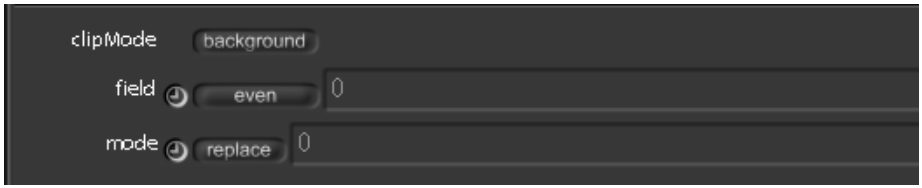
Location	Function	Description
Globals tab	*timecodeMode*	Sets the timecode format displayed in the Time Bar.
Time Bar	T on keyboard	Toggles timecode/frame display.
Image tab	*FileIn*	Has de-interlacing, as well as pulldown/pullup capabilities under the Timing subtree. See the "*Using the FileIn (SFileIn) Node*" on page 110 for more information.
Color tab	*VideoSafe*	Limits your colors to video-legal ranges. See "*VideoSafe*" on page 208.
Layer tab	*Constraint*	Limits effects by certain criteria, either zone, change tolerance, channel, or field. Naturally, field is of interest here. You can affect a single field with this node. This is generally done with field rendering off. See Chapter 16, "Compositing With Layer Nodes," for more information.
Layer tab	*Interlace*	Interlaces two images, pulling one field from one image, and the second field from the other image. You can select field dominance. This is generally done with field rendering off. See "*Interlace*" on page 205.
Other tab	*DeInterlace*	Retains one field from an image and creates the other field from it. You have three choices as to how this is done. The height of the image remains the same. This is generally done with field rendering off. See "*DeInterlace*" on page 206.
Other tab	*Field*	Strips out one field, turning the image into a half-height image. Generally done with field rendering off. See "*Field*" on page 207.
Other tab	*Swapfields*	Switches the even and odd fields of an image when fieldRendering is off. To do this when fieldRendering is on, just switch from odd to even or even to odd. Generally done with field rendering off. See "*SwapFields*" on page 207.

Interlace

Located in the Layer tab, this node interlaces two images. You can control field dominance, whether the input images are themselves separate field images (for example, half Y resolution), or if the fields are extracted from every other line.

Parameters

This node displays the following controls in the Parameters tab:



clipMode

Toggles between using the foreground (0) or background (1) images to define the resolution.

field

Specifies which field the first image uses.

- 0 = even field
- 1 = odd field

mode

Tells Shake if the input image is the same height as the result image.

- 0 = replace. Takes every other field from the input images; input images have the same height as the result.
- 1 = merge. Takes the entire input image; input images are half the result image height.

DeInterlace

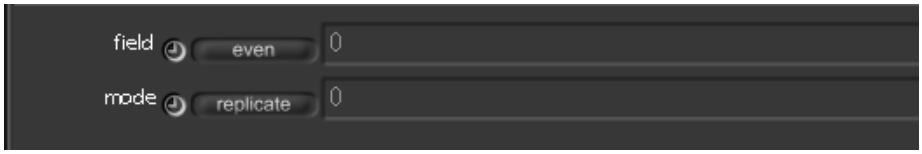
Located in the Other tab, this node is unlike the de-interlacing parameter available in the *FileIn* node because it is designed to *permanently* deinterlace a clip, discarding the upper or lower field. The *DeInterlace* node has three different modes you can use to replace the alternating field lines that are removed:

- *replicate:* Replaces one field with the other by duplicating the remaining fields.
- *interpolate:* Replaces a field with an average of the fields above and below it.
- *blur:* Replaces a field with an average of the fields above and below it, combined with the original field itself. More precisely, the field is replaced with a mix of 50 percent of itself, 25 percent of the field above, and 25 percent of the field below.

Note: The Convert option in a *FileIn* node's Timing tab provides de-interlacing options superior to those found in this node. For more information, see "Setting Up Your Script to Use Interlaced Images" on page 196.

Parameters

This node displays the following controls in the Parameters tab:



field

The field that is retained.

- 0 = even field
- 1 = odd field

mode

The mode in which the removed field is replaced (see above).

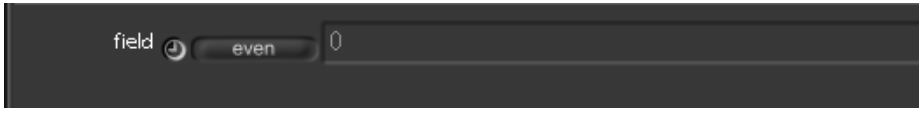
- 0 = replicate
- 1 = interpolate
- 2 = blur

Field

Located in the Other tab, this node extracts the even or the odd field of the image, returning an image of half the Y resolution.

Parameters

This node displays the following controls in the Parameters tab:



field

Specifies the field that is extracted.

- 0 = even field
- 1 = odd field

SwapFields

Located in the Other tab, this node switches the even and odd fields of an image.

There are no parameters in the *SwapFields* node.

VideoSafe

Located in the Color tab, this node clips "illegal" video values. As such, it is generally placed at the end of a composite. You can set the node for NTSC or PAL video, based on luminance or saturation. There is also an example (in the videoGamma subtree) of a conditional statement that toggles between 2.8 (NTSC) and 2.2 (PAL). Generally, these values are not touched.

Parameters

The *VideoSafe* node displays the following controls in the Parameters tab:

videoType

Toggles between NTSC and PAL.

- 0 = NTSC
- 1 = PAL

processingType

Determines whether the VideoSafe node affects the luminance or chrominance (saturation) of the image.

- 0 = luminance-based calculation
- 1 = saturation-based calculation

chromaRange

The pseudo-percentage of the clip, as specified by the actual video hardware.

compositeRange

The pseudo-percentage of the clip, as specified by the actual video hardware.

videoGamma

The gamma basis, based upon the videoType. NTSC uses a gamma value of 2.2 and PAL uses a gamma value of 2.8. This parameter defaults to the following expression, which links it to the videoType parameter:

```
videoType ? 2.8 : 2.2
```

The result of this expression is that if videoType is not zero (in other words, videoType is set to PAL), videoGamma is set to 2.8. If videoType is set to 1, videoGamma is set to 2.2.

For more information about how expressions work, see Chapter 30, "Installing and Creating Macros," on page 905.

About Aspect Ratios and Nonsquare Pixels

Shake has several controls in the Globals tab to help you work with nonsquare pixel images. These images are typically video images, or anamorphic film images. Different controls are used for the two types, due to the nature of the data that is manipulated.

- In order to avoid mixing up each frame's field information, nonsquare pixel distortion is corrected by extending the image horizontally (in the X direction) and not vertically.
- For anamorphic film plates, because the primary concern is the amount of data that is calculated, the image is vertically squeezed. This has the added benefit of reducing frame size of the image, which lets Shake process your script faster. In the case of CinemaScope, this not only corrects the anamorphic distortion, but also speeds Shake's interactivity by a factor of two.

When you correct nonsquare pixel images, you need to know the aspect ratio of the image in order to see the transformations and corrections without distortion. For this discussion of different aspect ratios, film anamorphic plates are used to illustrate how to work with such frames. Although this solution applies specifically to film plates, the principles and problems are similar for anamorphic and non-anamorphic video, although the aspect ratios vary depending on the video format.

What is Anamorphic Video?

Anamorphic processes such as CinemaScope create film frames that allow for an extremely wide-screen aspect ratio when projected. This is accomplished by filming with a special lens that horizontally squeezes the incoming image by half horizontally, in order to fit twice the image into a conventional frame of film.

When viewed without correction of any kind, each frame appears very thin on the physical negative. When the film is projected in the theater, a reverse lens is used to expand the image by 2:1 horizontally, which returns the image to its original (wide) format. It is important to understand that the recorded image is only widescreen in two places—in front of the lens when filming, and on the projection screen. During the postproduction process, you are usually working with the squeezed image.

This is a fundamental principle when compositing anamorphically squeezed elements—the actual image should never actually be scaled, but in order to work on the image, you still need to see the results as they will look in widescreen. Shake has specific parameters that allow you to preserve the original anamorphic data, while viewing the frame at the proper unsqueezed ratio for purposes of layering other images, rotoscoping, and painting.

Anamorphic Examples

The following screenshot depicts an anamorphic frame. The image resolution is 914 x 778, or half of a standard 1828 x 1556 anamorphic plate. You can see from the shape of the circle that the image is squeezed along the X axis.

Properly Viewing Squeezed Images

There are two ways to view this image in its unsqueezed, widescreen proportions. You might be tempted to change the resolution of the image with a *Zoom* or *Resize* node, but this would be the wrong thing to do. Zooming the image horizontally (by the Y axis) down by 2 (or up by 2 in the X axis), compositing, and then zooming it back to its squeezed proportions would result in an unacceptable and unnecessary loss of quality.

The correct way to work with an anamorphic image is to modify either the Viewer's aspect ratio, or the script's proxyRatio. The script proxyRatio is better for film elements, and the Viewer aspect ratio is better for video elements.

- *Video:* To change the Viewer aspect ratio, expand the format subtree in the Globals tab and set the defaultViewerAspectRatio parameter to 2 (the anamorphic ratio is 2:1 for this film plate).

  For video, use the appropriate ratio, which is listed in the Table of Common Aspect Ratios at the end of this chapter. Doing this renders the image at full resolution, and then doubles the Viewer's width. Because you are modifying only the Viewer parameters, this has no effect on your output resolution, and images take exactly the same amount of time to render.

The only speed hit is in the interactivity to adjust the viewed frame. This is the parameter you should use when dealing with video clips, since you change the X axis and leave the Y axis, which contains the sensitive field information, alone.

- *Film:* With film media, you should set the proxyRatio (in the useProxy subtree of the Globals tab) to .5 (1/2). Use of the proxy system reduces your render time for interactive tests by half. Unlike viewerAspectRatio, this procedure halves the Y value, rather than doubling the X value. However, the proxy system affects your output files, so be sure to set the proxyRatio back to 1 when you render your images to disk. Remember, you want to send squeezed files to the film recorder.

Following the above guidelines for this anamorphic film plate, open the useProxy subtree and enter a proxyRatio of .5 to correct the squeezed element.

Node Aspect Ratio and the defaultAspect Parameter

Certain functions need to be aware of the current aspect ratio—specifically functions dealing with circles or rotation. For example, if you apply a *Rotate* node to an anamorphic plate, the image is distorted.

The *Rotate* node has an aspectRatio parameter. Set the parameter to .5, and the rotation is no longer distorted.

The *RGrad* node is backward from the other nodes. The aspectRatio for this should be 1/defaultAspect (what it uses as its creation rule). Here, an *RGrad* with an aspectRatio of 1 is composited on the image.

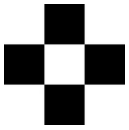
Since it is distorted, change the aspectRatio of the RGrad to 2 and the world is beautiful.

Compositing Square Pixel Images With Squeezed Images

In this example, the following image represents a square pixel frame, which is typical of 3D-rendered (CG) images.

When composited over the image, there is distortion because of the proxyRatio.

There are two options to correct this. You can scale the X parameter by half, or increase the Y parameter by two. The first option ensures the highest quality, but means you have to render the original CG element at twice the resolution of the second option. In this example, the image is scaled in the Y parameter by 2 with either a Transform–*Scale* or *Zoom* node:



Inheritance of the defaultAspect Parameter for Individual Nodes

If you're working on a nonsquare pixel video composite, it's important that you correctly set the defaultAspect parameter in the format subtree of the Globals tab *before* you begin working on your composition. The aspectRatio parameter in *Rotate* and other nodes inherits this parameter automatically whenever the nodes are created.

Changing the defaultAspect after the fact does not change any nodes you've already created, but it only affects nodes that are created after you set the defaultAspect.

The following nodes inherit the defaultAspect parameter when they're created:

- *ApplyFilter*
- *Checker*
- *CameraShake*
- *Blur*
- *Defocus*
- *DilateErode*
- *EdgeDetect*
- *FilmGrain*
- *Grain*
- *IBlur*
- *IDefocus*
- *IDilateErode*
- *IDisplace*
- *IRBlur*

- *ISharpen*
- *PercentBlur*
- *Pixelize*
- *Sharpen*
- *RBlur*
- *Sharpen*
- *AddText*
- *MatchMove*
- *Stabilize*
- *Text*
- *PinCushion*
- *Randomize*
- *Turbulate*
- *AddShadow*

3D Software Renders

If your software allows, render your scene with the proper aspect ratio. This ensures the highest quality in your composite.

Tuning Parameters in Squeezed Space

In addition to *Rotate* and *RGrad*, other nodes should be tuned with a squeezed aspect ratio. In the following example, a *Blur* node is applied to the image.

Because the default yPixel value is set to the xPixel value, you get twice the blur on the X parameter in the squeezed space. To correct this, double the Y value (or halve the X value) with an expression in the yPixels parameter:

*xPixels*2*

The blur now looks proportionately correct.

Rendering Squeezed Images

Once your composite is complete, reset the proxyRatio (in the Globals tab) to 1, and render. Do not change any other parameters. The resulting image appears squeezed on the X axis, but this distortion is corrected during the film projection process in the theater.

Although you worked on the image in a squeezed state, all elements are properly positioned when the proxyRatio is returned to 1. Use of the proxy system temporarily squeezes the image down and then stretches it back out, thereby maintaining the pixel data.

Handling Video Elements

Nearly all standard-definition video formats use nonsquare pixels, which results in a squeezed image similar to that caused by anamorphic distortion. In addition, each video format has its own aspect ratio. Using the proxyRatio parameter is not recommended for video elements because proxyRatio squeezes the image along the Y axis, which causes problems with field separation.

The correct way to account for video pixel ratios is to use the viewerAspectRatio parameter (within the format subtree of the Globals tab) to set the aspect ratio of the Viewer, leaving the fields of your video frames untouched. This also only affects the Viewer—rendered images are not affected. However, Viewer playback performance may be slightly affected.

You need only to change the defaultAspect for proper rendering. Unlike using the proxyRatio method, you set defaultAspect to 1/YourVideoAspectRatio. For example, PAL HD uses 1.422 as its aspect ratio. Therefore, set viewerAspectRatio to 1.422, and defaultAspect to 1/1.422. Shake resolves the expression of 1 divided by 1.422 to become 0.703235. All other principles of image manipulation apply.

Preset Formats

The format pop-up menu in the Globals tab provides a number of preset formats. You can also create your own formats.

Table of Common Aspect Ratios

Below is a table of common aspect ratios, containing the values you should assign to specific parameters. For nodes, the aspectRatio parameter is taken from the defaultAspect value at the time the node is created. It is not changed if you later change the defaultAspect.

Additionally, *RGrad* is the inverse of the other aspectRatio parameters, for example, 1/defaultAspect, which accounts for its own column in the table. Initially setting the defaultAspect guarantees that all nodes automatically get the proper aspect ratio.

Finally, follow the guidelines in Chapter 14, "Customizing Shake," and save your interface settings to preserve default values for these parameters. For more information, see "Customizing Interface Controls in Shake" on page 359.

Format	aspect Ratio	proxy Ratio	Viewer Aspect Ratio	default Aspect	aspect Ratio (common nodes)	aspect Ratio (RGrad)
2:1 Anamorphic Film	2	.5	NA	.5	.5	2
4:3 NTSC D1 720 x 486, 720 x 480	.9	NA	.9	1/.9 = 1.1111	1.11111	.9
16:9 NTSC 720 x 486	1.2	NA	1.2	1/1.2 = .83333	.8333	1.2
4:3 PAL D1 720 x 576	1.066	NA	1.06	1/1.066 = 0.938086	0.938086	1.066
16:9 PAL 720 x 576	1.422	NA	1.422	1/1.422	.703235	1.422

Using the Node View

7

The Node View is the heart of Shake's graphical compositing interface. This chapter covers all aspects of navigating, customizing, and organizing nodes using this powerful tool.

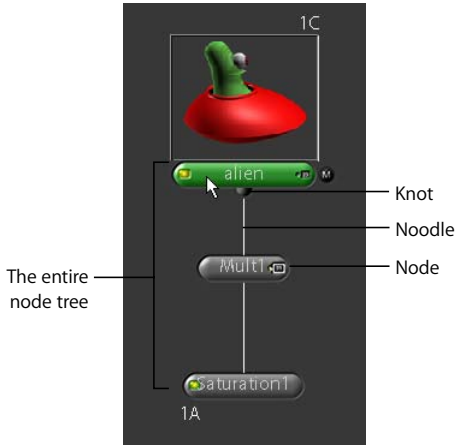
About Node-Based Compositing

The Node View in Shake displays all of the nodes that are used in a script. This amalgamation of all the nodes used in a script is referred to as the *node tree*.

Each *node* in the tree performs a specific function, and is connected to other nodes by lines referred to as *noodles*. The noodles that stretch between each node represent the flow of image data from node to node.

Noodles are connected to input *knots* on each node. The output knot of one node is usually attached to the input knot of the next node down in the node tree. In this way, image data is passed top-down, from one node to the next. Thus, the image is modified bit by bit until the final result is achieved.

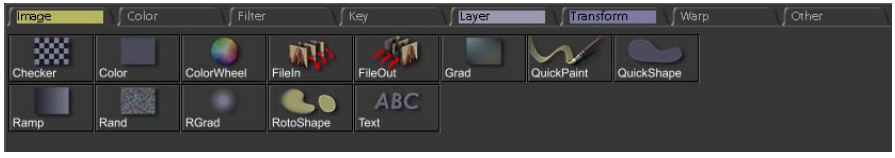
Note: Knots are only visible when the pointer is positioned over a node.



This node-based approach has many advantages. By expressing the entire compositing task as a big flowchart, of sorts, the flow of image data is easy to keep track of. Graphical manipulation of the individual nodes in the tree also lets you make changes by quickly turning on and off different functions in the composite, adding and removing nodes where necessary.

Where Do Nodes Come From?

All of the nodes available in Shake are available in the Tool tabs, found in the lower-left area of the interface.



You can also access the full list of nodes by right-clicking anywhere within the Node View, and choosing a node from the Nodes submenu of the shortcut menu.

Shake on Mac OS X also provides a Tools menu in the menu bar, with submenus for each Tool tab.

Clicking a node in the Tool tabs or choosing a node from the Node View shortcut menu creates that node in the Node View. For more information about creating nodes, see "Creating Nodes" on page 226.

Navigating in the Node View

Every effect in Shake is created by an individual node that has been inserted into the node tree, and each node has its own specific function and parameters. As you build progressively larger node trees, you'll find yourself spending more and more time navigating around the Node View as you make adjustments to nodes at various levels of the node tree.

As with all the other areas of the Shake interface, you can pan and zoom in the Node View to navigate around your node tree.

To pan in the Node View:

- Press the middle mouse button or press Option-click or Alt-click and drag.

To zoom into or out of the Node View, do one of the following:

- Press Control-Option-click or Control-Alt-click and move the mouse left to zoom out, or right to zoom in.
- Press the + or - key.

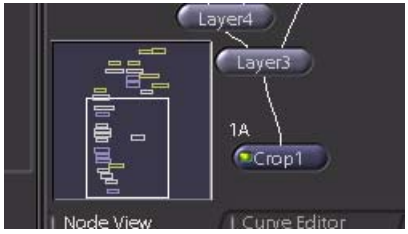
The Node View Overview

A dynamic overview available in the Node View is a helpful navigation tool, especially for complex node trees.

To toggle the overview on and off:

- Press O to display the overview. Press O again to hide it.

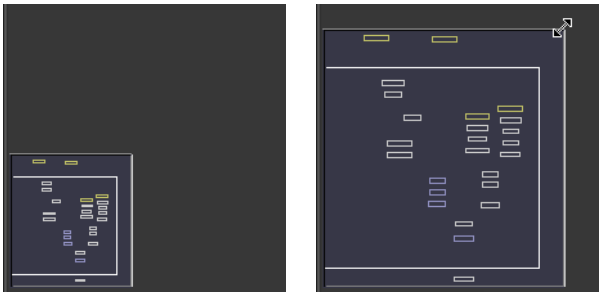
Drag within the overview to pan across the entire node tree.



The overview can be resized, making it easier to see.

To resize the node overview:

- Drag the upper-right corner, the top, or the right of the overview.



Default size After resizing

Favorite Views

If you've assembled an exceptionally large and complex node tree, you can navigate to specific areas of your node tree and save that position of the Node View as a Favorite View. This is mainly useful for saving the position of a collection of nodes that you'll be adjusting frequently.

Later on, when you're looking at a different part of the tree and you realize you need to adjust some of the nodes at one of your Favorite Views, you can instantly jump back to that part of the tree. You can save and recall up to five favorite views. For more information, see "Saving Favorite Views" on page 28.

To define a Favorite View:

1. Pan to a position in the Node View that contains the region you want to save as a Favorite View. If necessary, adjust the zoom level to encompass the area that you want to include.

   Note: You can optionally recall the state of the nodes—in other words, which ones are currently loaded into the Viewer and Parameters tabs.

2. To save a Favorite View, move the pointer over that quadrant and do one of the following:

   - Right-click anywhere within the Node View, then choose Favorite Views > View N > Save (where N is one of the five favorite views you can save) from the shortcut menu.
   - Press Shift-F1-5, where F1, F2, F3, F4, and F5 correspond to each of the Favorite Views.

To restore the framing of a Favorite View, do one of the following:

- Right-click in the Node View, then choose Favorite Views > View N > Restore Framing (where N is one of the five Favorite Views you can save) from the shortcut menu.

- Move the pointer into the Node View, and press F1-5, where F1, F2, F3, F4, and F5 correspond to each of the Favorite Views.

  That quadrant is set to the originally saved position and zoom level.

To restore the framing and state of a Favorite View, do one of the following:

- Right-click in the Viewer, Node View, or Curve Editor, then choose Favorite Views > View N > Restore Framing & State (where N is one of the Five Favorite views you can save) from the shortcut menu.
- Move the pointer into the Node View, and press Option-F1-5 or Alt-F1-5, where F1, F2, F3, F4, and F5 correspond to each of the Favorite Views.

  The nodes that were currently loaded into the Viewer and Parameters tabs are reloaded.

Using the Enhanced Node View

The enhancedNodeView subtree in the Globals tab provides additional display options for the Node View. These options can be turned on all together, or individually, to make it easy to spot image bit-depth at different parts of the tree, animated nodes, node concatenation, and expressions linking one node to another.

Unlike most other guiControls parameters, which are either off or on, these parameters—showTimeDependency, showExpressionLinks, showConcatentationLinks, and noodleColorCoding—have three states. This allows you to toggle all three parameters using the enhancedNodeView command from within the Node View. The three states are:

- *off:* Always off, regardless of whether or not enhancedNodeView is turned on.
- *on:* Always on, regardless of whether or not enhancedNodeView is turned on.
- *enhanced:* On when enhancedNodeView is on, and off when enhancedNodeView is off.

To toggle enhanced Node View off and on:

1 Before turning on enhanced Node View, make sure the proper subparameters are turned on in the enhancedNodeView subtree of the Globals tab.

2 Move the pointer over the Node View, and do one of the following:

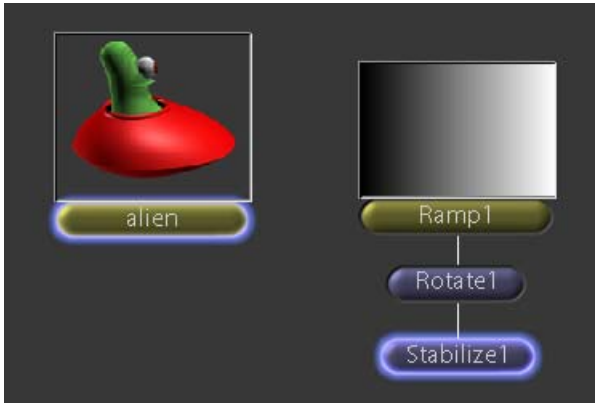
- Right-click, then choose Enhanced Node View from the shortcut menu.
- Press Control-E.
- Open the Globals tab, then click enhancedNodeView.

Enhanced Node View Parameters

There are seven parameters in the enhancedNodeView subtree.

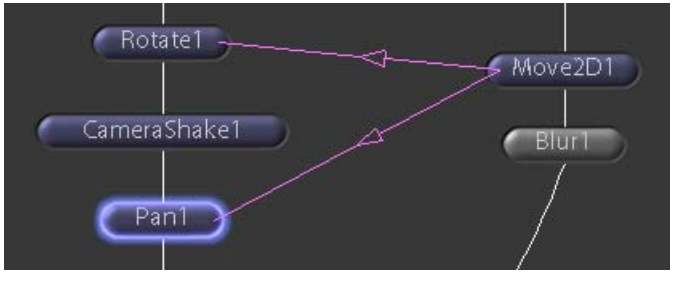
showTimeDependency

This parameter, when turned on, draws a bluish glow around nodes that are animated. This includes nodes that consist of *FileIn* nodes that reference a QuickTime movie or multiple-image sequence, nodes containing keyframed parameters, or nodes utilizing expressions that change a parameter's value over time. In the following screenshot, the alien *FileIn* node is highlighted because it is a multi-frame animation. The *Stabilize1* node is highlighted because it contains motion-tracking keyframes. The *Ramp1* node is not highlighted because it's only a single image, and the *Rotate1* node is not animated.



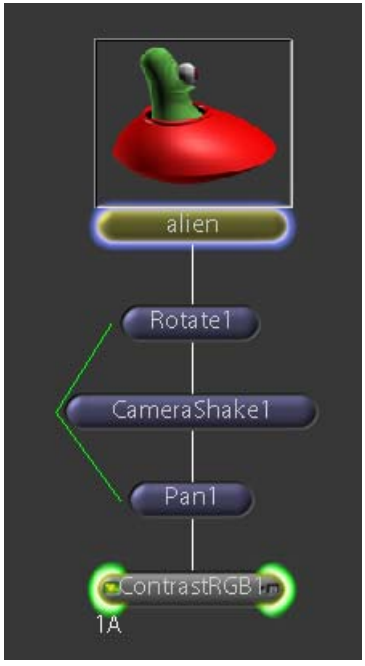
showExpressionLinks

Turning this parameter on draws a light purple line connecting nodes that use expressions to reference other nodes. An arrow pointing from the linked node toward the referenced node indicates their relationship. In the following screenshot, the *Move2D1* node contains expressions that link it to both the *Rotate1* and *Pan1* nodes.



Note: When you clone a node by copying it and then pasting it with the Paste Linked command, the resulting cloned node displays an expression link arrow when showExpressionLinks is turned on.

ShowConcatenationLinks
When this parameter is turned on, a green line connects a series of nodes that concatenate. For example, three transform nodes that have been added to a node tree in sequence so that they concatenate appear linked with a green line connecting the left edge of each node. As a result, nodes that break concatenation are instantly noticeable. In the following screenshot, the *Rotate1*, *CameraShake1*, and *Pan1* nodes are concatenated.



Note: As is often repeated, node concatenation is a very good thing to take advantage of. You are encouraged to group nodes that concatenater whenever possible to improve the performance and visual quality of your output.

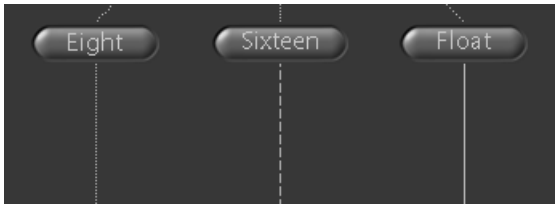
Noodle Color Coding

Turning on Shake's noodle color coding parameters provides an additional way to visually distinguish the bit depth and channel information of the image data flowing down your node tree. With color coding turned on, you can see where in the node tree the bit depth is promoted or reduced, which noodles contain RGBA channel data versus BW data, and so forth.

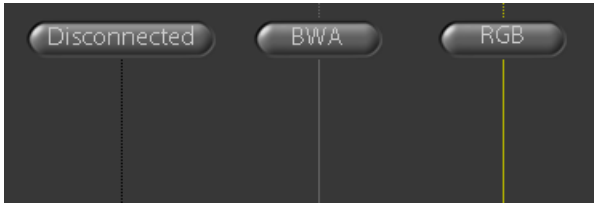
Note: The Node View redraw speed of extremely large scripts may be reduced with noodleColorCoding turned on.

There are two kinds of coding used to identify the image data that noodles represent.

stipple8Bit, stipple16Bit, stipple32Bit
The stipple pattern of a noodle indicates its bit depth. In the following screenshot, three renamed *Bytes* nodes output 8-, 16-, and 32-bit float data respectively. The stippling indicates each bit depth at a glance.



noodleColor Coding
Different combinations of color channels being propagated down the node tree are represented by different colors. In the following screenshot (showing four renamed *Reorder* nodes), the first node is disconnected from any incoming image data, while the next two are respectively propagating BWA and RGB channels. The color channels represented by each noodle are clearly distinguishable (this screenshot is viewable in color via the onscreen help).



Noodle Display Options

There are many ways you can customize the display of noodles in the Node View. If you find the numerous color-coding and stipple patterning distracting, you can also leave these options turned off.

Note: For purposes of simplicity, most of the examples in the Shake documentation have these noodle display options either turned off or left to their defaults.

Noodle Tension

The noodleTension parameter, within the guiControls subtree of the Globals tab, lets you adjust how much "slack" there is in the way noodles are drawn from knot to knot. Higher values introduce more slack, and noodles appear more curved. Lower values reduce the slack, and noodles are drawn in more of a straight line.

Customizing Noodle Color Coding

In the noodleColors subtree of the colors subtree, there are 12 different parameters for noodle color coding, corresponding to each possible combination of color channels in Shake. Each combination has a color control you can use to set up whichever color scheme makes the most sense to you.

The top parameter, noodleColor, lets you change the base color of noodles in the Node View. Noodles are white by default, but you can use the color control to change this to anything you like.

Customizing Noodle Stippling

You can customize the stipple patterns of noodles in the enhancedNodeView subtree of the Globals tab. You can also customize the colors used to identify noodle bit depth in the noodleColors subtree of the colors subtree of the Globals tab.

In the enhancedNodeView subtree, the stipple8Bit, stipple16Bit, and stipple32Bit parameters each have five different stipple patterns you can choose from, for maximum clarity.

Creating Custom Stipple Patterns

Different stipple patterns can be set in a .h preference file. Each stipple pattern is defined by a four-byte hex number that, when converted to binary, provides the pattern of the line drawn for each bit depth—each 1 corresponds to a dot, and each 0 corresponds to blank space.

For example, 0xFFFFFFFF is the hex equivalent of 1111111111, which creates a solid line. 0xF0F0F0 is the hex equivalent of 1111000011110000, which creates a dashed line.

The default settings are:

gui.nodeView.stipple8Bit = 0x33333333;
gui.nodeView.stipple16Bit = 0x0FFF0FFF;
gui.nodeView.stipple32Bit = 0xFFFFFFFF;

Creating Nodes

All effects in Shake are performed by creating and attaching nodes to one another in the Node View tab.

To add a node to the Node View:

1. To add a node to an existing tree, select a node in the tree that you want to add a node to.

   Note: If no node is selected, new nodes that are added are free-floating, not connected to anything.

2. Do one of the following:

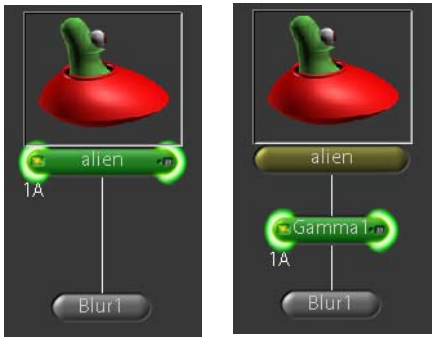
   - In Mac OS X, choose a node from the Tools menu at the top of the screen.
   - Click a node in the Tool tabs in the lower-left quadrant of the Shake interface.
   - Right-click in the Node View, then choose a node from the Node shortcut menu.

     For example, to add an *Atop* node, choose Nodes > Layer > *Atop*.

- Right-click any Tool tab to display a shortcut menu of the available node functions. The modifier keys (see below) work as well. Additionally, if you lower the Tool tabs so that only the tabs are visible, you can also access the pop-up menus with the left-mouse button (a cool trick).

Note: To add several nodes from the Tool tab shortcut menu at once, right-click the tab, then right-click each node you want to create in succession. To close the menu, left-click in the menu.

New nodes appear underneath the selected node in the tree. Adding a node to a selected node that's already connected to another node inserts the new node in between the two nodes.



You can also choose to add one type of node to every selected node in the Node View.

To add one type of node to multiple nodes in the Node View:

1 Select two or more nodes in the Node View.

2 In the Tool tabs, right-click the node you want to add, then choose Insert Multiple from the shortcut menu.

The new node is inserted after each selected node.

Selecting and Deselecting Nodes

There are numerous ways you can select nodes in the Node View.

To select one or more nodes, do one of the following:

- Click a single node to select it.
- Drag a selection box around one or more nodes in the Node View.
- Shift-click individual nodes to add them to the current selection.
- Shift and drag over nodes in the Node View to add nodes to the current selection.

  Control-click individual nodes to deselect them, leaving other nodes selected.
- Press the Control key and drag over nodes in the Node View to deselect nodes, leaving other nodes selected.
- Press Command-A or Control-A to select every node in the Node View.

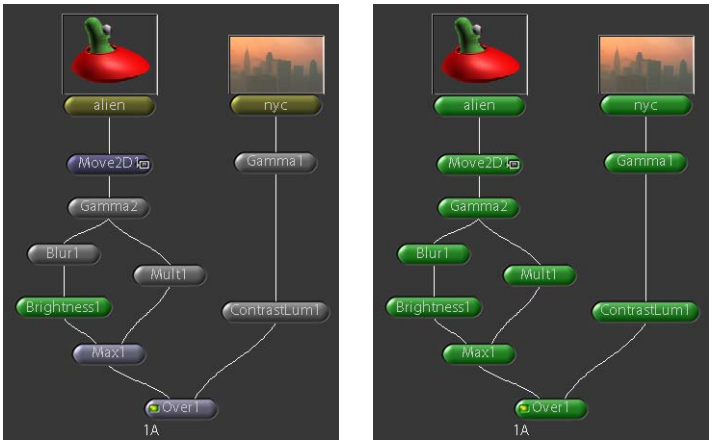
To select every node in a tree that's attached to a particular node:

1 Select the first node.

2 Do one of the following:

- Press Shift-A.

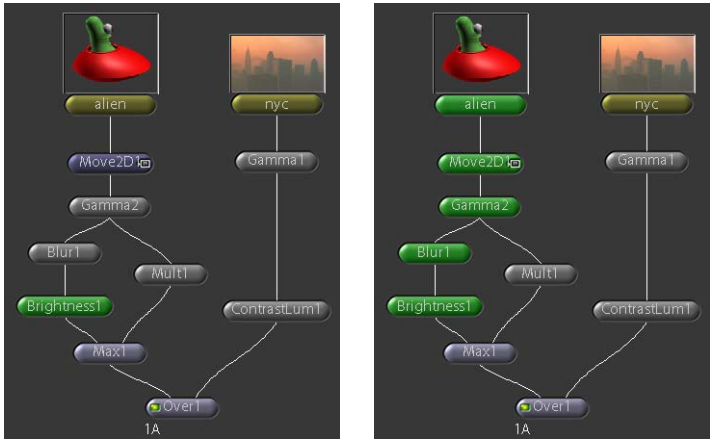
- Right-click the first node, then choose Select > Associated Nodes from the shortcut menu.



To select every node that's connected above (upstream from) a selected node, do one of the following:

- Press Shift-U.
- Right-click the selected node, then choose Select > Upstream Nodes from the shortcut menu.

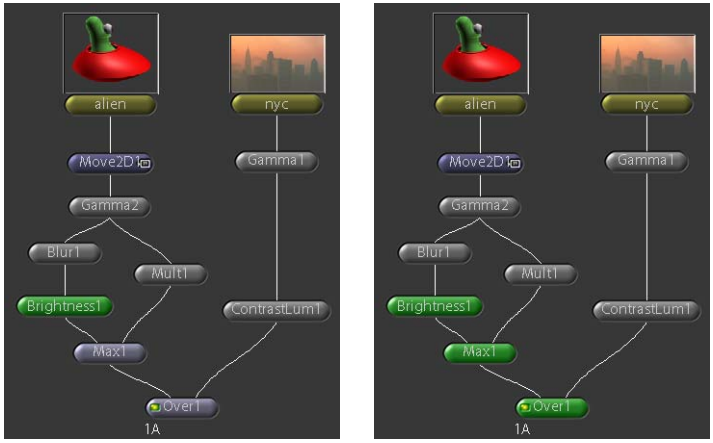
  Note: To limit the selection to only the node above the currently selected one, choose Select > Upstream 1 Level (or press Shift-Up Arrow).



To select every node that's connected below (downstream from) a selected node, do one of the following:

- Press Shift-D.
- Right-click the selected node, then choose Select > Downstream Nodes from the shortcut menu.

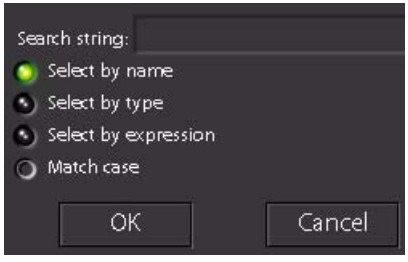
Note: To limit the selection to only the node above the currently selected one, choose Select > Upstream 1 Level (or press Shift-Down Arrow).



To select a node by its name:

1. Press Command-F or Control-F in the Node View.
2. When the Select Nodes by Name window appears, enter the name of the node you're trying to find into the Search string field.

   Nodes are selected in real time as you type. There are several options you can use to help you select the right nodes:



Function	Description
Select by name	Enter the search string, and matching nodes are immediately activated. For example, if you enter just *f*, *FileIn1* and *Fade* are selected. If you enter *fi*, just the *FileIn1* is selected.
Select by type	Selects by node type. For example, enter *Move*, and all *Move2D* and *Move3D* nodes are selected.
Select by expression	Allows you to enter an expression. For example, if you want to find all nodes with an angle parameter greater than 180, type "angle>180."
Match case	Sets case sensitivity.

To invert the selected nodes, reversing which ones are selected and deselected:

- Right-click any node, then choose Select >Invert Selection from the shortcut menu.

To deselect all nodes in the Node View:

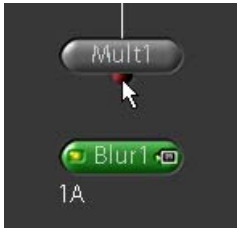
- Click an empty area of the background to deselect all nodes.

Connecting Nodes Together

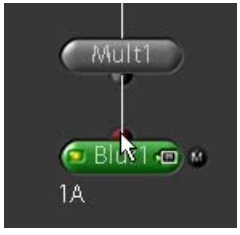
A node must be connected to the overall node tree in order to have any effect. The main method for doing this is to drag a noodle from one node's knot to another.

To connect one node to another by dragging:

1. Move the pointer over the knot of the first node you want to connect.



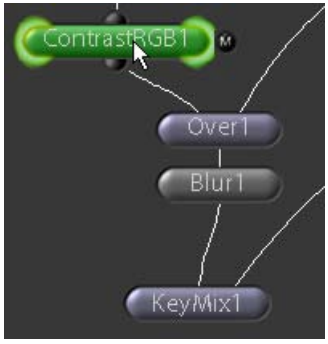
2. Click the knot, and drag the resulting noodle from the first node to the input knot of the second node.



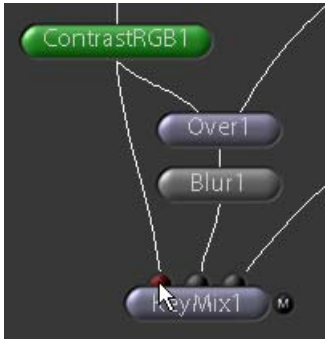
You can also connect one knot to another by Shift-clicking. This is a more convenient method to use if the two knots you want to connect are far away from one another in the Node View.

To connect one node to another by Shift-clicking:

1 Select the first node you want to connect.



2 Move the pointer over the second node you want to connect so that the knot is visible, then Shift-click the knot to connect both nodes together.

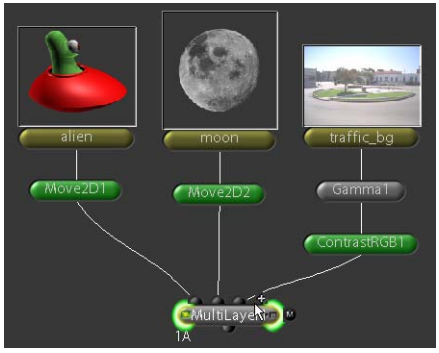


You can also use this technique to connect a group of nodes to a single multi-input node, such as the *MultiLayer, MultiPlane,* or *Select* node.

To connect several nodes to a multi-input node at once:

1 Select all of the nodes you want to connect to the multi-input node.

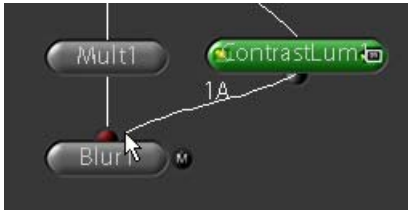
2 Shift-click the plus sign input of the multi-input node.

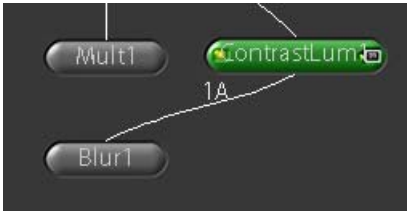


All selected nodes are connected.

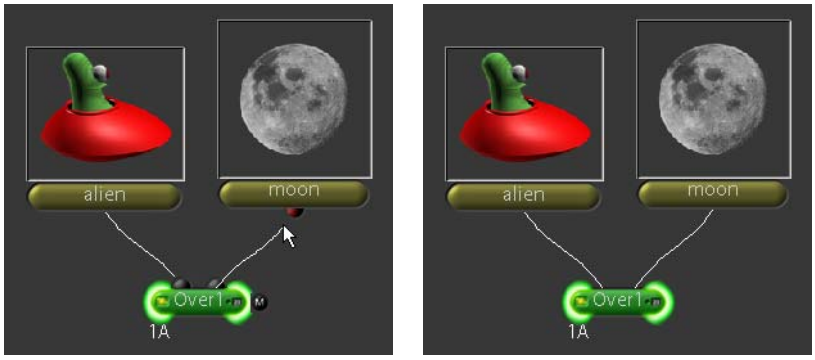
One Input, Many Outputs

Any single input knot on a node can only be connected to a single noodle. This is true even for multi-input nodes—each input knot can only be connected to a single noodle. If the node you want to connect is already attached to another noodle, the previous connection is broken, and replaced by the noodle you're dragging.

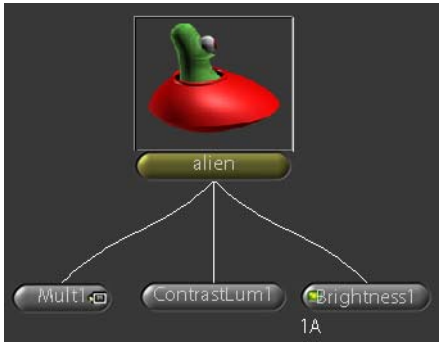




You can also drag a noodle from an input knot to the output knot of a different node. For example, you can drag a noodle from the *Over1* node to the moon node.



On the other hand, you can drag as many connections from a node's output as you want. For example, you can connect a *FileIn* node's output knot to several nodes. This creates multiple separate branches of image processing, which can be recombined later on in the tree.

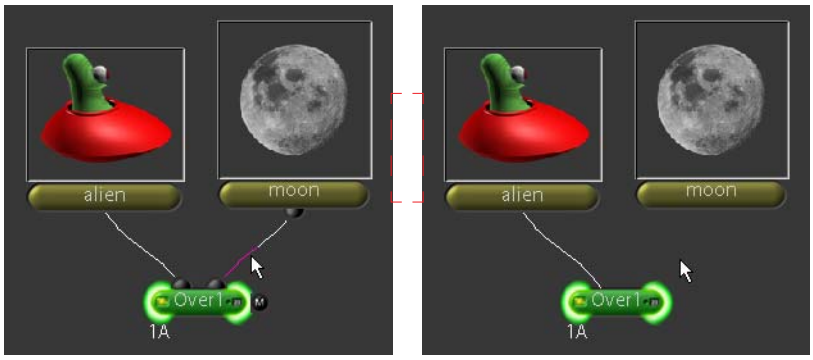


Breaking Node Connections

Node connections are broken by deleting the noodle that connects them.

To delete the connection between two nodes:

1 Select the noodle you want to delete by positioning the pointer over it so that it turns red (toward the bottom) or mustard yellow (toward the top).

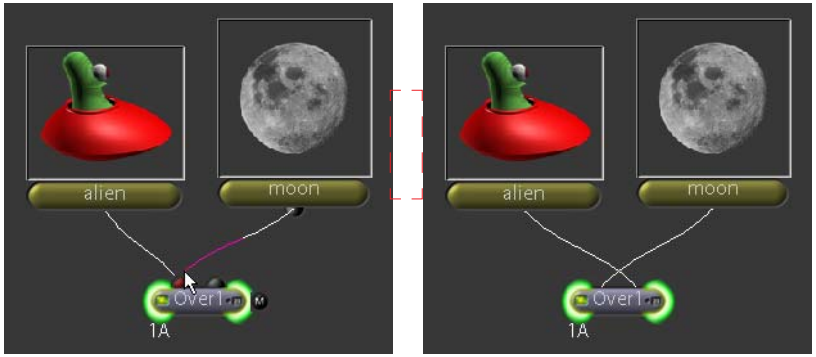


2 To delete it, do one of the following:

- Press Delete or Backspace.
- Right-click the noodle, then choose Edit > Delete from the shortcut menu.

To switch two inputs around:

- Drag the bottom half of a noodle (it turns red when selected) to another input. When you release the mouse button, the inputs are automatically switched.



Inserting, Replacing, and Deleting Nodes

Once you create a node tree, there are a number of different ways to adjust your composite by reorganizing the nodes in your project.

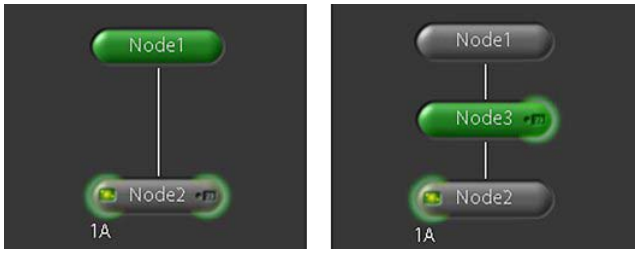
Inserting Nodes Into a Tree

You can insert nodes into the middle of the node tree in the Node View using either the Tool tabs, or the Nodes submenu in the shortcut menu of the Node View. There are several shortcuts you can use to create and insert nodes.

To insert a new node between two nodes, do one of the following:

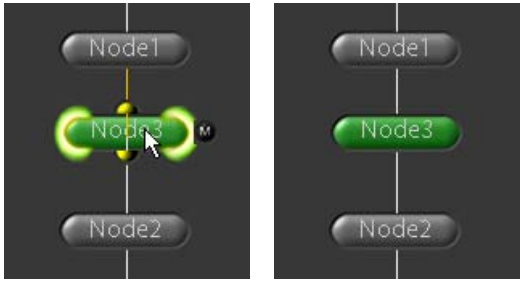
- Select a parent node in the Node View, and click a new node in the Tool tabs.
- Select a parent node in the Node View, then right-click it and choose a node to create from the Nodes submenu of the shortcut menu.

Note: When you insert a node between a node that branches to two other nodes, both branching nodes are connected to the output knot of the newly inserted node.



To insert a disconnected node between two existing nodes:

1 Drag a node directly over a noodle so that both its knots appear highlighted. You can select which input of a multi-input node to insert by dragging the node to the left or right, so that the desired input knot is highlighted.

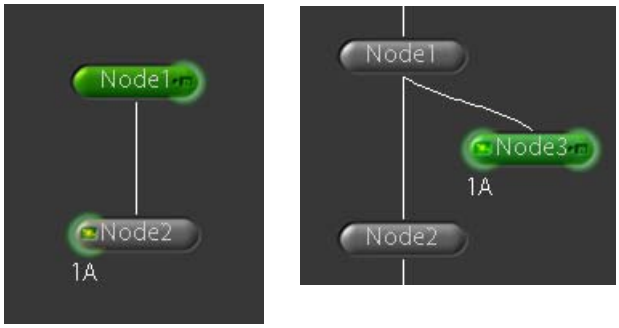


2 When you release the mouse button, the node is automatically inserted.

To create a new branch, do one of the following:

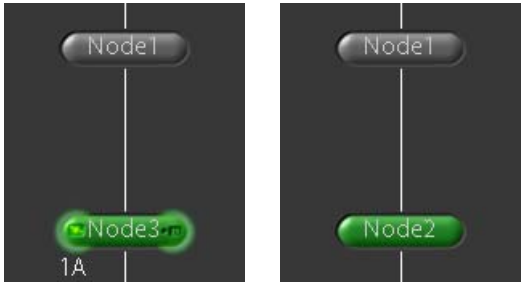
- Select a parent node in the Node View, then Shift-click a new node in the Tool tabs.
- Select a parent node in the Node View, then right-click a node in the Tool tabs, and choose Branch from the shortcut menu.

- Select a parent node in the Node View, then, pressing the Shift key while you right-click it, choose a node from the Nodes submenu at the top of the shortcut menu.



To replace an already existing node with a different one, do one of the following:

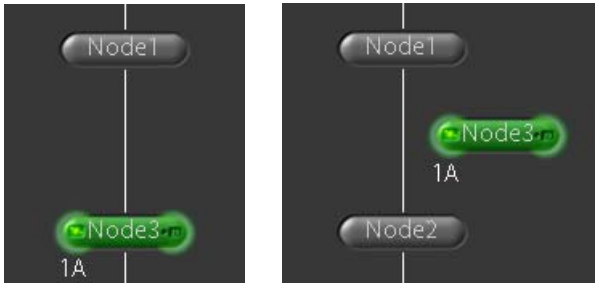
- Select the node you want to replace in the Node View, then Control-click the new node in the Tool tabs.
- Right-click a node in the Tool tabs, then choose Replace from the shortcut menu.
- Select the parent node in the Node View, then, pressing the Control key while you right-click it, choose a node from the Nodes submenu at the top of the shortcut menu.



To create a floating node that's not connected to anything, do one of the following:

- Control-Shift-click the node you want to create in the Node List.
- Right-click a node in the Tool tabs, then choose Create from the shortcut menu.

- Deselect all nodes in the Node View, then right-click in the background area of the Node View and choose a node from the Nodes submenu at the top of the shortcut menu.

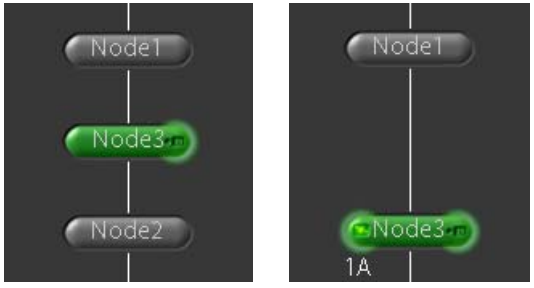


Deleting and Disconnecting Nodes From a Tree

There are several ways to remove nodes from a tree, either by isolating the node, or eliminating it completely.

To delete a node, do one of the following:

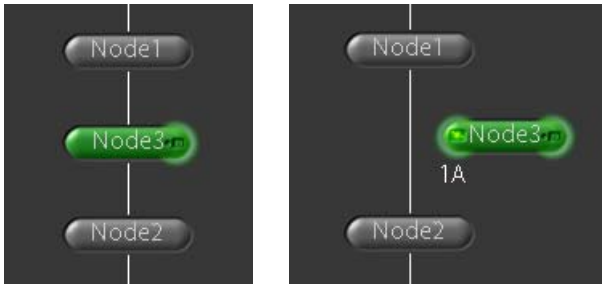
- Select one or more nodes and press Delete or Backspace.
- Select one or more nodes, then right-click one of them and choose Edit > Delete from the shortcut menu.



To extract a node from a tree without deleting it, do one of the following:

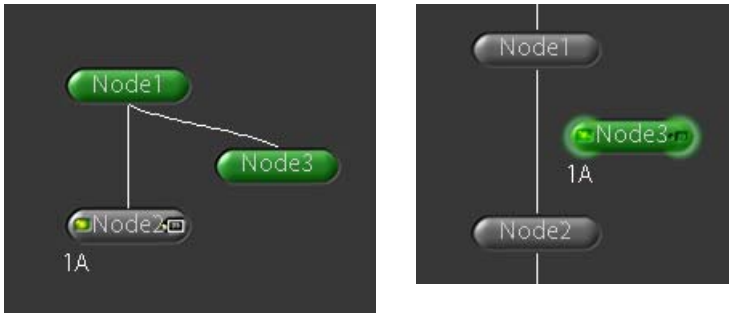
- Select the node and press E (for Extract).
- Select a node, then right-click it and choose Extract Selected Nodes from the shortcut menu.

- Click the node, and with the mouse button held down, drag it *quickly* to the left and right several times to "shake" it loose.



To disconnect a noodle without affecting the nodes above and below, do one of the following:

- Control-click a noodle.
- Move the pointer over the noodle, and when the noodle turns red, press Delete or Backspace.
- Select a node, then right-click it and choose Edit > Delete from the shortcut menu.



Copying and Pasting Nodes

Nodes can be cut and pasted within the Node View.

To copy a node or group of nodes, select one or more nodes and do one of the following:

- Right-click the selected node, then choose Edit > Copy from the shortcut menu.
- Press Command-C or Control-C.

After copying a node, you can now paste or "clone" it.

To paste or "clone" a node, do one of the following:

- Right-click the background of the Node View, then choose Edit > Paste from the shortcut menu.
- Press Command-V or Control-V.

Moving Nodes

To move a node, select the node and drag it within the Node View. If you drag a node past the edge of the Node View, the workspace will scroll, allowing you to move the selection further into that direction.

Loading a Node Into a Viewer

You can view the image output from any single node in your node tree. For example, if you have several color nodes attached in a row, you can load any of these operations into the Viewer to see the result of all of the nodes that have been attached up to that point.

To load a node into the Viewer, do one of the following:

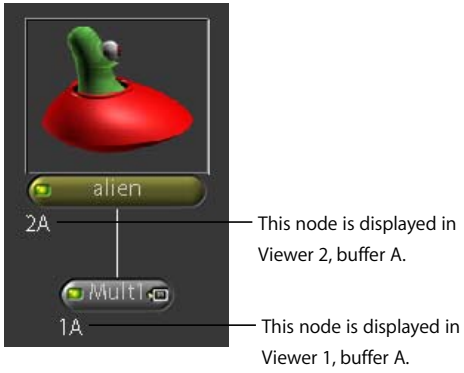
- Click the left side of a node to load that node into the current Viewer.

  A green Viewer indicator appears on the left side of the node.

- Double-click anywhere on a node to simultaneously load it into the Viewer, and its parameters into the Parameters tab.

  The Viewer indicator appears on the left side, and the parameters indicator appears on the right side of the node.

  When a node is loaded into the Viewer, a number and a letter appear underneath the Viewer indicator, identifying the compare buffer that the node's image occupies.



For more information on working with multiple viewers, see "Using and Customizing Viewers" on page 45.

Loading Node Parameters

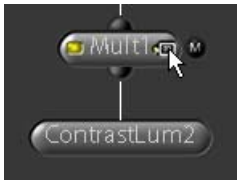
In order to modify a node's parameters, you must first load them into one of the two Parameters tabs. The Parameters tabs remain empty until you click the right side of a node (or double-click the node).

To load a node's parameters into the Parameters1 tab, do one of the following:

- Click the right side of the node.

  The parameters indicator appears on the right side of the node.

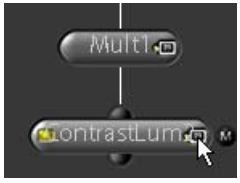
  Note: The node does not have to be selected in order to load its parameters into either of the Parameters tabs.



- Double-click anywhere on the node to load its parameters into the Parameters1 tab and its image into the Viewer.

To load a node's parameters into the Parameters2 tab:

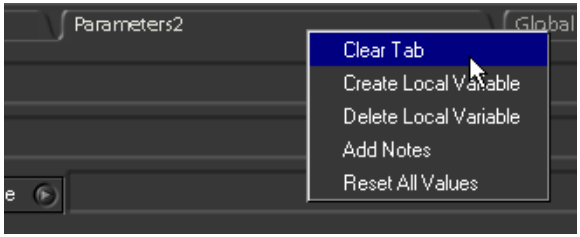
- Shift-click the right side of the node.



Loading a node's parameters into a tab automatically clears out whatever previous parameters were loaded. If necessary, you can clear a Parameters tab at any time.

To clear a tab so that no parameters are loaded into it:

- Right-click the Parameters1 or Parameters2 tab, then choose Clear Tab from the shortcut menu.

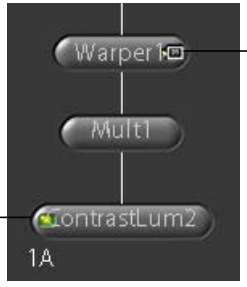


It's important to bear in mind that you can load the image from one node into the Viewer, while loading another node's parameters into the Parameters tab.

Click to load node parameters.

Click once to display node in Viewer. Double-click to load both the Viewer and the node parameters.

For example, you can view the resulting image from the bottommost node in a tree, while adjusting the parameters of a node that's farther up in that tree.



Parameters indicator shows that this node is loaded into one of the Parameters tabs.

Viewer indicator shows that this node is loaded into Viewer 1, buffer A.

The Viewer indicator shows which nodes are loaded into Viewers, and the parameters indicator shows which nodes have been loaded into one of the Parameters tabs.

For more information on adjusting parameters, see "The Parameters Tabs" on page 72.

Ignoring Nodes

Nodes in the node tree can be disabled without actually removing them from the tree, using the Ignore command. Ignored nodes have no effect on your script whatsoever, and are never rendered.

This is a good way to see what effect a node is having on your composition. Ignoring nodes also allows you to disable nodes you may not need any more, without permanently deleting them in the event you change your mind later on.

To toggle nodes off and on, ignoring and restoring them, do one of the following:

- Select one or more nodes, then press I.
- You can also load a node into the Parameters tab, then turn on the ignoreNode parameter.

Ignored nodes appear with a red diagonal slash in the node tree. To reactivate a node, press I again.

Renaming Nodes

For organizational purposes, you may find it useful to rename a node in order to keep track of its function in your composition. This can be especially useful if you have numerous identical nodes in close proximity to one another in a dense node tree.

To rename a node:

1. Load the node's parameters into the Parameters1 tab.
2. Enter a new name into the nodeName field of the Parameters tab.

FileIn and *FileOut* nodes are automatically named based on the media files they're linked to on disk.

Name Nodes Carefully

Here are some rules about names to avoid using:

- Avoid using spaces or non-alphanumeric characters (', .!, and so on).
- Don't name any node "color."
- To avoid confusion, don't give a node another node's name, for example, renaming a *Brightness* node to *Fade*.
- Don't use a name that's used by a local variable within that node.
- Don't name nodes with single characters that typically have other meanings within Shake, such as x, y, z, r, g, b, or a.

Arranging Nodes

Shake has several commands to help you organize and navigate complex node trees. Keeping your node trees clean and organized will save you much time later on, when you're fine-tuning a massively involved 200-node tree.

Grid Snap

You can toggle a grid in the Node View to line up nodes evenly.

To activate a grid in the Node View, do one of the following:

- In the Node View, right-click, then choose Snap to Grid from the shortcut menu.
- Open the guiControls subtree of the Globals tab, and turn on the gridEnabled parameter.

  When the Node View grid is enabled, the nodes you move are automatically aligned.

To temporarily activate grid snapping when Snap to Grid is disabled:

- Press Shift as you drag a node.

To temporarily deactivate grid snapping when Snap to Grid is enabled:

- Press Shift as you drag a node.

Grid Parameters in the Globals Tab

Five parameters, located in the Globals tab in the guiControls subtree, allow you to define the spacing of the Node View grid.

gridWidth, gridHeight

Specifies how wide and tall each rectangle of the grid is.

gridEnabled
Lets you turn grid snapping on and off. This control also toggles the background grid pattern in the Node View if gridVisible is turned on.

gridVisible
Displays the grid as a graph in the background of the Node View. This graph is only displayed when gridEnabled is turned on.

layoutTightness
This parameter affects the Layout Arrangement commands in the next section. It lets you specify how closely nodes should be positioned to one another when they're newly created, or whenever you use one of the arrangement commands.

Automatic Layout Arrangement

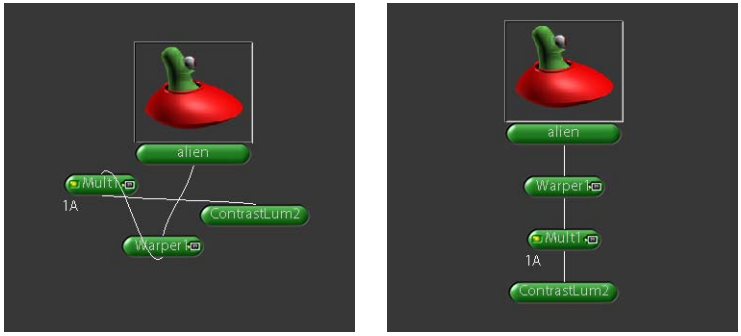
Shake has several commands to help you automatically arrange nodes in the Node View. *Use these commands with caution*—complicated trees with lots of cross-branching may produce odd results.

When nodes are created or organized using the Node Layout commands, they are spaced according to the layoutTightness parameter in the guiControls subtree of the Globals tab. Newly created nodes are spaced this distance from their parents, so a smaller number creates tighter trees.

To automatically organize a group of nodes:

1. Select one or more nodes.
2. Do one of the following:
   - Right-click one of the selected nodes, then choose Node Layout > Layout Selected from the shortcut menu.
   - Press L.

The selected nodes are automatically rearranged in an organized manner.



To align nodes vertically (on the same X axis):

- Select one or more nodes and press X.

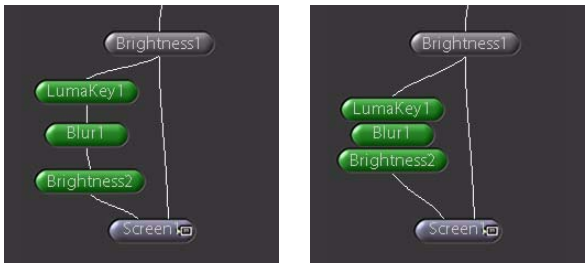
To align nodes horizontally (on the same Y axis):

- Select one or more nodes and press Y.

To compress and align nodes vertically:

- Select one or more nodes and press Shift-L.

The selected nodes are lined up and stacked together one against the other.



Groups and Clusters

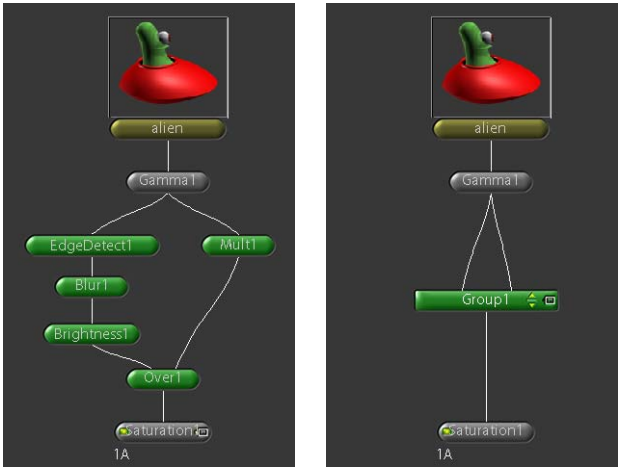
A group is a collection of several nodes that are collapsed to appear as a single object in the Node View. Grouped nodes save space and allow you to organize your node tree into related sets of nodes that perform specific and easily labeled functions.

When you collapse several nodes into a group, the input and output noodles for the topmost and bottommost nodes in the group connect into and out of the group object.

To create a group of nodes:

1. Select one or more nodes.
2. Do one of the following:
   - Right-click in the Node View, then choose Groups > Group Selected Nodes from the shortcut menu.
   - Select the nodes and press G.

- To create a group and immediately open it into a cluster, right-click in the Node View, then choose Group Selected Nodes and Maximize from the shortcut menu (or press Option-G or Alt-G).



To consolidate two or more groups into a larger group:

1. Select two or more groups in the Node View.
2. Do one of the following:
   - Right-click a group, then choose Groups > Consolidate Selected Groups from the shortcut menu.
   - Press Shift-G.

To ungroup nodes, do one of the following:

- Right-click a group, then choose Groups > Ungroup Selected Nodes/Groups from the shortcut menu.
- Select a group, then press Control-G.
- Open the group into a cluster, then click the Ungroup button in the cluster's title bar.

The nodes go back to being a set of individual nodes in the tree. The original connections remain the same.

Clusters

Groups are typically collapsed to save space, but are easily expanded to reveal their contents for further modification. Expanded groups are referred to as *clusters*.

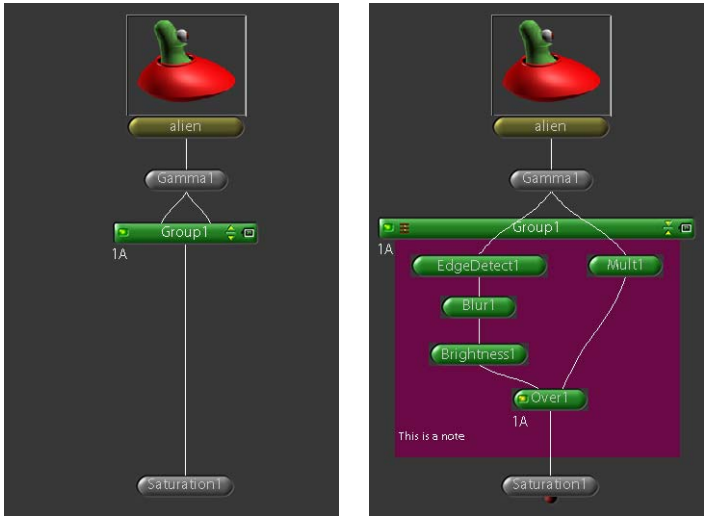
Even if you don't keep a collection of nodes collapsed into a group, having an expanded cluster of nodes in the Node View allows you to color-code them. This color coding can aid visual navigation within a large node tree.

To expand the node group into a cluster:

1. Click the Expand button on the group node.

2 Select a group, then press G.

The group expands into a cluster.



Once a group is expanded into a cluster, the group node includes two additional controls:

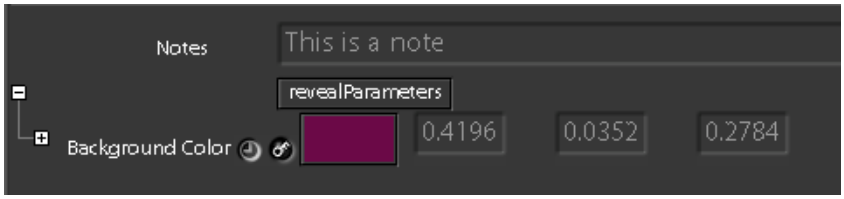
The Ungroup button
The Ungroup button (on the left side of the group node) removes all grouping/cluster information.

When you click this button, a warning window appears that states, "You are about to ungroup the selected group. Continue?" Click Yes to ungroup the selected group.

The Collapse button
The Collapse button (on the right side of the group node) closes the cluster back into a normal group.

Group Parameters

Loading the parameters of a group into the Parameters tab allows you to change the color of the cluster background (see below), add notes, and expose selected parameters.



To add a note to a group:

1. Load the group into the Parameters tab.
2. Type your annotation into the Notes parameter field.

Cluster notes appear at the bottom-left corner of an open cluster.

To change the background color of a cluster:

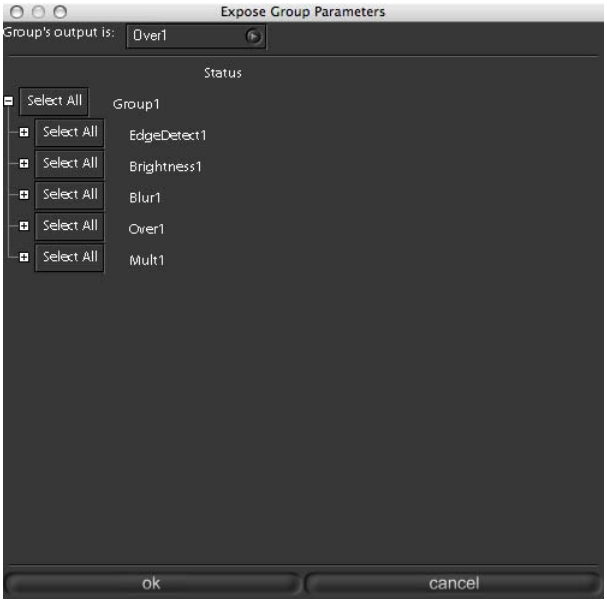
1. Load the group into the Parameters tab.
2. Open the revealParameters subtree, and use the Background Color control to pick a new color.

The revealParameters button lets you isolate important sliders from multiple nodes and load them into a single Parameters tab, saving you the trouble of jumping between multiple nodes.

To expose individual parameters in a group:

1. Load the parameters of the group (click the right side of the group node).
2. In the group Parameters tab, click the revealParameters button.

The Expose Group Parameters window appears.

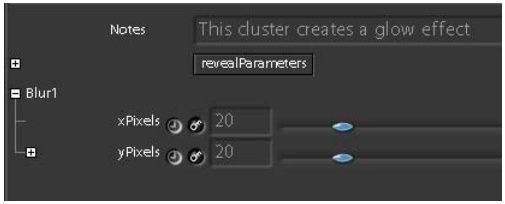


3 To select one or more node parameters from nodes within the cluster, do one of the following:

- To expose every parameter within a node, click Select All.
- To expose individual parameters within a node, expand the node's Select All subtree, then enable the desired parameters.

4 Click OK.

The selected nodes parameters appear in the group Parameters tab.

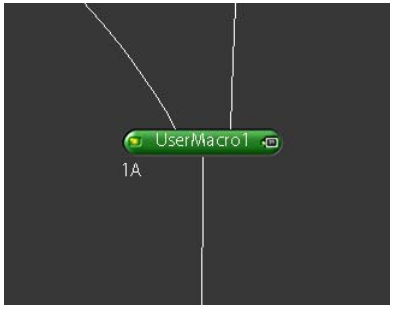


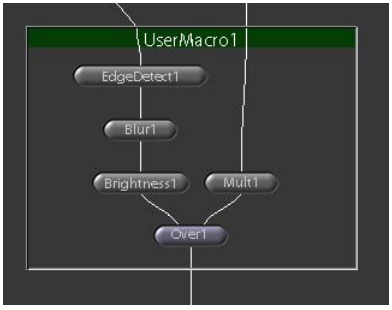
Opening Macros

If you're using macros within your script, they can be opened and closed in much the same way as groups.

To examine the contents of a macro, do one of the following:

- Right-click a macro, then choose Macro > Show Macro Internals from the shortcut menu.
- Press B.





When a macro is open, you can view any parameter or stage of the macro, but you cannot edit parameters or rewire nodes. This functionality is primarily useful for understanding the workings of a macro you may be unfamiliar with.

To close a macro, do one of the following:

- Click the macro.
- Position the pointer on an empty area, and press Option-B or Alt-B.

Cloning Nodes

It's possible to create clones of nodes within your node tree. For example, if you've created a color-correcting node with specific settings, and you want to use it multiple times in your script, you can create *clones*.

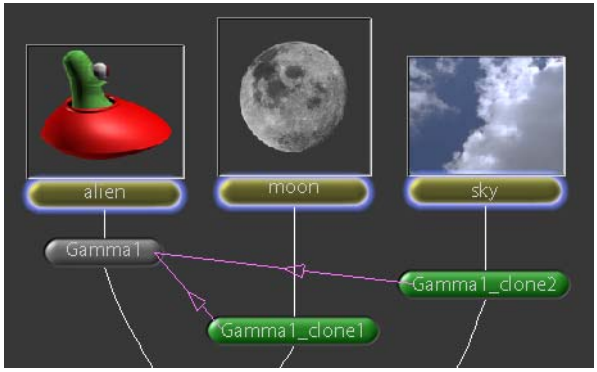
The principal advantage to cloned nodes is that changes made to one cloned node are automatically applied to every other cloned duplicate of that node within your script.

To create a clone of a node:

1. Copy a node by pressing Command-C or Control-C.
2. Paste a clone of that node by doing one of the following:
   - Right-click in the Node View, then choose Edit > Paste Linked from the shortcut menu.
   - Press Command-Shift-V or Control-Shift-V.

Note: You cannot clone *Tracker*, *LookupHSV,* or *FilmGrain* nodes in the node tree using the Paste Linked command.

The cloned node is pasted, and automatically named *OriginalNode_CloneN*, where *N* is the number of clones that have been made. You can clone a node as many times as you want, and all cloned nodes are linked to the original node. In the following screenshot, the *Gamma1* node has been cloned twice, in order to apply an identical gamma correction to the other two images in the tree.



The links between cloned nodes can be viewed by turning on ShowExpressionLInks in the enhancedNodeView subtree of the Parameters tab (and then turning on enhancedNodeView). For more information, see "Using the Enhanced Node View" on page 221.

Thumbnails

By default, thumbnails are automatically generated in the Node View for image nodes, including but not limited to the *FileIn, Grad, Ramp,* and *RotoShape* nodes. These thumbnails are meant to help you navigate the Node View by showing where the originating images in your script are. In order to prevent these thumbnails from slowing down Shake's processing speed, they do not automatically update to reflect changes made to them. You can manually update them, if necessary, a process described later in this section.

Thumbnails are stored in the cache and, since they are rarely modified, are fast to load when you load a script.

Customizing Thumbnail Display

Four parameters in the displayThumbnails subtree of the guiControls subtree of the Globals tab allow you to customize how thumbnails are displayed in the Node View.

displayThumbnails
Turns all thumbnails in the Node View on and off.

thumbSizeRelative
Makes thumbnails a similar size or relative to their actual sizes. By default, all thumbnails are displayed at the same width. To display thumbnails at their relative sizes, turn on thumbSizeRelative.

In the following images, the greenscreen image is PAL and the truck image is 410 x 410.

thumbSizeRelative deactivated

thumbSizeRelative activated

thumbSize
Lets you adjust the size of thumbnails in the Node View. If thumbSizeRelative is turned on, all nodes are resized relative to one another.

thumbAlphaBlend
Turns thumbnail transparency on and off. When thumbAlphaBlend is on, moving one thumbnail over another results in a fast look at how the nodes might appear when composited together in the Viewer. More usefully, it gives you an instant view of which images have transparency in them.

Once you've customized the thumbnail settings for your project, you can save these parameter settings.

To save the new settings:
- Choose File > Save Interface Settings.

Adding Thumbnails to Nodes

Most nodes are not created without thumbnails. However, you can display a thumbnail for any node in the Node View. This can help you to emphasize key points in your node tree that you'd like to use as roadsigns to show how things are working.

To toggle a thumbnail on or off for any node:
1. Select one or more nodes.
2. Do one of the following:
   - Right-click the selected node, then choose Thumbnails > Show/Hide Selected Thumbnails from the shortcut menu.

- Press T.

When a node with a thumbnail appears in the middle of a node tree, the input noodles feed into the top of the thumbnail.

Updating Thumbnails

To prevent unnecessary processing, thumbnails are not automatically updated, so they may not reflect the frame that's at the current position of the playhead. Furthermore, nodes aren't updated when you change their parameters. This can be especially noticeable in the thumbnails of *QuickPaint* and *RotoShape* nodes.

Thumbnails can be updated manually, so that they more accurately represent the current state of the node tree.

To update a thumbnail:

1. Select one or more nodes.
2. To display a different frame, move the playhead in the Time Bar.
3. Do one of the following:
   - Right-click in the Node View, then choose Thumbnails > Refresh Selected Thumbnails from the shortcut menu.
   - Press R.

Toggling Thumbnails Between Color and Alpha Channels

When the pointer is positioned over a thumbnail, a number and letter appear in the upper-left corner, indicating which frame is loaded as a thumbnail, and whether you are looking at the RGB/Color view (C) or the Alpha view (A). Thumbnails can be toggled between Alpha and Color view on an individual basis.

Indicates a thumbnail displaying frame 1, in Alpha view.

To toggle thumbnails between Color and Alpha view:

1. Select one or more nodes.
2. To display the Color view, right-click in the Node View, then choose Thumbnails > View RGB channels from the shortcut menu.
3. To switch back to the Alpha view, right-click in the Node View, then choose Thumbnails > View Alpha channels from the shortcut menu.

You can also place the pointer over a thumbnail and press A or C to toggle between the Alpha view and the Color view.

Defining Which Nodes Are Created With Thumbnails

You can declare any specific type of node to always contain thumbnails upon creation. For example, to add *FileOuts* into the list of nodes receiving a thumbnail, set the following *ui.h* code:

nuiNodeViewEnableThumbnail("FileOut");

You can also disable an enabled node with the following *ui.h* code:

nuiNodeViewDisableThumbnail("FileOut");

For all nodes, use *NRiFx* as your class. Note, however, that specifying downstream nodes (especially *FileOut* nodes) can cause pauses at script load time, as the entire tree must be calculated to derive the proper thumbnail. Use with caution.

To have thumbnails always off, disable the thumbnails in the guiControls of the Globals tab, then choose File > Save Interface Settings, or add the following *ui.h* code:

script.displayThumbnails = 0;

The Node View Shortcut Menu

The following commands are available in the shortcut menu that appears when you right-click in the Node View.

Shortcut Menu	Option	Keyboard	Desription
Nodes			Create nodes directly in the Node View from the node list.
Edit	Cut	Command-X or Control-X	Removes selected nodes and places them into the paste buffer.
	Copy	Command-C or Control-C	Copies the selected nodes into the paste buffer.
	Paste	Command-V or Control-V	Pastes the buffer into the Node View. You can also copy nodes from the Node View and paste them into a text document.
	Delete	Del or Backspace	Deletes the selected nodes. If the branching is not complicated, the noodles between the parent(s) and children are automatically reattached to each other.
	Undo	Command-Z or Control-Z	Undo up to 100 steps. Rearranging nodes counts as a step.
	Redo	Command-Y or Control-Y	Redo your steps unless you have changed values after several undos.
View	Zoom In	+	Zooms into the Node View (also use Control-Option-click or Control-Alt-click).
	Zoom Out	–	Zooms out of the Node View (also use Control-Option-click or Control-Alt-click).
	Reset View	Home	Centers all nodes.
	Frame Selection	F	Frames all selected nodes into the Node View.
Render	Render Flipbook		Renders a Flipbook of the node visualized in the active Viewer.
	Render Disk Flipbook		Mac OS X only. This option launches a disk-based Flipbook into QuickTime. This has several advantages over normal Flipbooks. It allows for extremely long clips and allows you to attach audio (loaded in with the Audio Panel in the main interface). You can also choose to write out the sequence as a QuickTime file after viewing, bypassing the need to re-render the sequence.
	Render FileOuts		Opens the render window, which lets you set how you want to render FileOuts in your script.
	Render Proxies		Opens the render proxy parameters window.
Overview On/Off		O	Turns on the Overview window to help navigate in the Node View.

Shortcut Menu	Option	Keyboard	Desription
Enhanced Node View On/Off		Control-E	Turns the selected enhanced Node View options off and on.
Snap to Grid On/Off			Turns gridEnabled on and off in the Globals tab.
Select	Find Nodes	Command-F or Control-F	Activates nodes according to what you enter in the Search string field in the Select Nodes by Name window. • *Select by name*. Enter the search string, and matching nodes are immediately activated. For example, if you enter just *f*, *FileIn1* and *Fade* are selected. If you enter *fi*, just *FileIn1* is selected. • *Select by type*. Selects by node type. For example, enter *Transform*, and all *Move2D* and *Move3D* nodes are selected. • *Select by expression*. Allows you to enter an expression. For example, to find all nodes with an angle parameter greater than 180: angle >180 • *Match case*. Sets case sensitivity.
	All	Command-A or Control-A	Selects all nodes.
	Associated Nodes	Shift-A	Selects all nodes attached to the current group.
	Invert Selection	!	All selected nodes are deactivated; all deactivated nodes are activated.
	Select Upstream	Shift-U	Adds all nodes upstream from the currently active nodes to the active group.
	Select Downstream	Shift-D	Adds all nodes downstream from the currently active nodes to the active group.
	Select Upstream 1 Level	Shift-Up Arrow	Adds one upstream node to the current selection.
	Add Downstream 1 Level	Shift-Down Arrow	Adds one downstream node to the current selection.
Node Layout	Layout Selected	L	Automated layout on the selected nodes.
	Align Selected Vertically	X	Snaps all selected nodes into the same column.
	Align Selected Horizontally	Y	Snaps all selected nodes into the same row.

Shortcut Menu	Option	Keyboard	Desription
Thumbnails	Refresh Selected Thumbnails	R	Activates/refreshes the thumbnails for selected nodes.
	Show/Hide Selected Thumbnails	T	Turns on/off selected nodes. If you haven't yet created a thumbnail (R), this does nothing.
	View RGB Channels	C	Displays the RGB channels.
	View Alpha Channels	A	Displays the alpha channel.
Groups	Group/ Ungroup Selected Nodes	G	Visually collapses selected nodes into one node. When saved out again, they are remembered as several nodes. To ungroup, press G again.
	Group Selected Nodes and Maximize	Option-G or Alt-G	Groups nodes and automatically maximizes the group, for immediate editing.
	Maximize/ Minimize Selected Groups	M	Opens a group into a subwindow.
	Consolidate Selected Groups	Shift-G	Consolidates two or more selected groups into a larger group.
	Ungroup selected Nodes/Groups	Command-G or Control-G	Turns a group of nodes back into a collection of individual, ungrouped nodes.
Ignore/ UnIgnore Selected Nodes		I	Turns off selected nodes when activated. Select them again and press I to reactivate the nodes. You can also load the parameters into the Parameter View and enable ignoreNode.
Extract Selected Nodes		E	Pulls the active nodes from the tree, and reconnects the remaining nodes to each other.
Save Selection as Script		S	Saves the selected nodes as a script.
Force Selected FileIn Reload			When an image is changed on disk and you have already looked at that image in the interface, Shake does not recognize that the image has changed. Selecting these two functions forces the checking of the date stamp for the images on disk.
Force All FileIn Reload			See above.
Macro	Make Macro	Shift-M	Launches the MacroMaker with the selected nodes as the macro body.

Shortcut Menu	Option	Keyboard	Desription
	Show Macro Internals	B	Opens a macro into a subwindow so you can review wiring and parameters. You cannot change the nodes inside the subwindow.
	Hide Macro Internals	Option-B or Alt-B	Closes up the macro subwindow when the pointer is placed outside of the open macro.

Using the Time View

8

The Time View provides a centralized representation of the timing for each image used in a script. This chapter covers how to navigate this interface, and how to make adjustments to the timing parameters of each image.

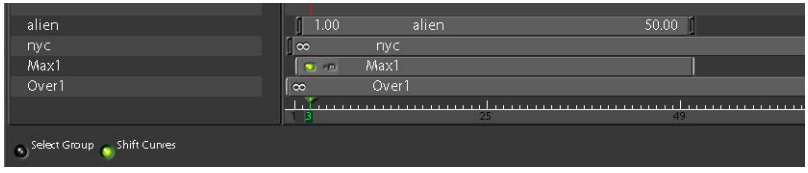
About the Time View

While the Node View allows you to arrange and adjust the nodes that comprise your composite, the Time View lets you view and arrange the timing of your nodes. Specifically, the Time View displays all image nodes, as well as nodes with more than one input, that appear in your node tree.

In the Time View, nodes are displayed as horizontal bars. You can select them and load them into the Viewer or Parameters tab, just as you can in the Node View. In addition, you can also set In and Out points for your clips, as well as shift a clip's start and end frames to change its duration. Finally, the Time View allows you to change the looping behavior of clips in order to extend their duration.

To display the Time View, click the Time View tab (on the right side of the Tool tabs).

The Time View appears, displaying each node stacked over a second Time Bar that mirrors the Time Bar found at the bottom of the Shake interface.

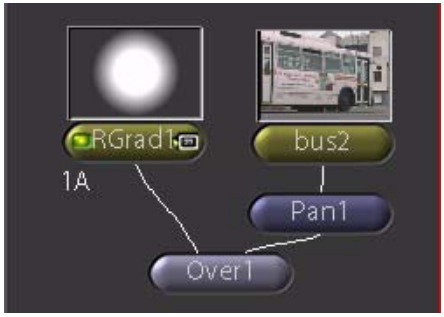


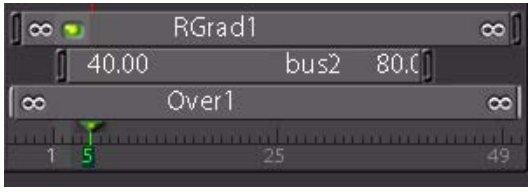
The Time View lets you modify the timing parameters that are found inside each *FileIn* node in your node tree. This means that you have the option of modifying these timing parameters either numerically, in the Parameters tab, or graphically, in the Time View. Either way, the effect is the same.

As soon you make changes to a clip's timing, an internal node called *IReTime* is associated with a *FileIn* node. The *IReTime* node is saved into the script, but is invisible in the Node View. The *IReTime* parameters are controlled by the *FileIn* node parameters and in the Time View.

Viewing Nodes in the Time View

The Time View displays only image nodes and compositing nodes with more than one input. Other nodes are not represented. For example, in the following node tree, the *Pan* node is not visible in the Time View.





It's possible to further reduce the number of nodes displayed in the Time View, allowing you to concentrate on only a select group of nodes.

To display only currently selected nodes:

- Turn on the Select Group, located at the bottom-left side of the Time View.

Clip Durations in the Time View

The duration of image sequences and movie files (hereafter referred to as *clips*) referenced by a *FileIn* node is simply that of the source media on disk. The duration of single image files, and of image nodes generated by Shake, is considered to be infinite.

When two or more nodes are combined, as with the *Over* or *MultiLayer* node, the final duration is that of the longest clip in the operation.

When a *FileIn* node is created, its timing parameters are referenced via that node's Timing tab in the Parameters tab. Other image nodes have a timing subtree within the main Parameters tab. Additional nodes that are connected to a clip inherit the source clip's timing. You cannot adjust the timing of non-image nodes.

Adjusting Image Nodes in the Time View

In the Time View, nodes representing images (whether clips, single image files, or Shake-generated images such as gradients and rotoshapes) have several controls attached to them. Handles at the beginning and end of image nodes allow you to adjust their timing, while other controls let you load parameters, ignore and "unignore" nodes, and perform other functions.

Trimming and Looping QuickTime Clips and Still Images

The methods described in this chapter for trimming and looping clips in the Time View do not work the same with QuickTime clips, Still Images, or Shake-generated images. The following exceptions apply:

- QuickTime clips cannot be trimmed using the timing handles, because QuickTime clips do not have corresponding startFrame and endFrame parameters in the Source tab of the *FileIn* parameters. However, QuickTime clips can be looped for their full duration.
- Still Images and Shake-generated images cannot be looped, as a single image's duration is infinite by default.

Image Node Controls

When you move the pointer over an image node in the Time View, three controls appear: the Viewer indicator, the parameters indicator, and the Ignore control.

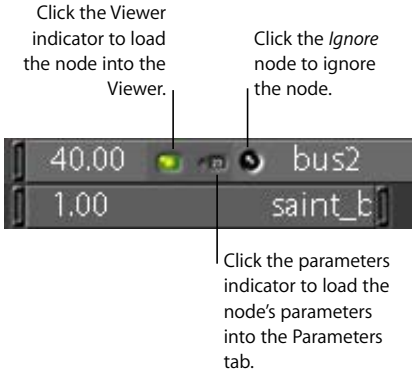
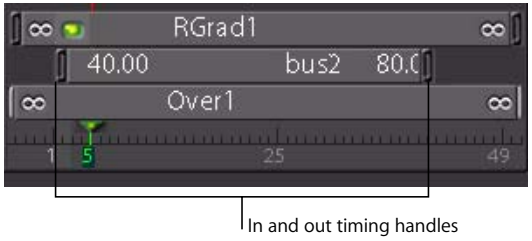


Image Sequence Timing Controls

In the Time View, image nodes also have *timing handles,* located at the beginning and end of the bars. Timing handles allow you to adjust the In and Out points of a clip or Shake-generated image. Non-image nodes have no timing handles.

You can adjust the In and Out points of an image node by dragging its timing handles to the left or right. You can also drag image nodes and compositing nodes in the Time View, changing their location in time.

In the following screenshot, the *RGrad1* and *bus2* image nodes both have timing handles, at the left and right edges of the bars. However, the *Over1* node has no handles because, as a non-image compositing node, it inherits the duration of the longest image node to which it is connected.



Adjusting a Clip's timeShift Parameter

The timeShift parameter corresponds the the clip's position in the Time View—adjusting the timeShift parameter moves the entire clip forward or backward along the Time View.

To shift a node in time:

- Drag an image node in the Time View to the left or right.

That node's timeShift parameter changes, and the start and end frames of the node are moved together. The inPoint and outPoint parameters, however, remain the same, since this operation does nothing to change the clip's duration.

All compositing nodes that are attached to a time-shifted image node are automatically shifted in time to match. When you drag a non-image compositing node in the Time View, such as a layering node, the timing of all image nodes connected to it is also modified.

In the above example, when the *Over1* node is dragged, all nodes above the *Over1* node (including the invisible *Pan1* node) are shifted, as well. This means that the frames of *bus2* are shifted in time, and any animation curves on *Pan1* and *RGrad1* are shifted as well. If you drag only the *bus2* clip, then only the *bus2* clip is modified in time—unless the Shift Curves control (in the bottom-left corner of the Time View) is activated.

When the Shift Curves control is enabled, all curves that are attached to the shifted nodes are shifted as well, so the animation is carried with the shift. This control is enabled by default, and in most cases should be left on.

Important: If Shift Curves is disabled, the curves remain locked in their previous positions, and will be offset from the new position of the clip.

Adjusting an Image Sequence's firstFrame and lastFrame Parameters

You can trim frames off of the beginning or the end of a clip by adjusting its firstFrame and lastFrame parameters—either in the Source tab, or in the Time View.

Note: QuickTime movies do not have firstFrame and lastFrame parameters.

To adjust the startFrame and lastFrame points of an image sequence:

- In the Time View, drag the left handle of an image node to adjust its inPoint parameter, or drag the right handle to adjust its outPoint parameter.

Notice that the firstFrame value for that clip is labeled on the left side of the bar, and the lastFrame value is labeled on the right side—in the example above, the inPoint parameter is 40 and the outPoint parameter is 138. These numbers are dynamic, and change while you modify a clip's In or Out point.

If you drag the timing handles of an image node in the Time View, you change the node's duration. For example, if you drag the ends of *RGrad1*, you set the boundaries within which that node appears in time. When you drag the left or right handles of a Shake-generated node (such as a *Color, Grad,* or *Ramp* node) in the Time View, the duration of that image is extended to fill the new time range.

Using the inPoint and outPoint to Repeat Clips

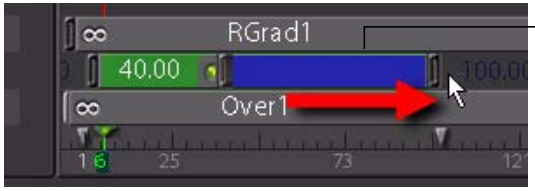
You can extend the beginning or end of an image sequence or QuickTime movie beyond its actual duration without retiming it by setting the clip to repeat frames.

To extend the inPoint or outPoint frame of a clip so that it repeats:

- In the Time View, Control-drag the In or Out handle of a clip.

The clip's duration extends as you pull a new handle to the right or left. The original firstFrame and lastFrame timing handles and parameters remain as they were, but the inPoint and outPoint parameters update to reflect the extended duration.

By default, the newly extended area of the clip is represented by a blue bar, which tells you that the expanded time range in the clip has been filled by a freeze frame of the In or Out point. In the example below, Control-dragging the Out point of the *bus2* clip to frame 100 creates an extended freeze frame effect from frame 41 to frame 100.



You can change the repeat mode of an extended-duration clip at any time using the controls in the Timing tab of that image node. A clip's repeating behavior is controlled by the inMode and outMode parameters in the Timing tab of the *FileIn* parameters.

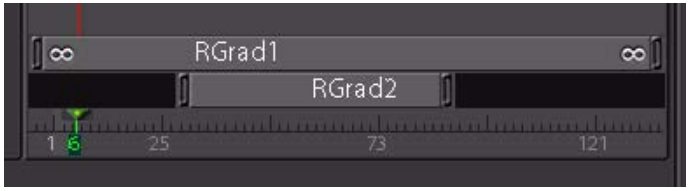
The inMode parameter controls the looping behavior of frames between the firstFrame looping handle and the inPoint, and the outMode parameter controls the frames between the outPoint and the lastFrame looping handle.

The following table lists the available repeat modes.

Mode	Result	Example
Black	No frames are repeated. Instead, black frames are inserted.	
Freeze	The first/last frames are repeated as a freeze-frame.	
Repeat	The clip is continuously repeated, always starting from the first frame.	
Mirror	The clip repeats, first in reverse order, and then forward, indefinitely. To provide a smooth transition, the first frame does not repeat between the loops.	
Inclusive Mirror	The entire clip repeats, first in reverse order, and then forward, indefinitely.	

Clips With Infinite Duration

Image nodes such as *RGrad* and *Ramp* have no preset range because they are generated by Shake. In the Time View, these types of nodes have infinity symbols on their left and right edges, indicating that these images have no end. To limit these nodes, grab the handles as you would with clip nodes.



Customizing How the Last Frame Is Represented

The Out point of an image node represents different things to different users, depending on which media applications they're used to. For video editing applications (usually), an Out point is the frame at which there is no more image (and is therefore black). For a clip of 50 frames, video editing applications usually consider the Out point to be frame 51. For CG artists, the Out point is the last frame to render, making it frame 50 in this example. To add to the confusion, keep in mind that even if you have 50 frames, you have meaningful information up to frame 50.99999—incremental frames are necessary for motion blur calculations and field rendering.

You can right-click in the empty part of the Time View to reveal a shortcut menu that lets you change how the Out point is represented. The shortcut menu contains the following options:

In/Out Point Display

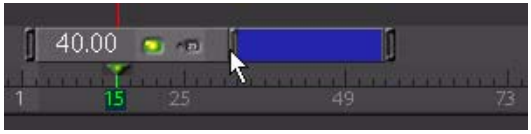
This option toggles the display of the In/Out point in the Time View. When enabled, the In and Out points are displayed whenever you click the end of a clip.

Const Point Display

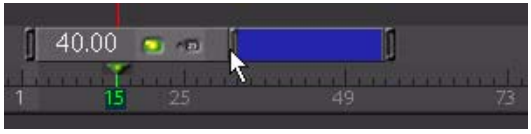
When Const Point Display is enabled, the frame considered as the Out point is toggled to the frame at which it becomes black, or the last frame on disk.

FileIn Trim

The FileIn Trim option controls what happens when you drag the outPoint handle and right looping handle past each other (first image). With FileIn Trim off, Control-dragging a timing handle past a looping handle collapses the looping portion of the clip as the clip's total duration is changed.



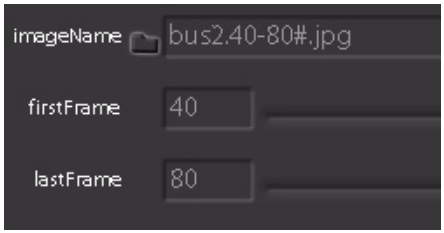
When FileIn Trim is turned on, you can Control-drag a timing handle past a looping handle. This is useful if you want to keep the original frame range of the clip as a reference.



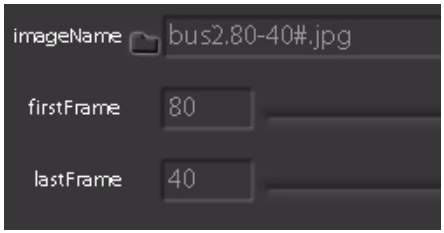
Reversing a Clip

To reverse a clips so that it plays backwards, simply switch the firstFrame and lastFrame values in the Timing tab of the clip's *FileIn* node. This cannot be done with QuickTime clips, because QuickTime clips do not have firstFrame and lastFrame parameters.

In the following example, a clip that begins at frame 40 and ends at frame 80 is reversed by manually swapping inPoint and outPoint values.



The modified clip now begins at frame 80, then plays in reverse until it reaches frame 40.



There are no interface controls in the Time View for this functionality.

The Transition Node

The *Transition* node, located in the Other tab, is an editing node to mix or cut two clips together. It is unique in that it is really a shell to drive other functions that determine the mixing. Modifying it also modifies the timing of the second input.

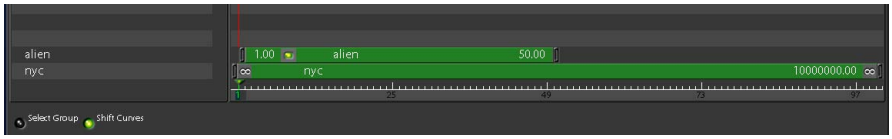
The mixers parameter of the *Transition* node can be set to cut, which simply cuts to the second clip at the frame that it starts. The *Transition* node also can be set to any of the following transitions:

- Dissolve
- HorizontalWipe
- VerticalWipe

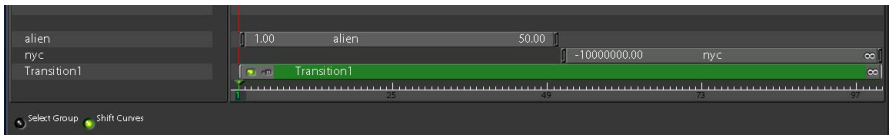
The duration of the transition is determined by the overlap value, and starts at the frame where the second clip appears. If you modify the timing of the second clip, the overlap value automatically changes as well.

A third parameter, mixPercent, appears whenever the mixer parameter is set to anything other than "cut." mixPercent determines the timing for the mixing. For example, for dissolve, if mixPercent is at 20, the second image is 20 percent mixed in and the first image is 80 percent. You can tune a curve interactively in the interface to adjust timing.

In the following example, two clips have been added to the script.



Connecting both *FileIn* nodes to a *Transition* node (located in the Other tab) offsets the second input clip from the first. Notice that the transition node appears underneath, spanning the combined duration of both clips.



You can now route the combined output of the *Transition* node to as complicated a node tree as you like, and both image nodes feeding the transition node will be treated as a single image stream.

So, how is this different from just compositing the clips with an *Over* node and offsetting the second clip in time? The advantage is that you can easily dial in an overlap value to determine how many frames they overlap, and add a simple transition effect. In the following screenshot, the overlap value is increased in the *Transition* node, and the second node shifts to the left as it increases. You can also shift the second clip, and read the overlap value in the *Transition* node.

Note: When the mixer parameter is set to cut, the cut point occurs at the beginning of the second clip, not at the end of the first clip.

Parameters in the Transition Node

The Transition node has the following parameters:

overlap

This parameter sets the amount that the second clip is shifted earlier (to the left) to provide overlap of the two clips.

mixer
Other default choices are:

- cut
- dissolve
- horizontalWipe
- verticalWipe

You can also add your own custom effects. For more information, refer to "Customizing the Transition Node" below.

clipMode
This option appears when mixer is set to Dissolve, allowing you to choose which image sets the DOD.

channels
This parameter appears when mixer is set to Dissolve, allowing you to choose which channels are dissolved.

blur
This parameter appears for horizontal and verticalWipe, and is used to soften the wiping edge.

reverse
This parameter appears for horizontal and verticalWipe, and is used to flip the direction, for example, from left to right to right to left.

mixPercent
This parameter, with accompanying graph, appears whenever the mixer parameter is set to anything other than "cut." mixPercent determines the timing for the mixing. For example, for dissolve, if mixPercent is at 20, the second image is 20 percent mixed in and the first image is 80 percent. You can tune a curve interactively in the interface to adjust timing.

Customizing the Transition Node

If you like, you can create custom mixers for use by the *Transition* node.

To create your own custom mixers in a *startup* .h file, you must do two things:

- Create a macro with two image inputs, *i1* and *i2*, and a float parameter named mixPercent that typically has the default value of "HermiteV(x,1,[0,50,50]@0,[100,50,50]@100)". This gives you the animation curve that can then be tuned in the interface. You can also add other parameters.
- Declare the function as a mixer for *Transition* with the *nfxDefMixer* command in a *startup* .h file. The first parameter is the name of the mixer as it appears in the list. The second entry is the call to the macro:

```
nfxDefMixer("horizontalWipe","HWipe()");
```

The following is an example from the *include/nreal.h* file for *horizontalWipe*:

```
image HWipe(
  image i1=0,
  image i2=0,
  float blur=0,
  int reverse=0,
float mixPercent="HermiteV(x,1,[0,50,50]@0,[100,50,50]@100)"
)
{
  Color1 = Color(
     max(i1.width,i2.width),
     max(i1.height,i2.height),
     1, 1, red, red, 1, 0);
  Crop1 = Crop(Color1, 0, 0, width, height);
  Pan1 = Pan(Crop1, mixPercent*width/100*(reverse?-1:1));
  BlurMe = Blur(Pan1,blur,0,0);
  IMult1 = IMult(i1, BlurMe , 1, 100, 0);
  Invert1 = Invert(BlurMe , "rgba");
  IMult2 = IMult(Invert1, i2, 1, 100, 0);
  IAdd1 = IAdd(IMult1, IMult2, 1, 100);
  return IAdd1;
}
nfxDefMixer("horizontalWipe","HWipe()");
```

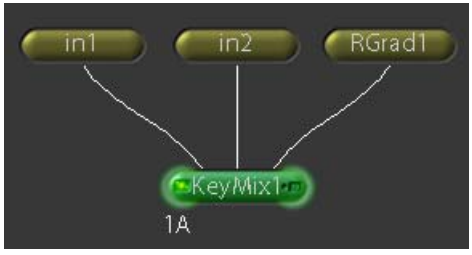
Notice that in mixPercent the default curve goes from 0,0 to 100,100 for mixPercent. Also, notice how the Color generator compares the two input resolutions to determine how large to make the max function.

Create Your Own Transition Type

In this example, you will make your own transition mixer by scaling the radius of an *RGrad* node to create a radial wipe.

To create a custom radial transition:

1 Begin with a simple tree that feeds two *FileIn* nodes into a *KeyMix* node. The two clips are named *i1* and *i2* (to help later).



The following images show the effect that can be achieved by increasing and decreasing the *RGrad* radius.

Select all of the nodes in the tree in the Node View, and copy them (press Command-C or Control-C).

2 Create a text file, and paste (press Command-V orControl-V) the nodes you've copied into it. You'll notice that the node tree you copied is automatically converted into Shake script. It should look like this:

```
RGrad1 = RGrad(720, 486, 1, width/2, height/2, 1
  min(width,height)/4,
  min(width,height)/4, 0.5, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0);
in1 = FileIn("myclip.1-39#.iff",
  "Auto", 0, 0);
in2 = FileIn("myotherclip.1-39#.iff",
  "Auto", 0, 0);
KeyMix1 = KeyMix(in1, in2, RGrad1, 1, "A", 100, 0);
// User Interface settings
SetKey(
  "nodeView.KeyMix1.x", "156.75",
  "nodeView.KeyMix1.y", "127",
  "nodeView.RGrad1.x", "317.4916",
  "nodeView.RGrad1.y", "198.6512",
  "nodeView.in1.x", "67",
  "nodeView.in1.y", "201.125",
  "nodeView.in2.x", "202",
  "nodeView.in2.y", "198.3111"
);
```

3 Save the text file into your *$HOME/nreal/include/startup* directory. Don't close the file yet, some changes need to be made first.

4 You can now prune a lot of the data, keep the bold sections of the above code, and format it as a macro. You also want to add the standard parameters of blur, mixPercent, and reverse. Copy these parameters from the *nreal.h* file's *HWipe* node (at the end of the file).

Finally, calculate the resolution of the *RGrad* by comparing the two input sizes. The script should now look like this:

```
image RadialWipe(
  image in1=0,
  image in2=0,
  float blur=0,
  int reverse=0,
  float mixPercent="HermiteV(x,1,[0,50,50]@0,[100,50,50]@100)"
)
{
  RGrad1 = RGrad(
     max(in1.width,in2.width),
     max(in1.height,in2.height),
     1, width/2, height/2, 1,
     min(width,height)/4,       //This is the radius
     min(width,height)/4,       //This is the falloff
     0.5, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0);
  return KeyMix(in1, in2, RGrad1, 1, "A", 100, 0);
}
```

5 The maximum distance to expand the *RGrad* can be calculated by measuring the distance from the center to a corner, which can be done with the distance() function. (For more information, see Chapter 31, "Expressions and Scripting," on page 935.) Once this is calculated, multiply it by the mixPercent. Also, plug the blur value into the falloff parameter, with a check on the radius to see if falloff should equal 0 when radius equals 0. Also, add the command to load it as a mixer in the *Transition* node:

```
image RadialWipe(
  image in1=0,
  image in2=0,
  float blur=0,
  int reverse=0,
  float mixPercent="HermiteV(x,1,[0,50,50]@0,[100,50,50]@100)"
)
{
  RGrad1 = RGrad(
  max(in1.width,in2.width),
  max(in1.height,in2.height),
  1, width/2, height/2, 1,
  mixPercent*distance(0,0,width/2,height/2)/100,
  radius==0?0:blur,
  0.5, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0);
  return KeyMix(in1, in2, RGrad1, 1, "A", 100, 0);
}
nfxDefMixer("radialWipe", "RadialWipe()");
```

6 Now comes the tricky bit—reversing the mix. You may think multiplying by -1 inverts the transformation, but you'd be wrong. Instead, you often have to subtract the value from the maximum value that you expect, in this case the distance from the center to the corner. This is part of a conditional statement that tests to see if reverse is activated. Also, invert the mask in the *KeyMix* to help it out.

```
image RadialWipe(
  image in1=0,
  image in2=0,
  float blur=0,
  int reverse=0,
  float mixPercent="HermiteV(x,1,[0,50,50]@0,[100,50,50]@100)"
)
{
  RGrad1 = RGrad(
     max(in1.width,in2.width),
     max(in1.height,in2.height),
     1, width/2, height/2, 1,
     reverse?distance(0,0,width/2,height/2)-
          mixPercent*distance(0,0,width/2,height/2)/100:
     mixPercent*distance(0,0,width/2,height/2)/100,
     radius==0?0:blur,
     0.5, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0);
  return KeyMix(in1, in2, RGrad1, 1, "A", 100, reverse);
}
nfxDefMixer("radialWipe", "RadialWipe()");
```

7 Save all of this as a .h file in your *startup* directory.

8 As a final touch, open a ui .h file and add an on/off button for the reverse parameter:

```
nuxDefExprToggle("RadialWipe.reverse");
```

Now when you launch Shake, a new mixer in *Transition* is available for you to use.

Using the Audio Panel

9

The Audio Panel lets you import reference audio clips that you can use for timing and to generate keyframe data within your script.

About Audio in Shake

Shake supports the use of PCM AIFF and PCM Wave files in your projects. You can import one or more audio files, mix them together, extract curves based on frequency, manipulate the timing of the sound, and save the files back out again. These audio curves can also be visualized directly within the Curve Editor.

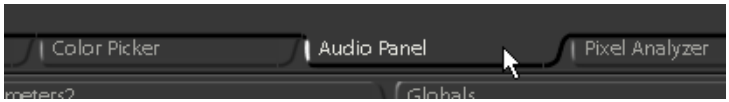
Note: Because audio playback is handled through the use of Macintosh-specific QuickTime libraries, you can only hear audio playback (either solo, or synchronized with your video) on Mac OS X systems. You can still analyze and visualize audio in Linux.

Although Shake supports a variety of frequencies and bit depths, playback is always 16-bit, defaulting to a sample rate of 44.1 kHz (the playback sample rate can be changed via the Playback Rate parameter). Independently of playback, audio is always exported at the highest quality specified in the Mixdown Options, corresponding to a 27-point symmetric Kaiser-windowed sinc interpolation.

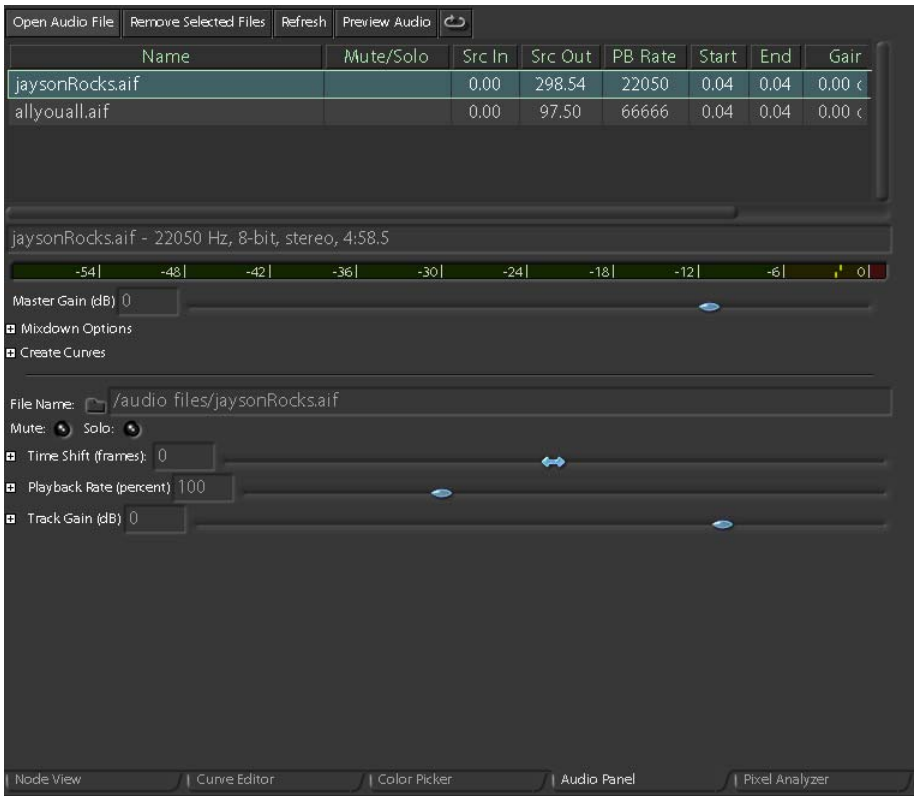
Audio and QuickTime

When you read in a QuickTime file via a *FileIn* node, the audio is not automatically imported into your project. Instead, the audio must be imported in a separate process using the Audio Panel, just like any other audio file in Shake. This means that you'll be reading in the same file twice, once via a *FileIn* node, and a second time using the Open Audio File button in the Audio Panel.

Most of Shake's audio functionality resides within the Audio Panel. To access the audio controls, click the Audio Panel tab.

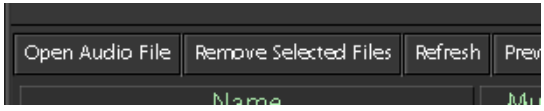


The Shake audio panel opens.



Loading, Refreshing, and Removing Audio Files

You can load both AIFF and Wave files into Shake. The first row of buttons on the top of the Audio Panel controls loading, removing, refreshing, and previewing .aif (.aiff) and .wav files.



To load an audio file into a script:

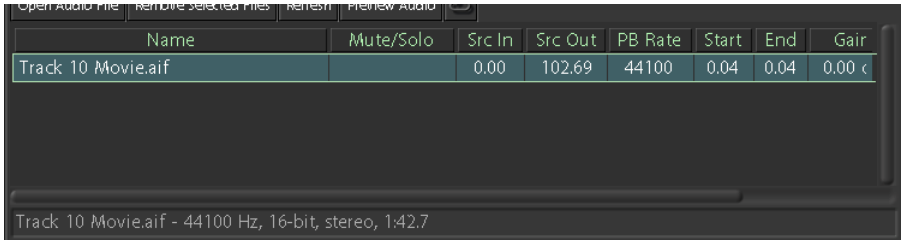
1. Open the Audio Panel.
2. In the Audio Panel, click the Open Audio File button.

3. In the Select Audio File window, select the audio file(s) you wish to import, then click OK.

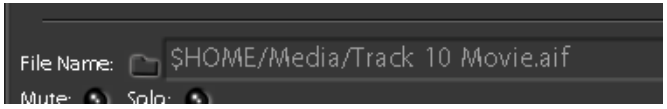
   Note: You can also double-click an audio file to import it.

   The audio file appears in the audio track list, at the top of the Audio Panel. You can import several audio files into your script—they all appear in this list.

   Details about the selected audio file appear in the audio track list.



   The path to the originating media file on disk appears in the File Name area.



   If an audio file is on an offline server or drive when a script is loaded, that audio file becomes unlinked, appearing red in the sound file list. Once the server or drive is online, you can use the Refresh command to reload the audio file. You can also use the Refresh control to update an audio file in your script that you've made changes to, simultaneous to it's being used by Shake.

To refresh a sound file used by a script:

- Click the Refresh button.

To remove an audio file from a script:

1 Select an audio file in the track list of the Audio Panel.

Note: You can Shift-select or drag to select multiple files.

2 Do one of the following:

- Click Remove Selected Files.
- Press Delete or Backspace.

The selected audio tracks are removed from your script.

Previewing and Looping Audio

You can use the Preview Audio button to listen to and edit audio tracks as they play. You can also loop an audio track within a designated time range. The preview tool works independently of other playback and previewing mechanisms in Shake. To view or sync an image sequence with audio, use the Audio Playback button on the Time Bar. For more information, see "Playing Audio With Your Footage" on page 282.

To preview an audio file in the Audio Panel:

1 To load the audio file, follow steps 1 through 3 in the "To load an audio file" section, above.

2 If necessary, perform the following optional steps:

- You can set the timeRange parameter in the Globals tab to a frame range to limit preview playback to a specific section of the audio.
- You can enable the looping control so that the preview playback loops continuously.

3 To begin playback, click Preview Audio.

If the audio clip's Time Shift subparameters (at the bottom of the Audio Panel) have been changed, these parameters will modify playback. For example, if the Source In parameter (in the Time Shift subtree) has been altered, then audio will begin previewing at the new Source In point.

While the audio plays, the Preview Audio button turns into the Stop Preview button.

4 To stop audio playback, click Stop Preview.

When you preview an audio clip, the audio meter lights up to display the clip's audio level.

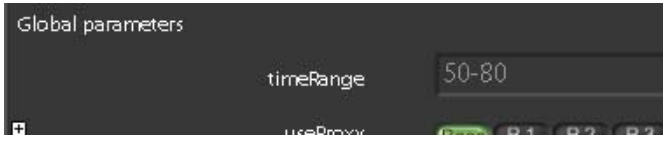
Secondary Peak Meter

You can also enable a small peak meter next to the Load/Save script buttons by entering the following line in a *ui.h* file:
gui.showTopPeakMeter = 1;

5 To change the level of the audio playing back, adjust the Master Gain (dB) slider underneath the audio meter.

To loop audio playback:

1 To loop a specific frame range, enter a frame range in the timeRange parameter of the Globals tab.



Note: If a frame range is not specified in the Globals tab, the audio preview continues to play (beyond the end of the actual audio track) until you click Stop Preview.

2 Open the Audio Panel, and toggle looping on.

3 In the Audio Panel, click Preview Audio.

The track loops within the designated frame range. Click Stop Preview to halt playback.

Muting and Soloing Tracks

If you have multiple audio tracks in your script, you can use the mute and solo controls to control which ones play back.

- Muting a track silences it.
- Soloing a track silences all tracks except for those with solo enabled.

Both of these controls are located in the parameters list below the audio track list, and can be enabled and disabled for each selected track individually.

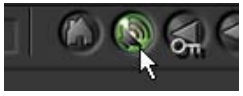
Playing Audio With Your Footage

This section discusses the tools associated with enabling audio playback with footage, viewing audio waveforms, and editing audio tracks.

Note: Depending on your hardware configuration, audio playback does not necessarily sync with the frame indicator in the Time Bar.

To enable simultaneous audio/video playback:

- Turn on the Audio Playback button, located among the playback controls next to the Time Bar.



When audio playback is enabled, clicking Play results in both the audio and video of your project playing together.

Important: Because Shake is designed primarily as a compositing application, and not a real-time editing application, audio sync is not guaranteed due to the excessive processor demands of most operations. If you want a synchronized preview of your script, use one of the Flipbook options. For more information, see "Previewing Your Script Using the Flipbook" on page 90.

Alternately, you can scrub through the audio directly in the Time Bar.

To scrub audio with the playhead:

- Hold the Command or Control key down, and drag the playhead in the Time Bar.

Viewing Audio

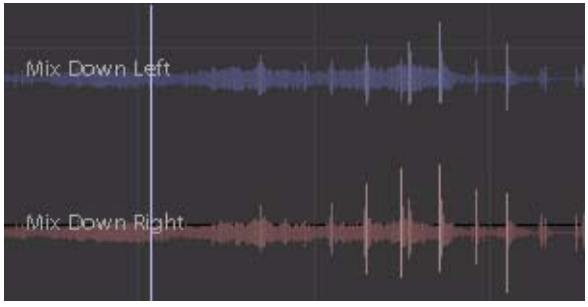
If you are syncing animated parameters in your script with specific audio cues, you can display the waveform of your audio in the Curve Editor. If you have multiple audio files loaded into Shake, the Curve Editor displays the overall mix.

To view an audio waveform in the Curve Editor:

- In the Curve Editor, click the Draw Waveform button.

  This control toggles curve display off and on.

When enabled, the waveform is displayed in the Curve Editor.



Slipping Audio Sync in Your Script

If you need to resync the audio tracks in relation to the visuals in your project, you can slip each audio track in your project back and forth in time.

To slip an individual audio track in time:

1 In the Audio Panel, select a track in the track list, and enable solo.

The waveform for the track is redrawn in the Curve Editor.

2 Do one of the following:

- Press Shift-Option or Shift-Alt and drag in the Curve Editor.
- In the Audio Panel, enter the slip amount (in frames) in the Time Shift value field.

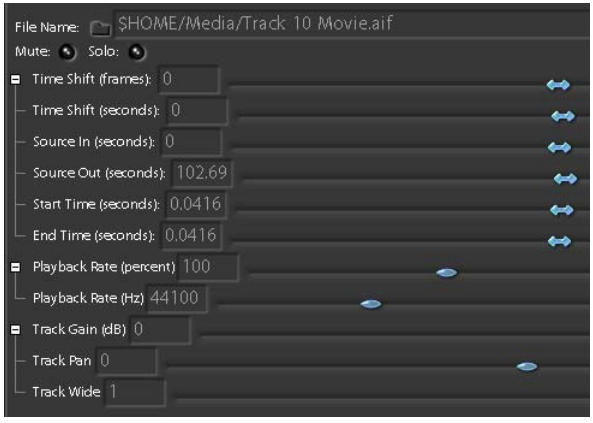
To slip all audio tracks in time at once:

- Press Shift-Option or Shift-Alt and drag in the Curve Editor.

Note: Although multiple tracks can be highlighted in the track list (and deleted), you cannot selected multiple tracks in the sound file list for slipping, muting, or setting to solo. Only the highlighted track with the line around it is actually selected.

Time Shift Subparameters

The following table lists the time shift subparameters associated with every audio track. These parameters let you adjust the timing of audio clips used in your script.



Time Shift (frames)
Sync slip control in frames. To interactively drag sound in the Curve Editor, press Shift-Opt or Shift-Alt and drag in the Curve Editor.

Time Shift (seconds)
Sync slip control in seconds.

Source In (seconds)
Beginning point of the clip, listed as seconds.

Source Out (seconds)
End point of the clip, listed as seconds.

Start Time (seconds)
Beginning point of the clip, listed as frames.

End Time (seconds)
End point of the clip, listed as frames.

Playback Rate Subparameters

These parameters allow you to specify the rate at which audio plays back, resampling the audio tracks if necessary to make them conform. Audio playback is only possible on computers using Mac OS X.

Playback Rate (percent)
Same as playback rate, but allows you to enter a specific cycle rate.

Playback Rate (Hz)
Same as playback rate, but allows you to enter a specific cycle rate.

Track Gain Subparameters

These parameters let you adjust the relative volume and stereo imaging of each audio track you've imported into your project. Mixing the different channels of audio allows you to adjust the loudness of a voiceover track in relation to a separate background music track.

Track Gain (dB)
A decibel gain.

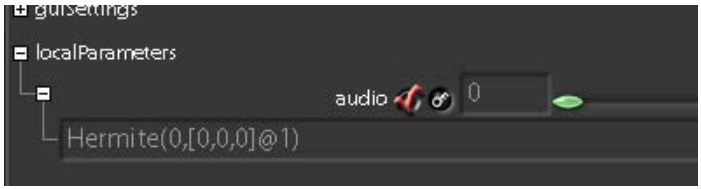
Track Pan
Adjusts the stereo effect by panning a track toward or away from one ear.

Track Wide
Controls how much uniqueness exists in the left and right channels, giving a feeling for wider stereo.

Extracting Curves From Sound Files

The Create Curves subtree gives you control over sampling of the audio to be converted into a curve. These curves appear in the Globals tab as a local parameter, and can then be used by other functions in Shake as standard expressions. This is an extremely powerful feature, and can be used to synchronize nearly any animated parameter to the waveform of your audio.

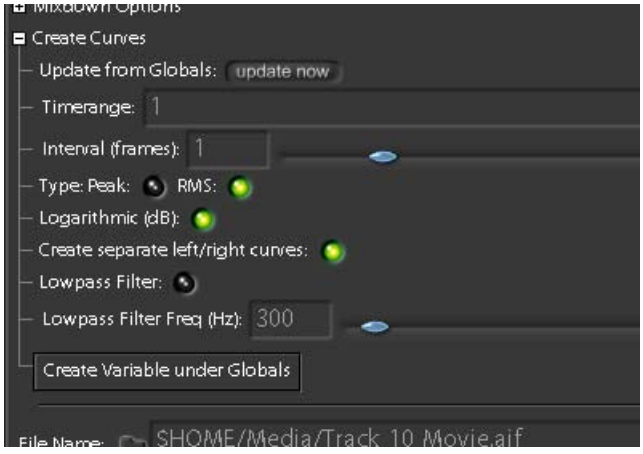
The parameters located in the Create Curves subtree let you analyze the current audio mix, creating a keyframed curve that's stored as the Audio parameter, within the localParameters subtree of the Globals tab.



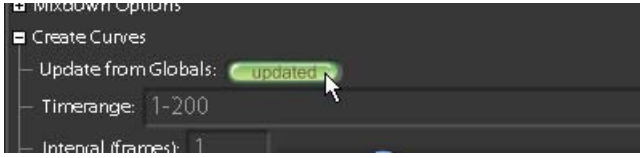
The audio parameter can be referenced as an expression from any other parameter in your project.

To create an audio curve:

1. Open the Audio Panel, and import one or more audio files into your project.
2. Open the Create Curves subtree.



3. Click the Update from Globals button to set the time range to be analyzed to match the timeRange parameter of your project.



4. Adjust the other Create Curves parameters as necessary (see the next section for more information).
5. To create the audio parameter, click the Create Variable Under Globals button.

A progress bar appears to show you how long this process takes.

Opening the Globals tab reveals the Audio parameter that has been created, underneath the localParameters subtree. This parameter is now ready for use as an expression from within other parameters in your project.

Create Curves Options

These parameters in the Create Curves subtree of the Audio Panel let you customize how keyframe information is extracted from the audio waveform.

Update from Globals

Indicates if timing for the audio export should come from script globals or from the audio tab. Click "update now" to read settings from the Globals tab.

Time Range

The frame range of the audio to be extracted.

Interval (frames)

A key is entered every N frames, controlled by the Interval parameter.

Type

This parameter defines how the volume of the audio in your project is analyzed.

- *Peak* means the generated value is the highest absolute value the waveform reaches during that interval.
- *RMS* ("Root Mean Square"), on the other hand, uses the square root of the mean of the squares on the absolute values of the waveform over the interval. In other words, each absolute displacement value in the interval is squared, the average of all those squared values is calculated, and its square root is taken as the representative value for that interval. The RMS method is said to be more representative of the actual perceived volume of an interval of digital audio.

Logarithmic (dB)

When deactivated, the returned value is the actual value from the audio file. The sample values are normalized from -1.0 to 1.0, with 1.0 =0 dBFS. Therefore, the usual range in this mode is 0.0 to 1.0. Values may exceed 1.0 if the audio is clipping (exceeding 1.0, or 0 dBFS).

If "logarithmic" mode is on, the value generated is converted to a dB scale, normalized linearly over 90 decibels from 0.0 to 1.0. A value of 0.0 would thus represent -90 dBFS (or lower), and 1.0 would represent 0 dBFS.

For example, if in peak mode, and the peak audio value over an interval is 0.5 (approximately -6 dBFS), the value entered with logarithmic mode off would be 0.5. With logarithmic mode on, it would be (-6 + 90) / 90 = 0.9333.

Create separate left/right curves
Either one curve is created, or left and right channels are sampled and two curves are created.

Lowpass Filter
Activates the LPF, as described below.

You can use the lowpass filter to restrict the frequency range that Shake analyzes to create the resulting curve data. For example, you can filter out the majority of the high-frequency content of a song, allowing the bass (such as drums) to become the primary source of curve data.

Lowpass Filter Freq (Hz)
This control determines the cutoff frequency for the lowpass filter. All frequencies at or above this frequency are cut off completely when the filter is activated. This setting is useful for analyzing strictly low-frequency content, such as bass drums, rumble, earthquakes, and other subwoofer-shaking phenomena. When the filter is active, its effect may be heard during playback. (Use "Preview Audio" to hear the effect of sliding the filter freq in real time.) The lowpass filter setting does not affect mixdown files or QuickTime renders.

Exporting an Audio Mix

If necessary, you can export the audio mix you've created within your project as a separate audio file. If you've created an audio mix that must accompany an image sequence that you're outputting, the audio mix must be output separately.

Note: Rendering a QuickTime file via a *FileOut* provides the option to export the audio and video within a single media file.

Mixdown Options

The following parameters are available in the Mixdown Options subtree:

Update from Globals
Indicates if timing for the audio export should come from script globals or from the Audio Panel. Clicking update now reads settings from the Globals tab.

Time Range
The number of frames to be exported.

Mixdown File Name
The output file name.

Sample Rate, Bit Depth
The output sample rate and bit depth of the output file.

Resampling Quality
When input clips are adjusted and scaled in time, their sound samples must be interpolated to the output sampling rate. How closely this is done is determined by the quality parameter. "Highest" should be used if intended for broadcast. For the technically minded, this corresponds to a 27-point symmetric Kaiser-windowed sinc interpolation.

Clipping Info
When lighted, indicates that the right or left audio channel is clipping.

Save Mixdown
Click Save Mixdown to render out a single audio file.

Track Wide
Controls how much uniqueness exists in the left and right channels, giving a feeling for wider stereo.

Parameter Animation and the Curve Editor

10

Shake has a flexible keyframing interface for animating nearly any parameter in your script. This chapter covers how to create keyframed animation, as well as how to manipulate keyframed data using the Curve Editor.

Animating Parameters With Keyframes

Several controls exist throughout Shake that allow you to animate the parameters of various nodes over time. The Viewer, Parameters tabs, Time Bar, and playback controls all have controls specifically for creating, modifying, and removing keyframes.

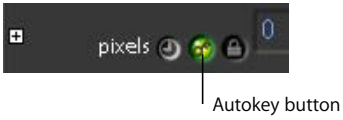
Animating Parameters Using Keyframes

It takes a minimum of two keyframes to animate a parameter. Each keyframe you create stores a particular value for a parameter. When you've keyframed a parameter, Shake automatically interpolates its value at each frame, from the first keyframe to the last, creating animation.

To animate one or more parameters:

1 Prior to animating a parameter, do one of the following:

- In the Parameters tab, turn on the Autokey buttons for the parameters you want to animate.



When a specific parameter's Autokey button is enabled, keyframes are created whenever you modify that parameter.

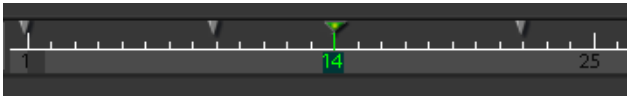
- To keyframe parameter changes you make using a node's Viewer controls, turn on the Autokey button in the Viewer shelf.

Whenever you first enable keyframing, a keyframe is automatically created at the current position of the playhead in the Time Bar for the affected parameters.

2 To create a second keyframe, move the playhead in the Time Bar to another frame, and then change the value of the parameter.

A keyframe appears in the Time Bar to show this change.

When a parameter is animated, keyframes appear in the Time Bar to indicate each change to that parameter. If the playhead is directly over a keyframe, that keyframe is green to indicate that it's "selected." In other words, changes you make to that parameter simply modify the current keyframe, instead of creating a new one.



Note: You can also edit keyframes in the Curve Editor by clicking the parameter's Load Curve button (the clock-shaped button to the left of the Autokey button in the Parameters tab).

To delete a keyframe:

- In the Time Bar, move the playhead to the desired keyframe, then do one of the following:
  - Click the Delete Keyframe button in the Viewer or Curve Editor.

  - Right-click the parameter from which you want to delete a keyframe, then choose Delete Current Key from the shortcut menu.
  - Click the Load Curve button for the parameter that contains the keyframe you want to delete, then click the keyframe in the Curve Editor and press Delete.

Rules for Keyframing

How keyframes are created and modified depends on two things—the current state of the Autokey buttons, and whether or not there's already a keyframe for that parameter in the Time Bar or Curve Editor at the current position of the playhead.

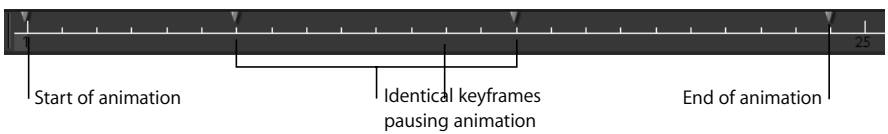
When animating any parameter, the following rules apply:

- When the Autokey button is off and you adjust a parameter that has no keyframes, you can freely adjust it at any frame, and all adjustments are applied to the entire duration of that node.
- When you adjust a parameter that has at least one keyframe already applied, you must turn on the Autokey button before you can make further adjustments that add more keyframes.
- If the Autokey button is off, you cannot adjust a parameter at a frame that doesn't already have a keyframe. If you try to do so, the change you make to the parameter doesn't stick, and that parameter goes back to its original value when you move the playhead to another frame.

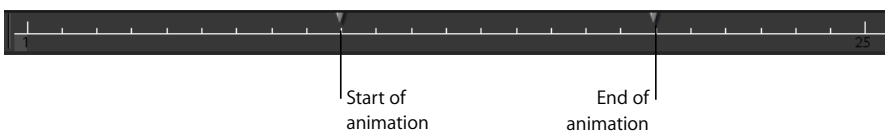
  Note: If you've made a change that you want to keep, turn on the Autokey button before you move the playhead to add a keyframe at that frame.

Adding Blank and Duplicate Keyframes to Pause Animation

If you want a parameter to be still for a period of time before it begins to animate, insert a pair of identical keyframes at the start and end frame of the pause you want to create.



If you want to delay an animated effect for several frames beyond the first frame, insert a keyframe with no animated changes at the frame where you want the animation to begin, then modify the shape at the keyframe where you want the animation to end.



To manually add a keyframe without modifying the parameter:

- Turn the Autokey button off and on again.

  A keyframe is created for the current parameter. If the parameter is already animated, the state of the parameter at the position of the playhead in the Time Bar will be stored in the new keyframe.

Navigating Among Keyframes in the Time Bar

Once you've created a number of keyframes, two keyframe navigation controls let you move the playhead from keyframe to keyframe, making it easy to jump to and modify existing keyframes.

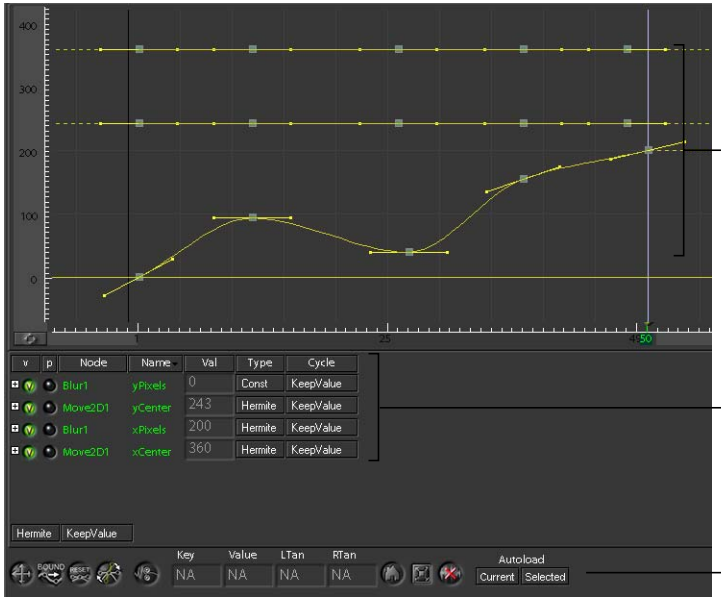
Go to previous keyframe

Go to next keyframe

Note: These two controls only work when the playhead is *within* a range of two or more keyframes in the Time Bar. If the playhead is located either before the first keyframe, or after the last keyframe, these controls have no effect.

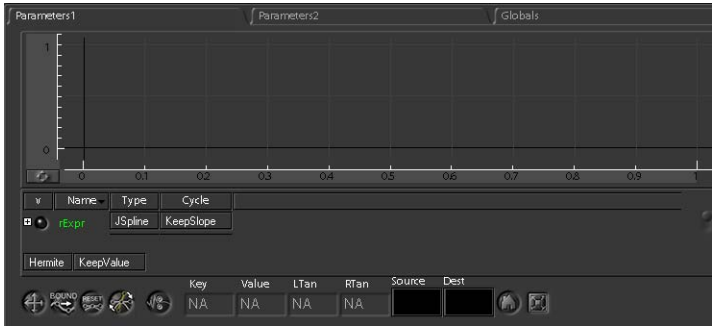
Using the Curve Editor

While the Time Bar is adequate for creating keyframes for one parameter at a time, the Curve Editor gives you a much more complete environment in which to create, move, and otherwise modify keyframes. The Curve Editor also provides the only place where you can see and modify the animation or Lookup curves that represent the interpolated values that lie in between each keyframe.



Parameters can be represented by any one of a number of different curve types, each of which gives you a different way of controlling how a parameter's values are interpolated from keyframe to keyframe. You can change a curve's type and cycling mode at any time. For more information on specific curve types, see "More About Splines" on page 316.

In addition to the Curve Editor tab, there are also curve editors that appear within the Parameters tabs. These are used to edit lookup-style curves that generally relate to color correction. These are the *Lookup*, *LookupHLS*, *LookupHSV*, and *HueCurves* nodes.

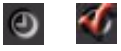


Loading and Viewing Curves in the Editor

By default, the Curve Editor appears empty. You have to specify one or more parameters to be displayed in the Curve Editor. If multiple curves appear in the Curve Editor, they overlap.

To load curves into the Curve Editor from the Parameters tabs:

1. Load a node into one of the Parameters tabs.
2. To display a parameter's curve in the Curve Editor, do one of the following:
   - Inside of the Parameters tab, click a parameter's Load Curve button.

     A red checkmark appears to let you know the button is enabled, and that parameter is loaded into the Curve Editor.

     Note: Just because a parameter is loaded into the Curve Editor doesn't mean the parameter is animated.
   - Click a parameter's Autokey button to create a keyframe at the current playhead frame.

Note: Whenever you turn on an Autokey button, the corresponding parameter's curve loads into the Curve Editor.

In the following example, even though the pan and angle parameters are both animated, the angle parameter is the only one that's actually displayed in the Curve Editor.

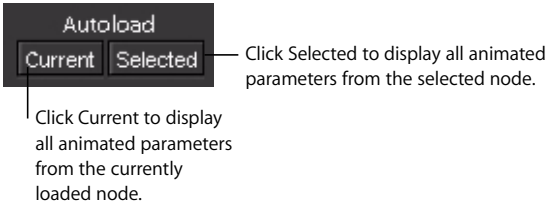
Controlling Curves Displayed in the Curve Editor

There are several controls that allow you to control the visibility of curves from within the Curve Editor.

Autoload Controls

Use the Autoload curve viewing controls within the Curve Editor to choose which parameter curves are automatically loaded into the Curve Editor. There are two options:

- Turn on *Current* to show all the parameters that are animated within a node that's loaded into one of the Parameters tabs, regardless of whether or not the Load Curve controls are enabled.
- Turn on *Selected* to show all animated parameters from all nodes that are selected in the Node View, whether or not any nodes are loaded into one of the Parameters tabs.



Visibility and Persistence Controls

In the loaded parameters list, additional controls let you toggle the individual visibility and persistence of parameters in the Curve Editor.

To toggle the visibility of individual curves, do one of the following:

- Click a parameter's Visibility button to turn the display of a curve on and off in the Curve Editor, keeping it in the list.

- Select a parameter in the parameters list, and press V to toggle its visibility.
- Click a parameter's Persistence control to keep that curve loaded in the Editor regardless of the currently selected Autoload settings.

To remove a curve from the Editor:

- Select the curve in the parameters list and press Delete or Backspace, or click the Load Curve control in the node's parameters.

The curve is removed from the Editor, and the Load Curve control is disabled.

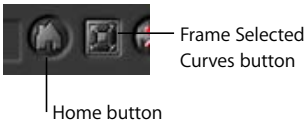
> ***Warning:*** If you rename a node that is already loaded into the Curve Editor, the new name is not automatically updated in the Curve List. To display the new node name, remove the node from the Curve Editor, then reload it into the Curve Editor.

Navigating the Curve Editor

There are many controls you can use to move around the Curve Editor.

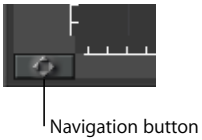
Useful controls for automatically framing curves:

- To frame one or more selected curves to the size of the Curve Editor, press F.
- To frame all curves to the size of the Curve Editor, press the Home key, or click the Home button.
- To frame only the selected curves, click the Frame Selected Curves button.



To pan around within the Curve Editor, do one of the following:

- Press the middle mouse button, or Option-click or Alt-click, and drag.
- Click the Navigation button, and drag to pan around.



To zoom into and out of the Curve Editor, do one of the following:

- Press the + or - key.
- Middle-click or Control-Option-click or Control-Alt-click.
- With the pointer over the Navigation button, Control-click and drag.

To zoom the Curve Editor tab to take up the full screen:

- Press the Space bar.

Curve Editor Right-Click Menus

The shortcut menu within the Curve Editor has many controls you can use to manage curve visibility, and other options for curve editing.

Shortcut Menu	Description	Keyboard
Add All Curves	Adds all animated curves into the Curve Editor.	
Remove Curves	Removes selected curves from the Curve Editor. Does not delete the animation.	Delete or Backspace
Visibility > Hide Curves	Turns off visibility of selected curves. You can also toggle visibility in the parameters list.	
Visibility > Show Curves	Shows selected curves. You select curves in the parameters list prior to this function.	

Shortcut Menu	Description	Keyboard
Visibility >Toggle Visibility	Inverts the visibility of all selected curves.	
Select > All Curves	Selects all curves in the parameters list.	Command-A Control-A
Select > CVs	Selects all keyframes on active curves. To select all keyframes on all loaded curves, press Command-A or Control-A and then Shift-A.	Shift-A
Display Timecode	Toggles the time display from frames to timecode.	T
Sticky Time	When enabled, the current frame is set to the keyframe you are modifying.	S
Time Snap	When enabled, keyframes snap to frames, rather than float values.	
Display Selected Info	Displays data on selected curves and keyframes when active.	

The Curve Editor Buttons

The following table describes the Curve Editor buttons.

Button		Description
	Set Horizontal/ Vertical Lock	Locks off movement on the X or Y axis. You can also press X to restrict movement to the X axis, or press Y to restrict movement to the Y axis. Press the key again to reenable movement.
	Keyframe Move Mode	This mode determines the behavior when keyframes are moved left and right past other nonselected keys. This behavior is discussed in "Using Keyframe Move Modes" on page 307.
	Reset Tangents	The Shake Hermite tangents are automatically set to the tangent of the curve. Modifying keys adjusts the tangents of neighbor keyframes. However, if you manually adjust a tangent, Shake recognizes this and disables this automatic adjustment. Click Reset Tangents to reset the tangents to their unset state.
	Flatten Tangents	Sets a Hermite-curve tangent horizontal to ensure smooth ease ins and ease outs. You can also press H.
	Apply Curve Operation	Applies an operation to the curve from a pop-up window. These functions are detailed in "Applying Operations to Curves" on page 309.
	Home	Frames all curves.
	Frame Selected Curves	Frames selected keyframes/curves.

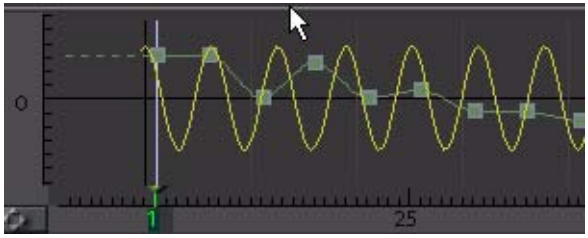
Button		Description
	Display Waveform	Displays the waveform of any currently loaded audio files from the Audio Panel.
	Color controls	These buttons appear in the Parameters tab for the *Lookup*, *LookupHSV*, and *LookupHLS* functions. They allow you to pick an input color (RGB, HSV, or HLS) and match it to a different color. Only visible curves receive keyframes on their curves. For example, if you only want to modify luminance, ensure the hue and saturation curves are disabled in a *LookupHLS* node.

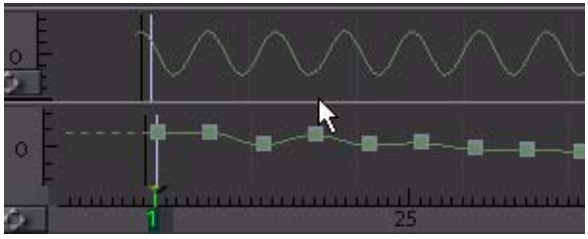
Splitting the Curve Editor

You can separate the Curve Editor into two horizontal panes. This is useful when you have two or more curves with completely different Y scales, such as the pan and scale curves from a *Move2D* node. Each pane has its own set of visibility toggles, so you can disable specific curves in one pane, and enable them in another. The V key is particularly useful here, since the active pane is the last pane your pointer crossed.

To split the Curve Editor:

- Click the top edge of the Editor window and drag down.





Working With Keyframes

This section presents different methods for adding and modifying keyframes within the Curve Editor.

Adding Keyframes

There are many different methods you can use to add keyframes to a curve in the Curve Editor.

To add keyframes to a curve by modifying a parameter:

- In the node's Parameters tab, click the Autokey button. Modifying a value when Autokey is enabled creates a keyframe at the current position of the playhead.

To create keyframes at the position of the playhead on every loaded curve:

1. In the Parameters tab, click the Load Curve button for each parameter you want to keyframe.

2. Move the playhead to the frame where you want to create a keyframe.
3. Click the Autokey button.

Keyframes are created for every curve that's currently loaded in the Curve Editor. For example, if the *Move2D* node has its x,yPan, x,yScale, angle, and x,yCenter parameters set for keyframing, keyframes are created on the curves of each of these parameters.

Note: Creating keyframes in this manner overrides any expressions within those parameters.

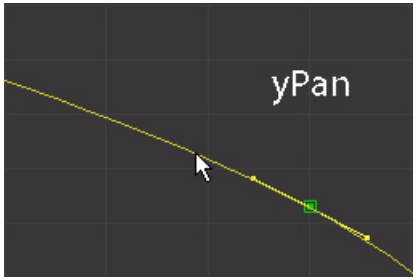
To insert a keyframe anywhere on a curve, do one of the following:

- Shift-click a segment. This lets you insert new keyframes at frames that aren't at the current position of the playhead
- Position the pointer over a curve so that it's highlighted, and press K.

In the Curve Editor parameters list, a keyframe is created whenever you enter or modify a value in the Val value field. However, the parameter's Autokey button (in the Parameters tab, below the Curve Editor) must be activated first.

The *Lookup*, *LookupHLS*, and *LookupHSV* functions have color control pairs embedded in their dedicated Curve Editors that remain inside the Parameters tab. You can use these to enter keys. However, these keys are not related to time, but, rather to a particular color channel. Therefore, these keys become points on the color-correction curves. For more information, see "The Curve Editor Buttons" on page 299.

Note: In the Curve Editor, when the pointer passes over a curve, the curve name is highlighted; when the pointer passes over a keyframe, the keyframe values are displayed.



Selecting Keyframes

You can edit a group of keyframes together by selecting them as a group. You can also add and remove keyframes from a previously selected group.

To select one or more keyframes, do one of the following:

- Click a single keyframe.
- Shift-click to select multiple keyframes.
- Drag to select a group of keyframes with a selection box.
- Press B while dragging to create a persistent manipulator box that you can use to manipulate a range of keyframes.

  This box allows you to scale the keyframe group in X and Y, only X, or only Y. The manipulator box also allows you to move the group within the boundaries of the surrounding (not selected) keyframes. For more information on this method, see "Using the Manipulator Box" on page 304.

To add keyframes to the current selection:

- Shift-click a point on a curve to add an additional keyframe.

To select all keyframes on a curve:

- Position the pointer over the curve so that it's highlighted (or select the curve in the parameters list), then press Shift-A.

To select all keyframes in the Curve Editor:

- Press Command-A or Control-A (select all curves), then press Shift-A.

To deselect one or more keyframes:

- Command-drag or Control-drag to remove keyframes from the current selection of keyframes.

Note: To remove keyframes from a group of selected keyframes within a manipulator box, press Shift and click the keyframes. For more information, see "Modifying Keyframes" on page 303.

To deselect all keyframes:

- Click an empty area of the Curve Editor.

Deleting Keyframes and Curves

There are several different ways of deleting keyframes in the Curve Editor.

To delete one or more keyframes, do one of the following:

- In the Curve Editor, select the keyframes you want to delete, then press Delete.

  Note: In Mac OS X, you must use the Delete key located near the Home and End keys. If you press the Delete key located under the F13 key, the entire curve will be deleted.

- Move the playhead to the location of the keyframe, then click the Delete Keyframe button (in the Viewer shelf).

The Delete Keyframe button only deletes keyframes related to the onscreen controls.

To delete an entire curve, do one of the following:

- Position the pointer over a curve, or select the curve in the parameters list, then press Shift-A (to select the points), then press Delete.
- In the parameters list, convert the curve to the Const (constant) curve type.
- In the node's Parameters tab, right-click in the parameter value field, then choose Clear Expression from the shortcut menu.

  Note: When a curve is deleted, it is replaced with a constant curve (set to the value of the curve at the point in time the curve was deleted).

To delete keyframes from the Parameters tab:

1. Move the playhead to the keyframe you want to delete.
2. Right-click the parameter's name, then choose Delete Current Key from the shortcut menu.

To delete keyframes related to onscreen controls currently in the Viewer:

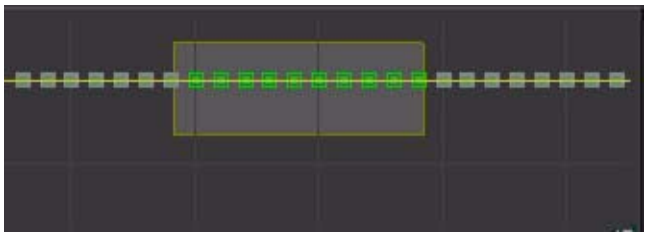
1. Move the pointer over the Viewer.
2. Press the Delete or Backspace keys.

Modifying Keyframes

You can modify keyframes by selecting and moving the keyframes, creating a manipulator box, modifying keyframes numerically, or using the value fields in the Curve Editor parameters list.

Using the Manipulator Box

You can use the manipulator box to move or scale a group of keyframes. The advantage the box has over simply selecting keyframes is that you can see the scale borders.



To use the manipulator box:

- Hold down the B key (for box) and drag to select a group of keyframes.

  The light gray manipulator box appears around the selected keyframes.

To move the selection:

- Position the pointer within the box, then drag.

To scale the selection in both the X and Y axes:

- Position the pointer at the corner of the box and drag. The box is scaled in X and Y around the opposite corner you select—if you grab the upper-right corner, the center of scaling is the lower-left corner.

To scale the selection in the X axis:

- Position the pointer along either side edge of the box, then drag.

To scale the selection in the Y axis:

- Position the pointer at the top or the bottom edge of the box, then drag.

Note: Once you make a selection with the manipulator box, clicking a keyframe or clicking outside of the box deselects the box.

To add to the keyframes selected in the manipulator box:

- Press Shift and click the additional keyframes.

To remove selected keyframes from the group:

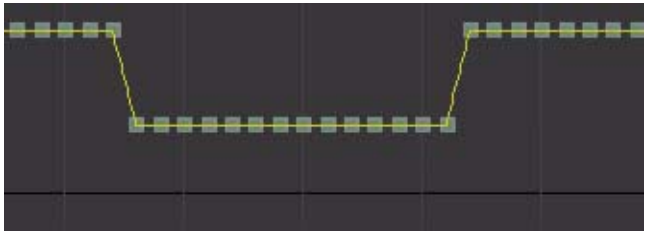
- Press Shift and click the keyframes you want to delete.

Using Transform Hot Keys

You can also move a group of keys using keyboard shortcuts—you are not obliged to select the keys with the manipulator box.

The following keyboard shortcuts let you make curve adjustments:

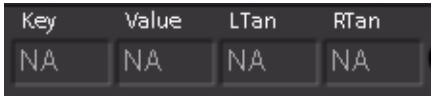
- *Move*: To move selected keys, press Q and drag.



- *Scale*: Press W and drag; the first point clicked is the scaling center.
- *Drop-off*: Press E and drag; a drop-off occurs on the pan.

Using the Keyframe Value Fields

To modify a keyframe numerically, you can also enter values in the fields at the bottom of the Curve Editor window.



- The Key field is the time of the keyframe.
- The Value field is the value of the keyframe.
- The LTan and RTan fields control the tangents. If the tangents are set to 0, the keyframe is flattened (on a Hermite curve). You can also press H to set horizontal tangents.

The value field displays "-----" when multiple keyframes are selected. To set all keyframes to that value, enter a number into the Value field.

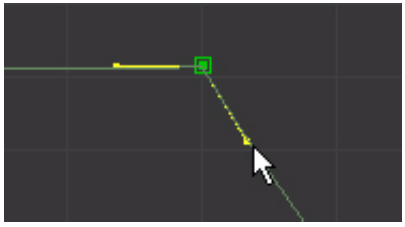
In the parameters list, the Val field displays the value of the curve at that point in time. You can also modify the value of a keyframe at that point in time in the Val value field. For the value to be saved, the Autokey button must be activated in the Parameters tab.

Editing Bezier Curve Tangents in the Keyframe Editor

Similar to control points on a shape in the Viewer, Bezier points on a curve have tangents that can be edited.

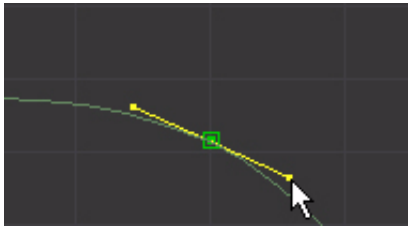
To break the tangents of a keyframe:

- Press Control and drag a tangent end.



To rejoin a tangent:

- Press Shift and click the tangent end.



To reset a tangent:

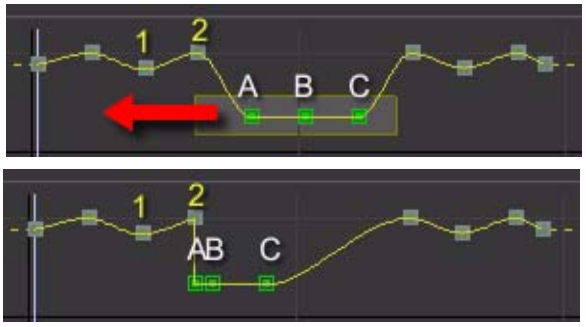
- Select the keyframe and click the Reset Tangents button.

Using Keyframe Move Modes

In the Curve Editor, there are four keyframe move modes—Bound, Interleave, Push, and Replace—that are activated by the four states of the Keyframe Move Mode button. These modes control the behavior of the keyframes when the keyframes are moved left or right past non-active keyframes. To change modes, click the Keyframe Move Mode button in the bottom-left corner of the Curve Editor.

Bound

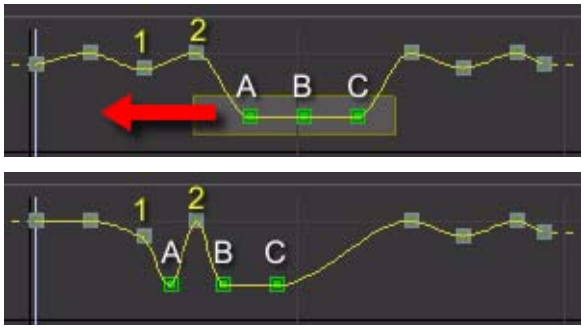
When the Bound mode is set, the movement of a selected range of keyframes (whether contiguously or noncontiguously selected) is clamped by the adjacent keyframes. In the following example, keyframes A, B, and C are selected (highlighted in green) and moved left in the Curve Editor.



When moved, the selected keyframes cannot move beyond the adjacent keyframe in the curve, so keyframe A stops at the frame occupied by keyframe 2, and the distance between A and B shrinks. If you continue to drag left, keyframes A, B, and C are placed on the same frame.

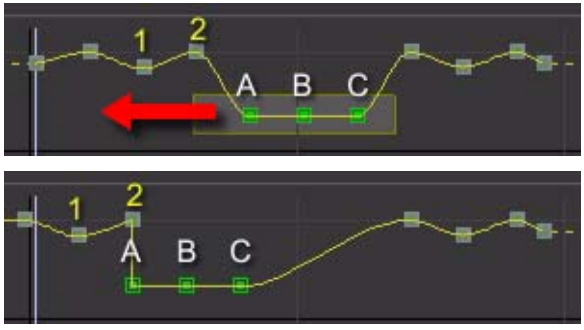
Interleave

When the Interleave mode is set, the selected keyframes jump over the adjacent non-selected keyframes to the next segment of the curve.



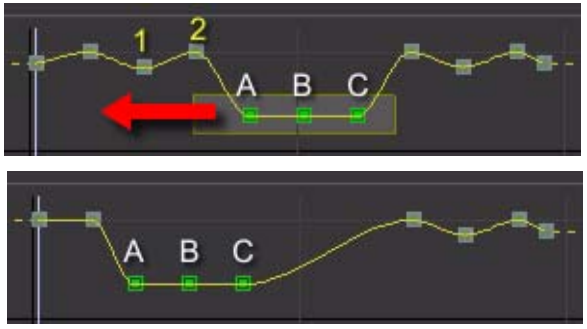
Push

When the Push mode is set, the selected keyframes push the other keyframes along the curve. In the following example, the selected keyframes are pushed to the left of the Curve Editor. Therefore, keyframe A pushes keyframes 1 and 2, as well as all other keyframes to the left of keyframe 1.



Replace

When the Replace mode is set, the selected keyframes replace the adjacent non-selected keyframe(s). In the following example, keyframes A, B, and C have slipped past the position of keyframes 1 and 2, removing them from the curve.



Applying Operations to Curves

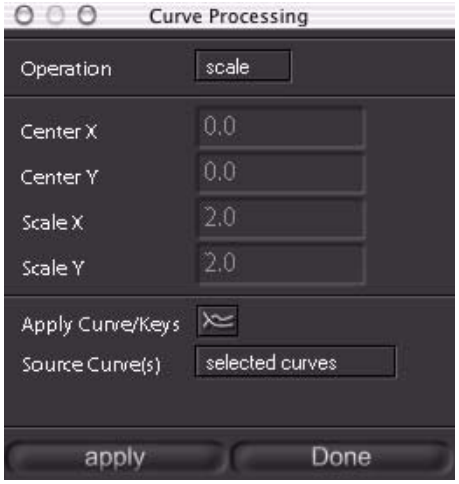
To apply operations such as Smooth and Jitter to curves or keyframes, use the Apply Operation button, located in the lower-left corner of the Curve Editor.

To apply an operation to a curve or to keyframes:

1. Select the curve from the parameters list, or, if applying the function to keyframes, select the keyframes in the Curve Editor.
2. In the Curve Editor, click the Apply Operation button.

3. In the Curve Processing window, select your operation.

In the following example, the "scale" operation type is selected.



4 Where appropriate, enter the value(s) for the expression in the value fields.

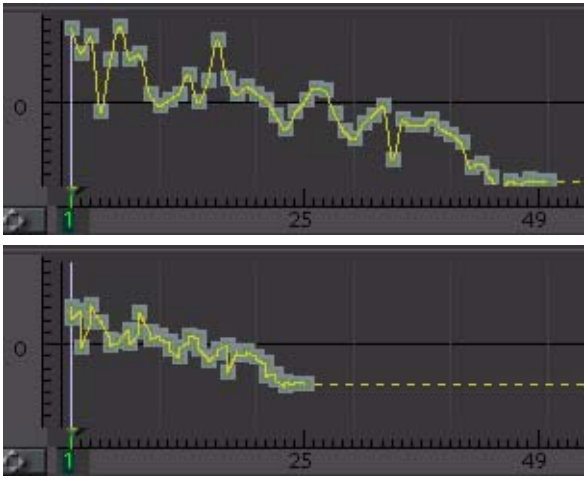
5 Do one of the following:

- To apply the operation to a selected curve, make sure that the Apply Curve/Keys button is set to Curve, then click Apply.

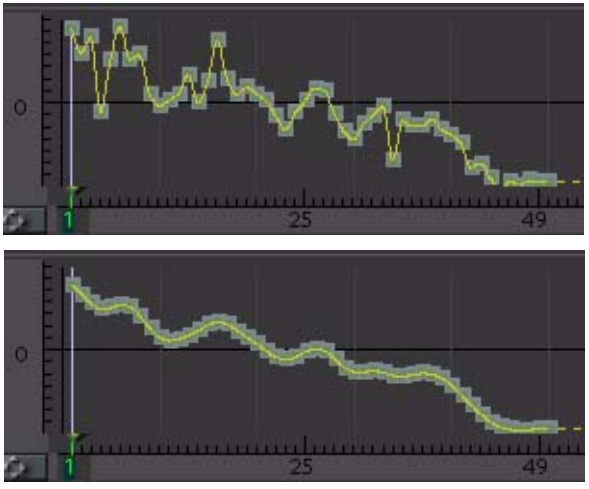
- To apply the operation to selected keyframes, make sure that the Apply Curve/Keys button is set to Keys, then click Apply.

The selected operation is applied to the selected curve or keyframes. These operations include the following:

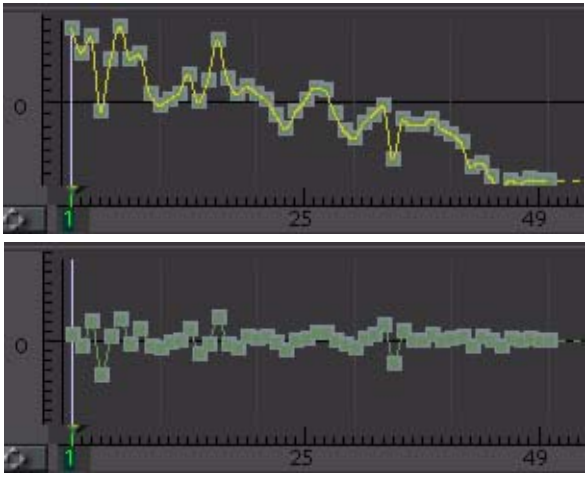
- *scale*: You can manually scale a curve using a manipulator box. This scale function in the Curve Processing window, however, allows you to enter specific scaling values. Enter the curve center and values. The following curve has a center of 0, 0 and .5, .5 for the scale values.



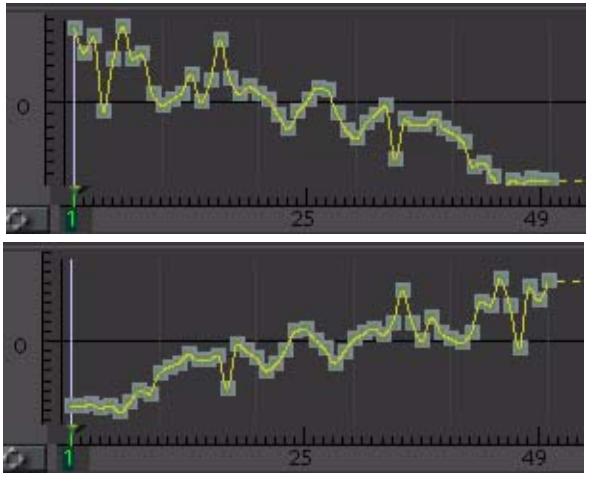
- *smooth*: "Blurs" the curve by the Amount value, the value that indicates how many neighbor keyframes are calculated in the smoothing. The higher the number, the smoother the result. In the following example, the second curve is the result of a smooth Amount of 10 applied to the first curve.



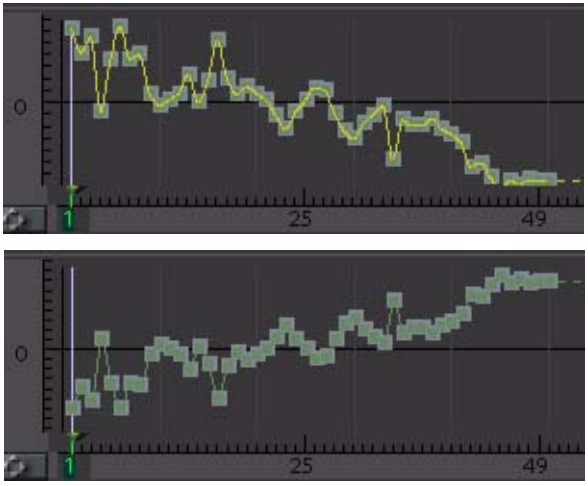
- *jitter*: The opposite of smooth, jitter removes all values except for the noise using the formula (*Unmodified Curve - Smoothed Curve = Jitter*). Once the function is applied, the curve snaps down to approximately the 0 value, so the curve may disappear (press F or click the Home button to reframe the Editor). This is useful for stabilizing a jerky camera move. You can keep the overall camera move, but remove the jerkiness. The vertical scale of this image is much smaller than the first example snapshot.



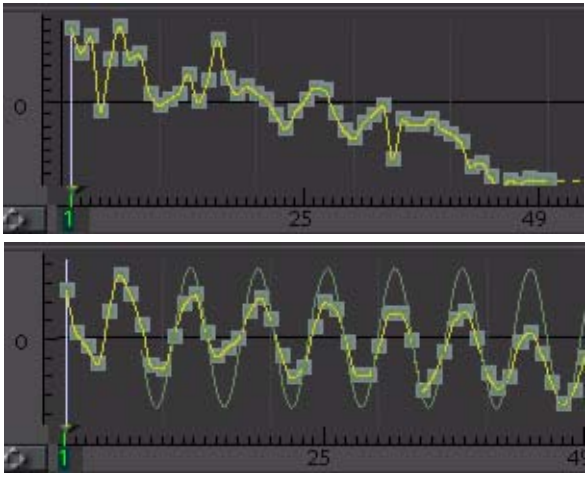
- *reverse*: Makes the curve go backward in time.



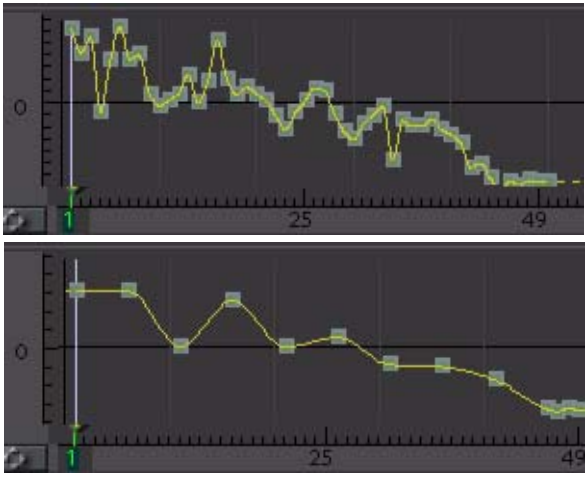
- *negate*: Flips the curve around the 0 point, so a value of 300 turns into a value of -300. Again, the curve may disappear, so press F or click the Home button to reframe the Editor.



- *average*: Allows you to average two curves together. When the Operation mode is set to average, the Dest Curve button appears, allowing you to select a second curve. Click this button to select the curve that is averaged with the current curve. In the following example, the random curve was averaged with a cos expression.



- *resample*: Replaces the curve or expression with a new sequence. This is useful for two purposes. First, you "bake" an expression, turning it into a keyframe curve. Second, you can adjust the number of keyframes that are on a curve. To set the resample, enter a frame range. For example, set *1-50* to enter 1 keyframe per frame from frames 1 to 50; *1-50x10* to enter only 5 keyframes every 10 frames, and so on.



Copying and Pasting Keyframes

You can copy and paste keyframes from one curve to another curve.

Note: You cannot simultaneously copy and paste keyframes from multiple curves.

To copy and paste a keyframe:

1. In the Curve Editor, select the keyframe you want to copy and press Command-C or Control-C.
2. Position the pointer over the curve (the same curve or a separate curve) at the time you want to paste the keyframe, and press Command-V or Control-V.

   The keyframe is pasted at the position of the playhead.

To copy and paste multiple keyframes on a single curve:

1. In the Curve List, select the curve that contains the keyframes you want to copy.

   Note: To select a curve, you can also position the pointer over the curve in the Curve Editor. The curve name is displayed and the curve is highlighted in the Curve Editor.
2. Press Shift-A to select the keyframes on the selected curve.
3. Press Command-C or Control-C.
4. In the Curve List, select the curve into which you want to paste the keyframes.

5 In the Curve Editor, position the playhead at the frame you want to paste the keyframes (the first keyframe pastes at the position of the playhead).

6 Press Command-V or Control-V to paste the keyframes.

The keyframes are pasted at the position of the playhead.

Modifying Curves

You can modify a curve type, and its repetition mode, as well as apply filter effects (smooth, jitter extraction, and so on) to a curve.

To change a curve type:

1 Select the curve (drag over the curve, or select the curve in the Curve Editor parameters list).

2 In the parameters list, choose a curve Type and select Hermite, Linear, CSpline, JSpline, NSpline, or Step curve. The most commonly used curves are Hermite, JSpline, and Linear.

For more information on curve types, see "More About Splines" on page 316.

Note: You can have only one curve type per curve.

You can also modify a value in the Val field of the parameters list.

A curve's Cycle setting determines the behavior before the first keyframe and after the last keyframe:

- *KeepValue (the default setting):* The value of the first and last keyframes is kept before the first keyframe and after the last keyframe.
- *KeepSlope:* Continues the tangent.
- *RepeatValue:* Repeats the curve between the first and last keyframes.
- *MirrorValue:* Reverses and repeats the curve.
- *OffsetValue:* Offsets and repeats the curve.

Note: The KeepSlope option cannot be used with curves that have expressions applied to them.

To learn how to use local variables and expressions to control your curves, see Tutorial 4, "Working With Expressions," in the *Shake 4 Tutorials*.

For more information on the cycle types, see "More About Splines" on page 316.

The Right-Mouse Menu

The lower portion of the right-click shortcut menu in the Curve Editor includes additional options.

- *Display Timecode:* Toggles between frame count and timecode count.
- *Sticky Time:* Lets you jump to the time of the keyframe you are modifying (so you can view the proper frame).

  Note: You can also press S to turn Sticky Time on and off.

- *Time Snap:* Toggles the locking of the keyframes to frame increments.
- *Display Selected Info:* Shows the numerical information for selected keyframes.

More About Splines

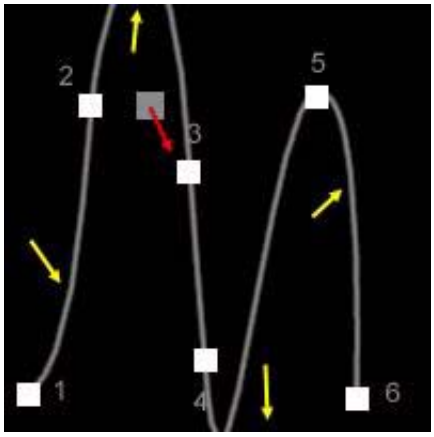
This section presents more technical information about the different spline types available in Shake.

Natural Splines

```
NSpline(cycle, value@key1,value@key2,...value@keyN)
NSplineV(time_value, cycle, value@key1,value@key2,...value@keyN)
```

The second-order continuity of natural splines ensures acceleration smoothly varies over time, so the motion is visually pleasing. The visual system is very sensitive to first- and second-order discontinuities, but not to higher-order discontinuities. But, in order to achieve the curvature continuity, the whole curve must be adjusted whenever a keyframe (CV) is moved. In the following example, when keyframe 3 is moved, the segments to keyframe 6 are changed. This is not good, even if the influence decreases very quickly as the number of intermediate keyframes increases.

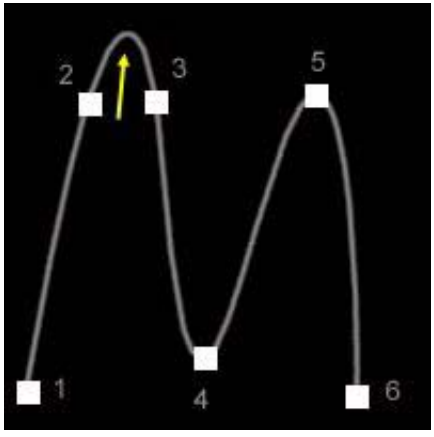
In addition, the keyframes completely define the curve, so there is no tangent control whatsoever.



Cardinal Splines

```
CSpline(cycle, value@key1,value@key2,...value@keyN)
CSplineV(time_value, cycle, value@key1,value@key2,...value@keyN)
```

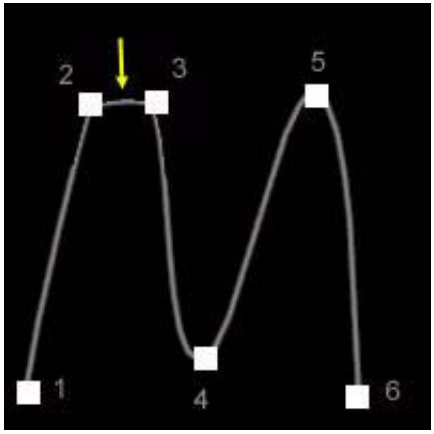
Cardinal splines trade off curvature continuity for local control. When a keyframe moves, only four segments are affected (two before and two after the keyframe). In addition, for any keyframe, tangents are automatically computed to be parallel to the segment joining the previous keyframe and the next keyframe. They are the programmer's best friend because they are so simple to evaluate—only two points are needed, which simplifies data management (no tangent or other complicated stuff).



Jeffress Splines

```
JSpline(cycle, value@key1,value@key2,...value@keyN)
JSplineV(time_value, cycle, value@key1,value@key2,...value@keyN)
```

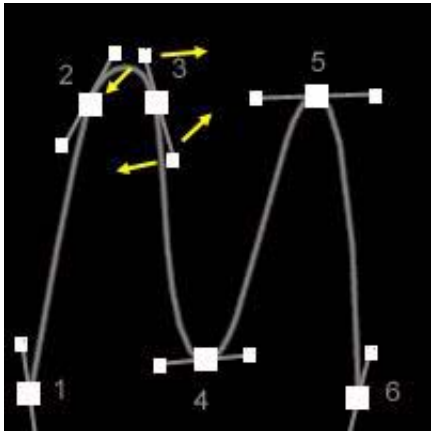
Jeffress splines are similar to CSplines, except they are guaranteed to never overshoot. If two keyframes have the same Y value, a flat segment connects them. This is very nice for animation, since you have a good idea of your limits.



Hermite Splines

```
Hermite(cycle,(value,tangent1,tangent2)@key1,...
  (value,tangent1,tangent2)@keyN)
HermiteV(time_value,cycle,(value,tangent1,tangent2)@key1,..
  (value,tangent1,tangent2)@keyN)
```

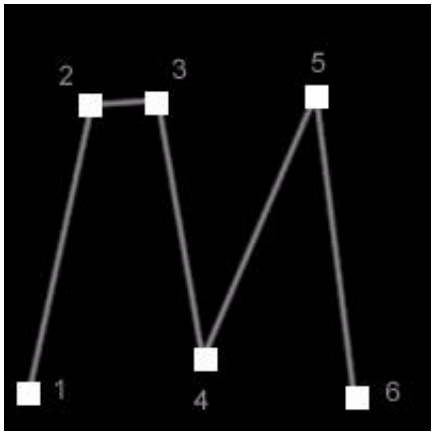
Hermite splines also give up on trying to produce curvature continuity, but they add tangent controls (so the animation is likely to look bad unless you eyeball the smoothness each time you move stuff around). You also have the ability to break the tangents (Control-click the handle end in the Curve Editor). It takes some effort to get right, but you can shape it the way you want.



Linear Splines

```
Linear(cycle,value@key1,value@key2,...value@keyN)
LinearV(time_value, cycle,value@key1,value@key2,...value@keyN)
```

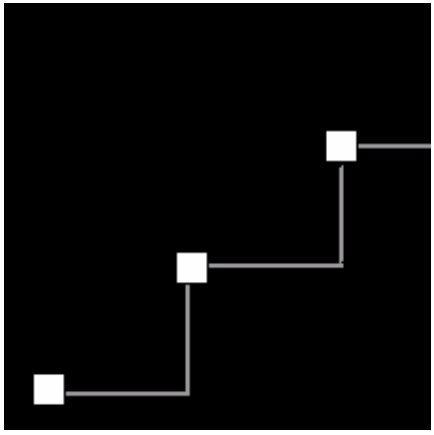
With Linear splines, there is not much mystery. No smoothness, but you know exactly what you get.



Step Splines

```
Step(cycle,value@key1,value@key2,...value@keyN)
StepV(time_value, cycle,value@key1,value@key2,...value@keyN)
```

Step splines create a stair-stepping spline that maintains its value until the next keyframe. This is great for toggling functions.



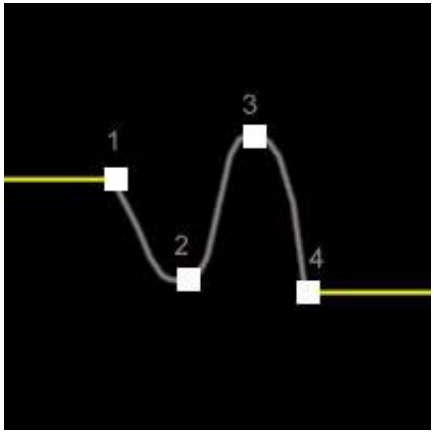
Cycle Types

You can change how the curve cycles its animation before and after the curve ends. The cycle is represented by a dotted line in the Curve Editor. The value is declared with the first parameter of a curve, for example, Linear (CycleType,value@frame1,...). Each cycle type has a numeric code:

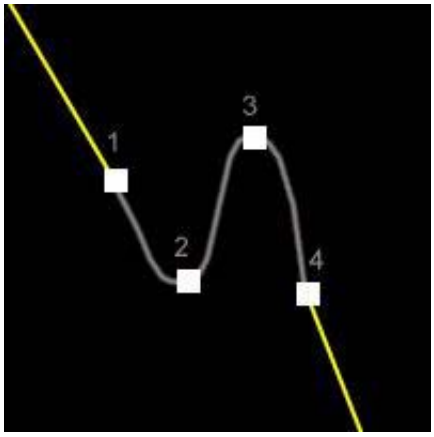
- 0 = Keep Value
- 1 = KeepSlope
- 2 = RepeatValue
- 3 = MirrorValue
- 4 = OffsetValue

KeepValue

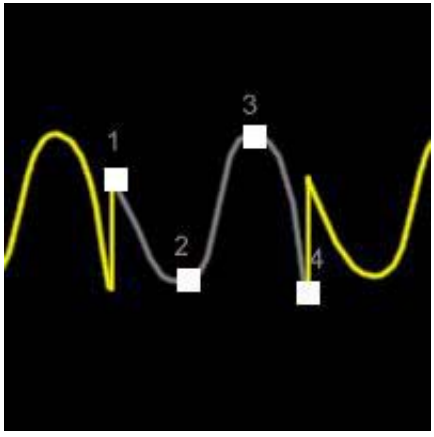
Keeps the value of the first and last keyframe when a frame is evaluated outside of the curve's time range.



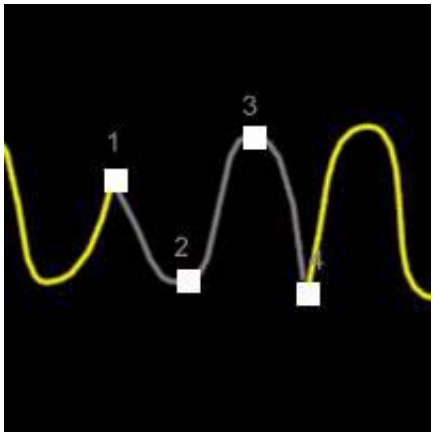
KeepSlope
Takes the slope of the curve at the last keyframe and shoots a line into infinity.



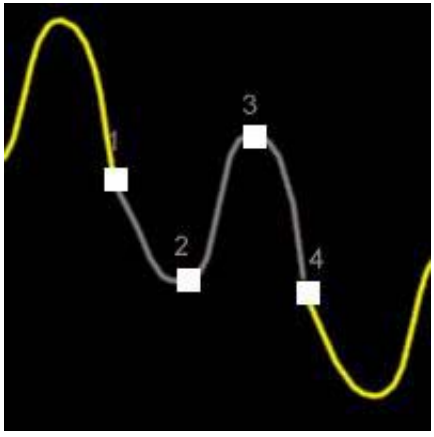
RepeatValue
Loops the animation curve. It works best when you set the first and last points at the same Y value, and maintain a similar slope to ensure a nice animation cycle.



MirrorValue
Also loops the animation, but inverts the animation each time the cycle repeats.



OffsetValue
Also loops the animation, but offsets the repeated curve so that the end keyframes match up.



The Flipbook, Monitor Previews, and Color Calibration

11

As you work with Shake, the Flipbook lets you preview your scripts in motion before actually rendering them to disk. The Monitor Preview control lets you send the Viewer output to an external monitor, allowing you to examine your image output on a different screen.

Cached Playback From the Viewer

You can cache frames using the Time Bar playback controls, to preview your work right in the Viewer.

To begin cached Viewer playback:

- Shift-click either the forward or back playback button in the Time Bar area.

The pointer changes into the cached-playback cursor, and Shake begins to render and cache all frames within the current frame range.

Once the frames have been cached, cached playback continues in a loop.

Launching the Flipbook

You can also render a temporary Flipbook to preview your work. Once the Flipbook has rendered into RAM, use the playback buttons (described below) to play back the sequence. The Flipbook is available on Mac OS X and Linux systems.

Note: On a Mac OS X system, you can create a disk-based QuickTime Flipbook. For more information, see "Creating a Disk-Based Flipbook" on page 326.

To launch the Flipbook from the Shake interface:

1. In the Globals tab, set the timeRange for the duration you want to preview. For example: 1-50, 1-50x2.

2 Load the node into the Viewer that represents the image you want to preview.

3 Do one of the following:

- Click the Flipbook button.

- Right-click in the Node View, then choose Render > Render Flipbook from the shortcut menu. Enter your settings in the Flipbook Render Parameters window and click Render. The images are rendered into RAM.

A Flipbook window appears, and the specified timeRange is rendered into RAM for playback.

You can also launch the Flipbook from the command line.

To launch the Flipbook from the command line:
Call up the files by relative or absolute paths. In the command line, indicate a time range and a frame placeholder—either # for padded numbers or @ for unpadded numbers. For multiple padding that is not four places, use the @ or # sign multiple times; for example, ##### = 00001, 00002, and so on. For example:

```
final_render.#.iff -t 1-56
final_render.#.iff -t 1-56x2
```

Flipbook Controls

When the Flipbook is finished rendering, there are a number of keyboard commands you can use to control playback within the Flipbook window.

Keyboard Command	Description
Period or >	Plays the sequence forward. The sequence automatically loops when it reaches the last frame.
Comma or <	Plays the sequence backward. The sequence automatically loops when it reaches the first frame.
Space bar	Stops rendering and/or playback.
/	Continues rendering after an interruption.
Left Arrow key, Right Arrow key	Steps through the sequence one frame at a time.
Shift-click and drag (in the Flipbook window)	Scrubs through the sequence.
Control->	Plays the sequence forward once, without looping.
Shift->	Plays the sequence forward, using ping-pong looping when reaching the last frame.
+ on numeric keypad	Increases the frame rate of playback.
- on numeric keypad	Decreases the frame rate of playback.

Keyboard Command	Description
T	Fixes the frame rate to real-time playback.
O	Displays information on the image (available on Linux systems only).

As the Flipbook plays, the frame rate is displayed in the title bar. If the Flipbook is playing back in real time, the title bar frame rate is followed by the word, "Locked."

If your computer cannot maintain the desired speed, Shake will drop frames, indicating the percentage of dropped frames in the title bar.

Viewing, Zooming, and Panning Controls

In the Flipbook, you also have access to the same viewing functions that are available in the Viewer.

Function	Key	Notes
View R, G, B, alpha, or lum channel	R, G, B, L, or A	
View RGB channels	C	
Get RGBA and x, y values of a pixel	Drag in the Flipbook window.	The values appear in the title bar.
Linux: overlay information	O	
Change color values between 0-1, 0-255, Hex	I	
Zoom in/out	+ or - key	
Pan image	Middle-click and drag	Some mouse button behavior may vary, depending on the manufacturer. If the middle mouse button does not pan, try right-clicking.
Re-center image	Home	
Close Window	Esc	

Memory Requirements

Real-time playback is a function of RAM, processor, image size, clip length, and graphics card. In Shake, images are loaded into memory and then played back. Current systems cannot achieve real-time playback with 2K-resolution images. With sufficient RAM and a good graphics card, files of up to 1K resolution should play back in real time.

Use the following formula to determine the amount of required memory:

```
width * height * channels * bytes per channel * images = bytes
```

For example, a single 1024 x 768 RGB 8-bit (1 byte) per channel image is:

```
1024 * 768 * 3 * 1 = 2359296 bytes
```

Or, it is approximately 2.4 MB per frame.

To convert from bytes to megabytes (MB), divide by 1024 two times (1024 equals the number of bytes per kilobyte). Thankfully, all operating systems come with calculators. For a rough approximation, drop the last 6 digits.

Note: An 8-bit image is 1 byte, a 10- or 16-bit image is 2 bytes, and a float image is 4 bytes.

Customizing the Flipbook

The following arguments have been added to the Flipbook executable as global plugs. This lets you specify an external Flipbook to come up as the default. Specify these plugs using a .h file in the *startup* directory. The global plugs and their default values are:

```
gui.externalFlipbookPath = "shkv"; // the flipbooks name -- this should
     include the full path
gui.flipbookStdInArg = "-"; // instructs the flipbook to take data from
     StdIn
gui.flipbookExtraArgs = ""; // allows you to enter any extra arguments the
     flipbook needs.
gui.flipbookZoomArg = "-z"; // sets the zoom of the flipbook
gui.flipbookTimeArg = "-t"; // the time range argument
gui.flipbookFPSArg = "-fps"; // the frames per second argument
```

If the specified external Flipbook doesn't support one of these arguments, setting its value to an empty string ("") prevents that value from being passed to it.

Creating a Disk-Based Flipbook

Available on Mac OS X systems only, the Render Disk Flipbook command launches a disk-based Flipbook into QuickTime. This approach has several advantages over normal Flipbooks. For example, the Disk Flipbook allows you to view very long clips and to attach audio (loaded with the Audio Panel in the main interface).

Note: Real-time playback performance varies depending on your system hardware.

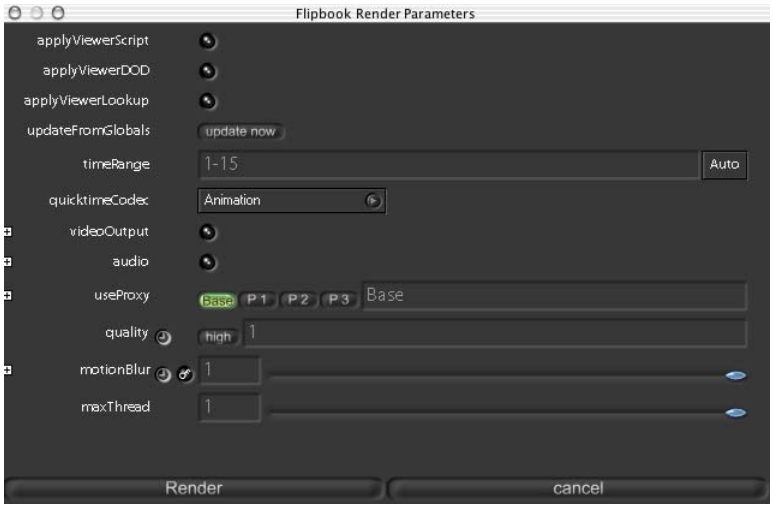
After viewing the Flipbook, you can write out the sequence as a QuickTime file and bypass the need to render the sequence again.

To render a Disk Flipbook:

Note: It is recommended to select a format for the Flipbook from the format pop-up menu in the Globals tab before rendering.

1 Choose Render > Render Disk Flipbook.

The Flipbook Render Parameters window appears.

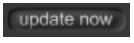


2 In the Flipbook Render Parameters window, set your Disk Flipbook parameters:

- *Viewer Scripts, DODs, and Lookups*: To apply an active Viewer Script, Viewer DOD, or Viewer Lookup to the Flipbook render, enable the desired parameter. For example, to render the Flipbook with the DOD currently set in the Viewer, enable applyViewerDOD.
- *updateFromGlobals* and *timeRange:* By default, updateFromGlobals is enabled.

When enabled, the time range and other Global settings (such as aspect ratio, proxy setting, motion blur, and so on) for the Flipbook render are constantly updated in the Flipbook Render Parameters from the Globals tab.

Note: To disable updateFromGlobals, toggle "updated" to "update now."



- *timeRange:* To override the updateFromGlobals parameter, enter a frame range in the time range field and press Return. The updateFromGlobals parameter is disabled.

  To automatically set the frame range based on the *FileIn*, click the Auto button.

- *quicktimeCodec*: Click the codecOptions button to open the Compression Settings dialog. Choose a codec from the Compression type pop-up menu. By default, the Animation codec is selected.
- *videoOutput*: To enable playback on a broadcast monitor, enable the videoOutput parameter.

Note: When using a broadcast monitor, ensure that the quicktimeCodec parameter is the same as the device parameter found in the videoOutput subtree.

The videoOutput subtree contains several options:

- *Device:* This pop-up menu contains a list of all the devices that can be output to. This pop-up menu is automatically set to the default, which uses your installed monitor card.
- *Conform:* This pop-up menu has three options that let you define how the image generated by your script is conformed to the frame size of the output device:
  - *Scale to Fit:* Outputs the current DOD to the broadcast monitor.
  - *Fit Maintaining Aspect Ratio:* Fits the DOD to the full screen of the broadcast monitor while maintaining the aspect ratio.
  - *Crop To Fit:* Crops the image to the DOD, and centers the image in the broadcast monitor.
- *aspectRatio:* Modifies the broadcastViewerAspectRatio in the monitorControls subtree of the Globals tab.

Underneath the videoOutput subtree, there are additional options:

- *audio*: To render the Flipbook with audio, enable the audio button. In the audio subtree, you can set the sample rate and audio bits.
- *useProxy*: You can use proxies in the disk-based Flipbook. For more information on proxies, see Chapter 4, "Using Proxies," on page 137.
- *quality*: Sets the quality for the Flipbook. By default, high (1) is enabled. Click the "high" button to toggle to "lo" quality.
- *motionBlur*: Enables and disables motion blur.
- *maxThread*: If you are working on a multiprocessor system, use the maxThread parameter to specify how many processors are devoted to the render.

3 Click Render.

The Shake QuickTime Viewer opens and the Pre-Rendering bar displays the render progress of the Flipbook. When the process is finished, the rendered QuickTime clip appears.

Note: The Shake QuickTime Viewer is a separate application—when launched, the viewer application icon appears in the Mac OS X Dock.

To view and save the Disk Flipbook:

1 In the Shake Preview (Shake QuickTime Viewer) window, click the Play button.

Note: You can also press the Space bar to start and stop playback.

2 To loop the playback, choose Movie > Loop.

Note: You can also choose Loop Back And Forth to "ping-pong" the playback.

If you're using a broadcast monitor, the Movie menu includes the following additional options:

- *Video Output:* Enables and disables the Flipbook display in the broadcast monitor.
- *Echo Port:* Enables and disables the Shake Preview window in the interface. When disabled, only the playback bar of the Shake Preview window is displayed.

Note: If audio is rendered with the Flipbook and Play Every Frame is enabled, you will likely lose audio in the playback.

3 To save the QuickTime render, choose File > Save Movie, and specify the name and location for the saved file.

4 To quit the QuickTime Viewer, do one of the following:

- Press Command-Q.
- Choose Shake QuickTime Viewer > Quit Shake QuickTime Viewer.
- Press Esc.
- Click the Quit button at the top of the Shake Preview window.

Disk-Based Flipbook Temporary Files

You can specify a location (other than the default) for the temporary disk-based Flipbooks. For example, if you have an Xserve RAID or other setup, you can store the temporary Disk Flipbooks on the array for real-time playback. The syntax for the default location for the temporary disk-based Flipbooks (in the *nreal.h* file) is:
sys.qtMediaPath = "/var/tmp/Shake/cache";

To change the location for the temporary files, create a .h file and put the .h file in your home directory in the */nreal/include/startup/* file. For example, create a .h file similar to the following:

```
sys.qtMediaPath = "/Volumes/Scene12/QTtemp";
```

Note: You must first create the folder to store the files—Shake does not create a folder based on the information in the .h file.

You do not need to comment out the default path in the *nreal.h* file. Any .h file in the startup folder overrides the *nreal.h* file.

Viewing on an External Monitor

When using the Mac OS X version of Shake, you can preview your work on a second display. This can either be a second computer monitor, or a broadcast video monitor connected to a compatible video output card (compatible video output cards support an extended desktop). For more information on compatible video cards, go to http://www.apple.com/shake/.

Note: The broadcast viewer option is not available on the Linux version of Shake.

To enable viewing via an external monitor:

1. Click the Globals tab.

   By default, the format parameter is set to Custom.

2. Choose the format of your footage from the format pop-up menu. For example, if you are working with NTSC D1 (4:3) non-drop frame footage, choose NTSC ND (D1 4:3) from the format pop-up menu.

3. In the Viewer shelf, click the Broadcast Monitor button.

The external video monitor mirrors the image displayed in the Viewer, which means that you can output the image from any selected node (the node displayed in the Viewer), as well as the Viewer scripts, VLUTs, and so on. In the Node View, the Viewers displaying the node are printed under the node (for example, 1A, 2A).

Note: If a broadcast Viewer is spawned prior to setting the correct format, the image may appear incorrect if the wrong aspect ratio is assigned. Go to the Globals tab and select the correct ratio from the format pop-up menu.

Customizing External Monitor Output

A group of parameters in the monitorControls subtree of the Globals tab let you adjust how the image sent to an external monitor will be displayed.

broadcastViewerAspectRatio

By default, this parameter is a link to `script.defaultViewerAspectRatio`, which mirrors the setting in the format subtree of the Globals tab. When first launched, Shake looks at the system's monitor card and outputs the proper aspect ratio based on the format you select in the Globals tab. For example, if you have a D1 card and you select NTSC D1 from the format parameter, Shake displays nonsquare pixels in the Viewer and sends square pixels to the video monitor.

Note: If you change the value of the broadcastViewerAspectRatio using the slider or the value field, the link to defaultViewerAspectRatio is removed. As with all Shake parameters, you can enter another expression in the broadcastViewerAspectRatio parameter to reset it.

broadcastHighQuality
When the broadcastHighQuality parameter is turned on, the image is fit to the size of the broadcast monitor in software mode (rather than hardware mode). The broadcastHighQuality parameter applies a scale node and a resize node, instead of using OpenGL. The broadcastHighQuality parameter is enabled by default.

broadcastGammaAdjust
Lets you adjust the gamma of your broadcast monitor to insure proper viewing (for example, if you are sending an SD NTSC signal to an HD monitor).

broadcastMonitorNumber
By default, Shake looks for the first available monitor with an SD or HD resolution to use as the external monitor. If you have more than one monitor card installed on your computer, this parameter lets you choose which monitor to use.

Note: The external display monitor doesn't have to be a broadcast display. If you have more than one computer display connected to your computer, the second one can be used as the external preview display.

Navigating the Broadcast Monitor

You can use the standard Viewer navigation keys, such as pan (hold down the middle mouse button and drag), zoom (press + or –), and Home in the broadcast Viewer.

To turn off the broadcast Viewer, do one of the following:

- Click the Broadcast Monitor button in the Viewer shelf.
- Position the pointer in the broadcast Viewer, right-click, then choose Delete Viewer from the shortcut menu.

Monitor Calibration With Truelight

Shake includes Truelight, a color management system that lets you simulate on your Shake preview monitor, as closely as possible, the image that will eventually be printed to film or displayed on a high definition monitor or projector. This simulation is based on calibration data acquired from a variety of film stocks, film recorders, monitors, digital projectors, and film projectors, and on calibration profiles that you can generate yourself.

There are three parts to the Truelight tools:

- *TLCalibrate*, in the Other tab, is a utility node that you can use to accurately calibrate your monitor's color characteristics. This node allows you to create a calibration profile by eyeballing adjustments to a series of ten images using the controls within this node. Once you've made your adjustments, you can save this profile for future use.

Note: This node also allows you to use calibration profiles generated by a Truelight Monitor probe.

- The calibration profiles generated using *TLCalibrate* can then be used by the Truelight VLUT control in the Viewer shelf.

The Truelight VLUT lets you previsualize the image in the Viewer as it will look after being output from a film recorder, or displayed by a high definition monitor or projector. You can use the Load Viewer Lookup command to make adjustments to the Truelight VLUT parameters in the Parameters tab, choosing which device profile you want to emulate.

- A second node in the Color tab, *Truelight*, performs the same function as the Truelight VLUT, except that it can be added to the node tree. The Truelight node has parameters that are identical to the Truelight VLUT that let you specify the device profile, current display profile, and color space for the preview. Additional controls let you fine-tune the preview.

Important: The Truelight functions are intended for previsualization only. They are not intended for use as color correction operations.

For more information on using the Truelight plugins, see the Truelight documentation, located in the Documentation folder on the Shake installation disk.

Rendering With the FileOut Node

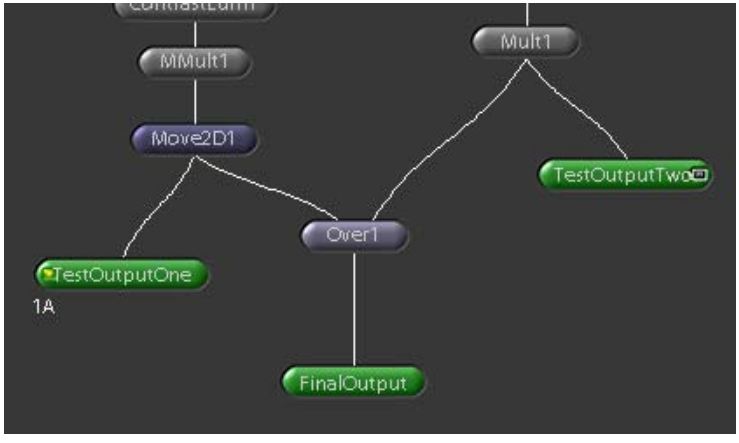
12

When you've finished your composite, you can set up one or more sections of your script to be rendered using FileOut nodes. This chapter covers how to render scripts from the Shake interface, from the command line, or by using Apple Qmaster.

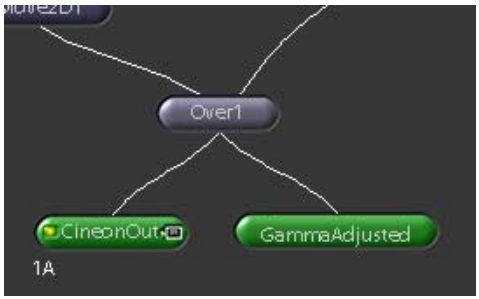
Attaching FileOut Nodes Prior to Rendering

After you've finished creating the necessary effect in Shake, you export your finished shot by attaching a *FileOut* node to the section of the node tree that you want to write to disk. You can attach an unlimited number of *FileOut* nodes anywhere along a node tree, which allows you to simultaneously output different sections of your composite at a variety of different resolutions, bit depths, or color spaces.

In the following example, three *FileOut* nodes have been added to different points in the node tree. The top two *FileOut* nodes will output each half of the composite individually, while the bottommost *FileOut* node will produce the combined results of the entire composite.



You can also branch multiple *FileOut* nodes from the same node, to output several versions of the same result. In the following example, two *FileOut* nodes simultaneously write both a 10-bit 2K log Cineon image and an 8-bit video resolution linear gamma-adjusted frame, in order to obtain a video preview of the composite before sending the filmout images to a film processing lab.



Note: You cannot export a QuickTime movie with a dynamically varying frame size by using a *FileOut* node. The resulting file will be unusable.

The FileOut Node

When you add a *FileOut* node to your node tree, the File Browser appears. You must choose a location for the rendered output to be written, and enter a name for the result.

For more information on using the File Browser, see "The File Browser" on page 38.

> ***Warning:*** You cannot name a *FileOut* node "audio."

File Paths

The *FileOut* node recognizes local, absolute, or URL paths.

Note: Local file paths in a script are local to the location of the script, not from where Shake is started.

- *If the image paths are local* (for example, *ImagesDirectory/image.#.iff*), images are written relative to where the script is saved.
- *If paths are global* (for example, *//MyMachine/MyBigHardDisk/ImagesDirectory/image.#.iff*), the directory images are written to have no relation to where the script is saved, and thus the script may be easily moved into different directories.

  If the script and the image are on different disks, you must specify the proper disk—local file paths do not work.
- *For a URL address*, place a *//* in front of the path. To write to another computer, write *//MyMachine/Drive/Directory/etc*.

File Names

If you write an image without a file extension (for example, *image_name* instead of *image_name.cin*), and you haven't explicitly set an output format, Shake writes the image to its native .iff format.

Otherwise, adding a file extension defines the type of file that will be written. For example, adding .tiff specifies a .tiff sequence, while adding .mov results in the creation of a QuickTime movie. If you need to change the type of file that's written later on, you can select a new file type from the fileFormat pop-up menu in the Parameters tab of that *FileOut* node.

If you're rendering an image sequence, you can also specify frame padding by adding special characters to the file name you enter:

- The # sign signifies a four-place padded number.
- The @ sign signifies an unpadded number.

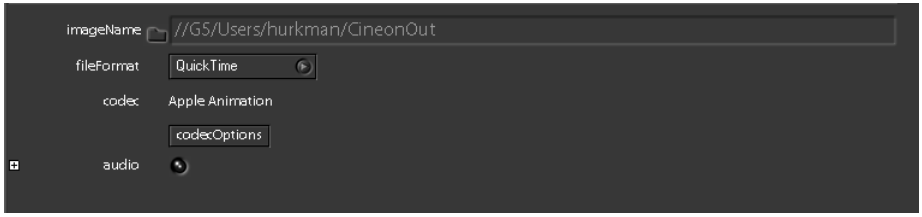
You can also use several @ signs to indicate padding to a different number. (For example, @@@ signifies 001.)

The following table lists some formatting examples.

Shake Format	Writes
image.#.iff	*image.0001.iff, image.0002.iff*
image.%04d.iff	*image.0001.iff, image.0002.iff*
image.@.iff	*image.1.iff, image.2.iff*
image.%d.iff	*image.1.iff, image.2.iff*
image.@@@.iff	*image.001.iff, image.002.iff*
image.%03d.iff	*image.001.iff, image.002.iff*

Parameters

The *FileOut* node displays the following controls in the Parameters tab:



imageName

The path and file name of the output image.

fileFormat
If no extension is given, the output format is .iff. To override this behavior, explicitly set the output format.

QuickTime Parameters

The following parameters become available from the codecOptions button if the fileFormat pop-up menu is set to QuickTime.

codec
A list of all available compression codecs installed on that computer.

compressionQuality
The quality of the compression algorithm. A value of 1 is maximum size, maximum quality. 0 is minimum size, minimum quality.

framesPerSecond
The frames per second for the playback of the QuickTime compression.

audio
Turn this parameter on to write audio out to the QuickTime file.

Rendering From the Command Line

To render a Shake script from the command line, each *FileOut* node is explicitly accompanied by a *-fo* (for -fileout). You can add multiple *FileOut* nodes along your command string to output different steps of the command.

For the batch system, you can use the *-fileout* option, or the abbreviation *-fo* to write your image. For example, to copy *my_image.cin* as a new image file in .iff format, use the following script:

```
shake my_image.cin -fo my_image.iff
```

The interface allows you to view frames anywhere along the node tree using multiple Viewers. In the script or the command-line mode, however, you may need to explicitly call intermediate nodes with either *-view* or *-monitor*. For example, to show two Viewers, one image rotated 45 degrees, and the second image rotated and flopped, use the following script:

```
shake my_image.rla -rotate 45 -view -flop
```

If you append a *.gz* to the end of the file name, Shake further compresses the file. Shake recognizes the file format and all of its channels when reading or writing one of these images:

```
shake uboat.iff -fo uboat.iff.gz
```

This further compresses *uboat.iff*, maintains it in .iff format, and retains the Z channel.

For more information on executing scripts, see Appendix B, "The Shake Command-Line Manual," on page 1015.

Using the Render Parameters Window

When a render is performed using the Render menu, the Render Parameters window opens. This window overrides the Global settings for your render. Note that these settings are not saved into the script; they only control how the Shake interface renders. To render to disk, you must attach an Image–*FileOut* node.

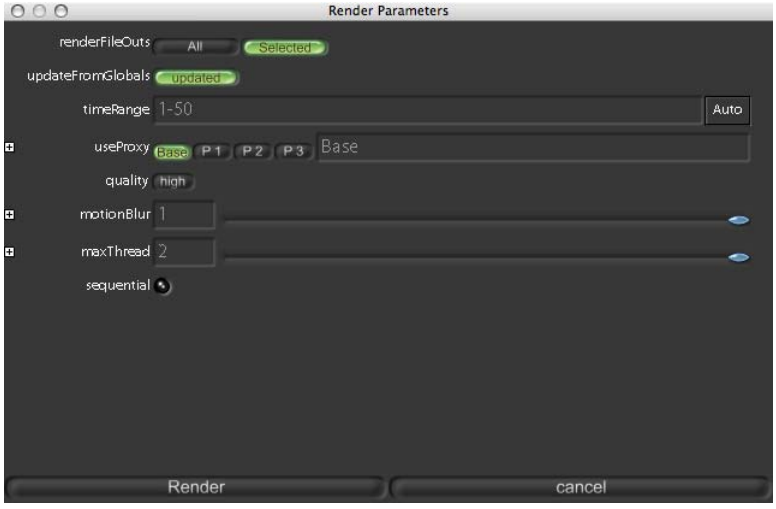
To render:

1 Attach Image–*FileOut* nodes to the nodes you want to render.

Note: To render only specific *FileOut* nodes, select the *FileOut* nodes in the Node View.

2 Choose Render > Render FileOut Nodes.

The Render Parameters window opens.



3 In the Render Parameters window, ensure that the timeRange (for example, 1-100) and other parameters are correct, then click Render.

Parameters in the Render Parameters Window

The Render Parameters window has the following parameters:

renderFileOuts
Indicates whether all *FileOut* nodes, or just the active nodes, are rendered.

updateFromGlobals
Indicates if your settings match the Globals tab settings (updated), or if you have modified the settings (update now), in which case the button allows you to update the settings from the Globals tab.

timeRange
Set a new time range using Shake's standard frame syntax; for example, 1-100 renders 1 to 100, 10-20x2 renders frames 10, 12, 14, up to 20, and so on.

useProxy
Sets your proxy settings.

quality
When this is set to lo (0), anti-aliasing is disabled. This results in poorer image quality, but improved render speed.

motionBlur
Motion Blur quality level. 0 is no blur, whereas 1 represents standard filtering. For more speed, use less than 1. This value multiplies the motionBlur parameter of every node that uses motion blur in your script.

shutterTiming
A subparameter of motionBlur. Shutter length 0 is no blur, whereas 1 represents a whole frame of blur. Note that standard camera blur is 180 degrees, or a value of .5. This value multiplies the shutterTiming parameter of every node that uses motion blur in your script.

shutterOffset
A subparameter of motionBlur. This is the offset from the current frame at which the blur is calculated. Default is 0; previous frames are less than 0. This value multiplies the shutterOffset parameter of every node that uses motion blur in your script.

maxThreads
Specifies how many processors are devoted to the render on a multiprocessor machine.

sequential
If you have multiple *FileOut* noded in your script, it may be more efficient to render the nodes sequentially. Turning sequential on causes each *FileOut* node to process every frame in the tree above it before allowing the next *FileOut* node to be rendered. When sequential is turned off, all *FileOut* nodes are rendered simultaneously. Sequential rendering is more efficient in cases where *FileOut* nodes share upstream nodes and trees. However, if there are too many processes running at the same time they will compete for CPU and memory resources, which may cause the overall processing time to increase, in which case turning sequential off may be more efficient.

The Render Menu

There are four options in the Render menu.

Menu Option	Description
Render Flipbook	Renders a Flipbook of the contents of the current Viewer. It first launches the Flipbook Parameters Window that allows you to override the Global parameters. To cancel the render, press Esc in the Flipbook window. See "Previewing Your Script Using the Flipbook" on page 90 for instructions on how to use the Flipbook.
Render Disk Flipbook	Available on Mac OS X systems only, the Render Disk Flipbook command launches a disk-based Flipbook into QuickTime. Disk Flipbooks have several advantages over normal Flipbooks. For example, the Disk Flipbook allows you to view very long clips and to attach audio (loaded with the audio tab in the main interface).
Render FileOut Nodes	Renders *FileOut* nodes in the Node View. Press F in the Node View to frame all active nodes. You have the option to render only the active *FileOut* nodes or all *FileOut* nodes.
Render Cache Nodes	Immediately caches sections of the node tree where *Cache* nodes have been inserted. This command lets you cache all Cache nodes in the Node View over a specific duration. For more information on using *Cache* nodes, see Chapter 13, "Image Caching," on page 343.
Render Proxies	Renders the proxy files for your *FileIn* nodes, leaving your *FileOut* nodes untouched. For more information on proxies, see Chapter 4, "Using Proxies," on page 137.

Support for Apple Qmaster

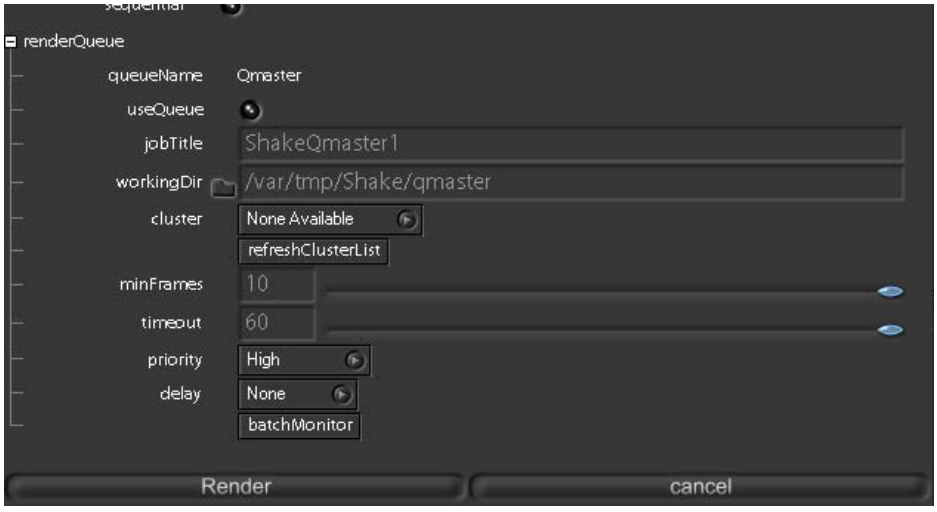
Apple Qmaster is a system of utilities that provides automated work distribution and processing projects created with Shake and other software packages on Mac OS X. Shake provides an optional interface, available as a subtree at the bottom of the Render Parameters window, which lets you submit jobs to the Apple Qmaster system. For more information on setting up and using Apple Qmaster, you are stongly encouraged to see the Apple Qmaster User Manual.

You can enable support for rendering using Apple Qmaster by adding the following global plug to a .h file in the *startup* directory:

```
sys.useRenderQueue = "Qmaster";
```

This setting causes additional options to appear in the Render Parameters window when you choose Render > FileOut Nodes. These options become visible when you open the renderQueue disclosure control.

Note: If Apple Qmaster isn't installed but the sys.useRenderQueue plug is declared, a message is sent to the console upon startup, and the following options do not appear.



renderQueue Options

The renderQueue parameter group contains the following options:

queueName

The name of the render queue software being used. If Apple Qmaster is installed, "Qmaster" appears here.

useQueue

When useQueue is turned on, the *FileOut* nodes specified by the renderFileOuts parameter are sent to the render queue when you click Render. By default, useQueue is turned off. Setting renderFileOuts to All sends all *FileOut* nodes to the render queue software. Setting renderFileOuts to Selected only sends the selected *FileOut* nodes to the render queue software.

jobTitle

Enter the name you want to use to keep track of this job here.

workingDir

The directory in which you want to store the temp script used by the render queue. The temp script is a temporary duplicate of your script that the computers in the specified cluster can access to perform the job.

Important: When you submit Shake jobs to a cluster, the working directory should reside on a shared volume that's accessible to all the computers in the cluster. This ensures that the working directory is accessible to the rest of the nodes in the cluster.

cluster
A pop-up menu that allows you to choose which cluster you want to use to perform the job. All clusters set up in your render queue software will appear here.

refreshClusterList
Shake checks for available clusters during startup. However, the available clusters may change depending on which computers are available on the network at any given time. Click this button to refresh the cluster pop-up menu with an up-to-date list of available clusters.

minFrames
Use this field to specify the minimum number of frames you want to be processed by each computer in the cluster.

timeout
The time, in seconds, a computer on a cluster can be idle before that part of the job is re-routed to another computer.

priority
A pop-up menu that allows you to choose the priority of the job. The options are High, Medium, and Low.

delay
A pop-up menu that allows you to delay when the render queue software starts the job you're submitting. The options are 15 minutes, 30 minutes, 1 hour, or 2 hours.

batchMonitor button
Click batchMonitor to launch the Apple Qmaster Batch Monitor application.

Image Caching

13

Shake has a powerful image caching system that keeps track of previously rendered frames in order to speed your workflow as you work within the interface. This system can be customized to optimize how you work.

About Caching in Shake

Shake's cache is a directory of pre-rendered images, with script information about each frame. When Shake displays the image data for a node tree at a given frame, it first checks the cache to see if that frame has been rendered before. If it has, the cached image is recalled to save time, as opposed to reprocessing the entire tree. Shake keeps track of how many times each cached frame has been viewed, eliminating the least viewed frames first when the cache runs out of room.

There are two ways you can control Shake's image caching—using the cacheMode parameter in the renderControls subtree of the Globals tab, or explicitly within the node tree using the Cache node.

Important: If you run two instances of Shake, only one instance is capable of reading from or writing to the cache. When you launch the second instance of Shake, you are given the option to either move the current cache to a temporary location, or disable caching.

Cache Parameters in the Globals Tab

The following parameters are found in the renderControls subtree of the Globals tab of the interface.

cacheMode

This parameter controls the caching behavior of all of the nodes in your script. Every node in your script is capable of being cached, in one way or another.

You can set the cacheMode to one of four states:

- *none:* Cache data is neither read nor written.
- *read-only:* Preexisting cache data is read, but no new cache data is generated.
- *regular:* The cache is both read from and written to, but normal caching does not occur when the global time is changed (as when moving the playhead).
- *aggressive:* The cache is both read from and written to, and normal caching occurs whenever the global time is changed (as when moving the playhead).

When setting the cacheMode, consider the following guidelines:

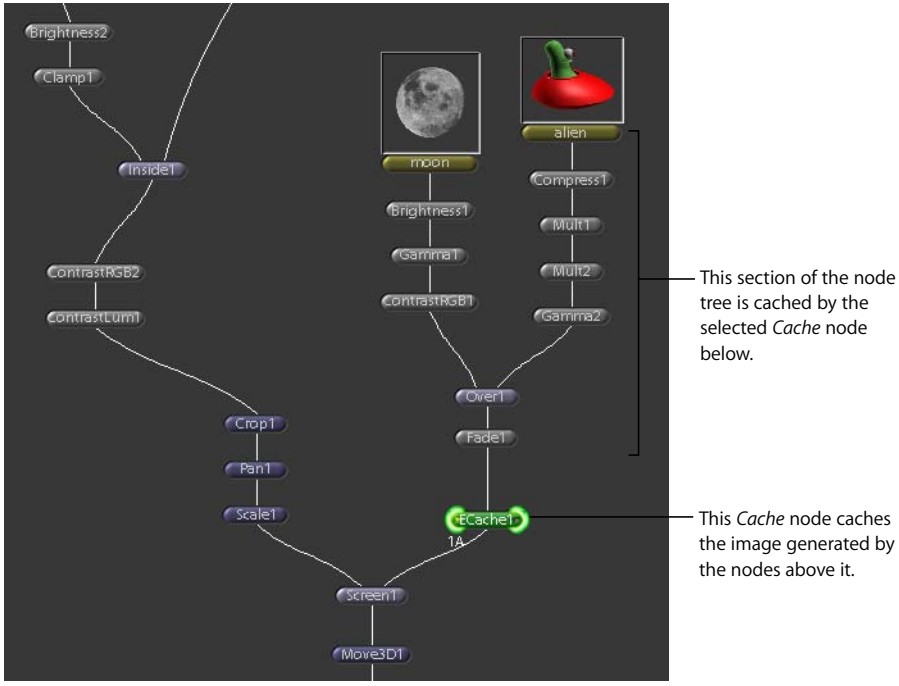
- In most circumstances, the regular cacheMode setting should be used.
- Consider setting the cacheMode to aggressive when you are constantly scrubbing back and forth between two or three frames (for example, when tweaking tracking or shape control points).
- You should only set cacheMode to none if you are using Shake on a system with extremely limited RAM and disk space. By setting the cacheMode to none, Shake is forced to re-compute each image that you select to view, which is the least efficient way to run.

Using the Cache Node

The *Cache* node lets you tell Shake to cache image data at specific points in the node tree. This gives you explicit control over which parts of the node tree require rendering while you work. For example, if there is a processor-intensive branch of your node tree that you're no longer working on, you can insert a *Cache* node in between the last node of that branch and the section of the node tree in which you're now working. Afterwards, the currently displayed frame is immediately cached. If you want to cache a range of frames in order to pre-render that branch of the node tree, you can use the Render Cache Nodes command. All cached image data is stored within the same cache, in memory or on disk.

Note: *Cache* nodes cache image data at the currently set proxy resolution.

From that point on, Shake references the cached image data within that node, instead of constantly reprocessing the nodes that precede it, unless a change is made to one of the preceding nodes.



Important: Using a *Cache* node crops the image to the current image size, eliminating data from the Infinite Workspace from that point in the node tree on.

When the Cache Becomes Full

Cache nodes that can't be cached appear red in the Node View.

This node is able to cache.

This node is unable to cache.

There are two possible situations when a *Cache* node won't be able to actually cache:

The input image size is larger than the maximum allowable cache file size.
You can easily tell if this is the case by opening the indicated *Cache* node into the Parameters tab, then checking to see if the value of the imageSize (an input image's Bit-depth * its Width * its Height) is larger than the value of the imageSizeLimit. If this is the case, you need to either increase the value assigned to the diskCache.cacheMaxFileSize global plug, or change the size of the incoming image.

The total cache memory limit has been exceeded.
The second possibility is that the amount of memory needed by all the *Cache* nodes in your script exceeds the memory assigned to the cache by the diskCache.cacheMemory global plug. In this case, no additional *Cache* nodes may be cached without increasing the diskCache.cacheMemory global plug.

For more information on the global plugs referenced above, see "Cache Parameters in the Globals Tab" on page 343.

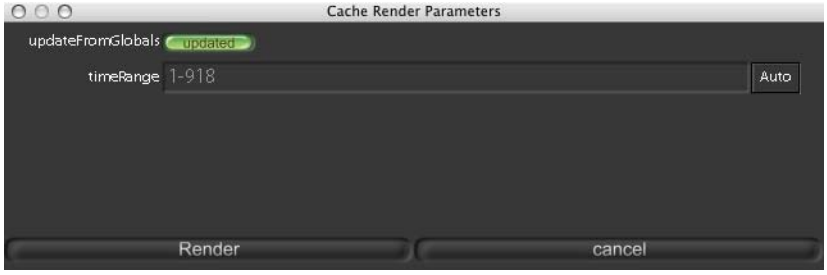
Caching and Updating Frames

The *Cache* node updates whenever the playhead moves, caching additional frames if necessary because of changes that have been made in the preceding nodes. If necessary, you can also render one or more *Cache* nodes and cache a range of frames in advance using the Render Cache Nodes command.

If you later make changes to one or more nodes in a section of the node tree that's been cached, the affected cached frames are discarded, and can be re-cached.

To use the *Cache* node:

1. Insert a *Cache* node after the last node of a section of the node tree that you want to cache.
2. Load the *Cache* node's parameters into the Parameters tab.
3. Select an option from the forceCache parameter. The disk+memory option is the default forceCache setting, and is almost always the preferred setting to use.
4. If you want to immediately cache that section of the node tree for a specified duration, choose Render > Render Cache Nodes.
5. The Cache Render Parameters window appears, which automatically updates the timeRange to the Global timeRange.



6. Click Render to render the *Cache* node. A Flipbook appears, allowing you to view the progress of the render, and play through the cached image sequence.

Parameters in the Cache Render Parameters Window

The Cache Render Parameters window has the following parameters:

renderCacheNodes

If you have multiple *Cache* nodes in the node tree, you can select one or more of these and render them simultaneously by setting renderCacheNodes to Selected in the Cache Render Parameters window. Or you can render all *Cache* nodes in the node tree by setting renderCacheNodes to All.

timeRange

If necessary, you can change the timeRange to cache a different frame duration.

useProxy

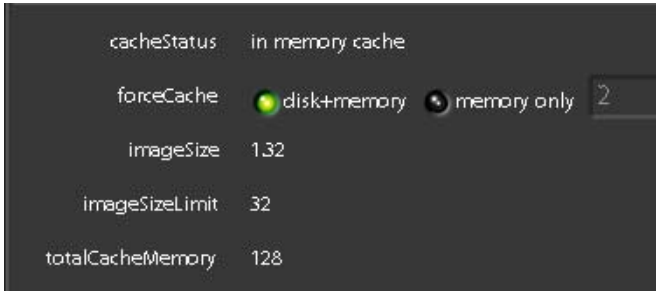
Images are cached using your script's current proxy setting. You can manually override the proxy setting in the Cache Render Parameters window, but those cache files won't actually be used by Shake until you change the script's proxy setting to match. This gives you the option to render multiple sets of cache images to match each proxy setting you plan on using.

sequential

Turning sequential on causes each *Cache* node to process the tree above it for each frame *before* allowing the next *Cache* node to process. When sequential is turned off, all *Cache* nodes are rendered simultaneously. This is more efficient in cases where *Cache* nodes share upstream nodes/trees. However, if there are too many processes running at the same time they will compete for CPU and memory resources, which may cause the overall processing time to increase.

Parameters in the Cache Node

The *Cache* node has the following parameters:



cacheStatus
This is a display-only parameter that shows whether the input image has been cached or not.

- *not cached:* Nothing has been written to the cache.
- *in disk cache:* Input image data has been moved to the disk cache. This is a result of the memory cache becoming full, or cache images having been saved after exiting a previous Shake session.
- *in memory cache:* The input image data has been written to the memory cache.
- *in transient memory cache:* The input image data has been written to the transient memory cache.

forceCache
This parameter lets you set how cached image data is stored when you update the cache with the Render Cache Node command. The selected forceCache behavior bypasses the Global cacheMode caching behavior. There are two options:

- *disk+memory:* The input image is written to the memory cache whenever the cache node is updated, and then transferred to the disk cache when the memory cache is full. All frames in the memory cache are moved to disk when Shake quits. In most cases this is the preferred behavior.
- *memory only:* Moves the input image into the memory cache every time the *Cache* node is updated, but never writes cache data to disk.

Internal Cache Parameter Display

The *Cache* node also displays the following parameters:

imageSize
The size (in megabytes) of the input image. The imageSize is determined by the following formula:

```
Bit-depth * Image Width * Image Height
```

imageSizeLimit
The maximum allowable size of the input image, in megabytes. This is set with the diskCache.cacheMaxFileSize global plug. The default value is 32 MB.

Note: The imageSizeLimit display lets you easily spot situations where a *Cache* node is not rendering because the imageSize is greater than the imageSizeLimit.

totalCacheMemory
The total RAM (in megabytes) available for caching to memory. This is set with the diskCache.cacheMemory global plug. The default value is 128 MB.

For more information on the global plugs referenced above, see "Customizing Image Caching Behavior" on page 352.

Commands to Clear the Cache

Ordinarily, cached frames in memory are written to disk and cleared as appropriate whenever you quit Shake. If necessary, you can also choose to clear parts of the cache manually while Shake is running.

Important: It's advisable to quit Shake normally whenever possible. If Shake quits unexpectedly, or you force-quit Shake yourself, the disk cache is invalidated. As a result, the remaining data on disk is unusable by Shake. Upon startup, you are prompted to either move the current cache to a temporary location, or disable caching.

Two commands in the File menu in Mac OS X are available to clear the cache:

Flush Cache

When you choose Flush Cache, all appropriate images are copied from the memory cache to the disk cache (depending on how the cacheMode parameter is set), but the memory cache *is not* cleared. This command is similar to what Shake does when you quit (the delay that occurs when you quit is Shake flushing the memory cache to disk).

Purge Memory Cache

Similar to the Flush Cache command, but the memory cache is cleared afterwards. This is useful if most of your RAM is filled with cache data, and you want to free it up to create and play a flipbook without needing to exit Shake in order to clear the memory cache.

Memory and the Cache in Detail

Shake incorporates two separate caches: an image cache and a processing cache.

The *image cache* is used to store output images that nodes produce. By storing the entire output image, the image cache can effectively "bake" portions of the processing tree, thereby saving re-computation time. Whether or not a node's image data is sent to the image cache primarily depends on the node's position in the node tree. When editing, the nodes directly above the node that is currently being viewed have the highest priority, which increases user interactivity. During playback, the node currently being viewed has the highest priority. The global plugs used to control the image cache are as follows:

- diskCache.cacheSize
- diskCache.cacheMemory
- diskCache.cacheMemoryLimit (Shake v3.5 and later)
- diskCache.cacheMaxFile
- diskCache.cacheMaxFileSize

The *processing cache* is mainly used to store image tiles (tiles are portions of the complete image) generated by nodes that need surrounding pixels to perform their image processing operations during a render. Example nodes include the *Blur*, *Transform, Warp, PinCushion*, and *Twirl* nodes. The processing cache also provides secondary functionality for caching rendering buffers (in particular for the *QuickPaint* node that utilizes a full-frame rendering buffer), color look-up tables, and transformation matricies. The global plugs used to control the processing cache are as follows:

- cache.cacheMemory
- cache.cacheMemoryLimit (Shake version 3.5 and later)

Limits to Shake RAM Usage

Shake is currently compiled as a 32-bit application, which can theoretically address up to 4 GB of virtual RAM (2^32). However, because of the RAM needs and constraints imposed by the operating system, and competition for RAM from other running applications, most 32-bit applications have a practical limit of approximately 2 GB of addressable RAM per process.

Because of this, even if you install 4 GB or more of RAM on your workstation, each Shake process can only take advantage of about 2 GB of that RAM. The good news is that, if you launch a Flipbook while running Shake on a system with 4 GB of RAM, the Flipbook (as a separate process) is able to take advantage of the additional 2 GB of RAM and is less likely to swap to disk, which is slower.

Mac OS X v10.3 and above (a 64-bit operating system) running on a PowerPC G5 computer configured with 8 GB of RAM partly addresses this issue. A 32-bit application running natively on a 64-bit OS is still limited to approximately 2 GB of addressable RAM. However, a Macintosh G5 computer configured with 8 GB of RAM running Panther can keep a larger number of applications in physical RAM without swapping out any one application's memory to disk. As a result, Shake is able to allocate larger contiguous segments of physical RAM, allowing large Shake scripts to be edited and rendered in less time.

The Image Cache

The main purpose of the image cache is to improve interactivity while you're working on a Shake script in the interface. Shake accomplishes this by attempting to output image data from nodes in the compositing tree at, and near, the portion of the compositing tree being edited or viewed. All nodes are capable of having their image data cached.

Similar to the processing cache, the image cache has both a fast RAM-based component and a slower disk-based component. However, the disk-based component of the image cache is only active during interface sessions (unlike the processing cache, which is active in all Shake run modes). In addition, the disk-based component of the image cache is limited in size and, when the disk cache fills up, Shake discards image data using an algorithm similar to that used by the processing cache.

Shake assigns cached image data one of three priorities. This determines which part of the RAM cache images are written to, and how long they're preserved. These priorities are:

- *Low (transient):* The low priority (transient) cache contains images which have only been accessed once. When the cache mode is set to regular, updating a parameter within a node moves the node directly upstream from it into the transient cache on the first update.

  When you use the Viewer playback controls to play through the frames in your script, Shake caches every frame that's been played into the high priority cache.

- *Medium (RAM only):* Shake keeps images in the medium priority cache as long as possible—they're only discarded when the RAM cache is completely full. Medium priority is assigned to images that have been accessed more than once without modification.
- *High (disk cache):* Shake also keeps images designated as high priority in the RAM cache as long as possible, transferring them to the disk cache when the RAM cache is full. All entries marked as high priority in the RAM cache are moved to disk when Shake quits.

  Cached images of medium priority are promoted to high priority when they have been accessed from the image cache four (counting the progression from low to medium) or more times without modification.

Preservation of the Disk Cache

The Shake disk cache is preserved on disk after you quit Shake. When you open a script with image data in the disk cache the next day, the cached image data from the previous day is recalled and used.

The Processing Cache

The processing cache has a fast RAM-based component and a slower disk-based component. If the memory limit of the RAM-based component is exceeded, Shake caches image tiles to disk using an algorithm that is based partially on when the image was last used. There is no memory limit imposed on the on-disk component of the processing cache.

The size of the RAM-based component of the processing cache is set in the *nreal.h* file using the cache.cacheMemory global plug. The default size is 128 MB and Shake internally sets a 256 MB upper limit on the size of this cache. This internal upper limit can be modified using the cache.cacheMemoryLimit plug. This is only recommended when working on systems with over 2 GB of RAM.

The following general guidelines apply when setting the cache.cacheMemory plug:

- For scripts with image resolutions of 2K or less, keeping the cache.cacheMemory at 128 MB should provide good performance.
- For scripts with larger image resolutions (over or equal to 4K) or scripts that include a large number of nodes that perform warps and distortions, consider increasing the size of cache.cacheMemory to 256 MB. However, you must first consider the amount of physical RAM installed in the workstation. If a workstation has 1 GB (or less) of RAM, it is not advisable to set cache.cacheMemory above 128 MB.
- On computers with 2 GB or more of RAM installed, you can raise cache.cacheMemory even higher, but you need to make sure that you also raise the cache.cacheMemoryLimit value.
- When running Shake on computers with limited RAM (for example, 512 MB) or when running both a Shake interface session and a background render on workstations with 1 GB (or less) of RAM, you may want to reduce cache.cacheMemory to 64 MB.

Both the RAM-based and the disk-based components of the processing cache are active in all of Shake's run modes—including interface sessions, background renders, and renders that are started from the command line.

Customizing Image Caching Behavior

This section provides details on how to customize Shake's caching behavior. The following parameters for caching in Shake must be manually declared via a .h file in the startup directory.

diskCache.cacheMemory

This global plug controls the size of the RAM-based component of the image cache. Larger values enable Shake to cache more of the node tree currently being edited/viewed. This enhances interactivity, especially when recursively viewing nodes both near the top and near the bottom of the node tree. Larger values also cache a greater number of images during playback—greatly increasing playback speed.

The default value for diskCache.cacheMemory is 128 MB, which enables Shake to cache approximately 86 images (8bit @ 720 x 486) into the RAM-based portion of the image cache.

The following guidelines apply when setting the diskCache.cacheMemory size:

- When editing large node trees in the interface, working at higher bit depths (that is, float), or repeatedly playing back an image sequence, you should consider increasing the diskCache.cacheMemory size to 256 MB, depending on the amount of physical RAM installed in the workstation.
- When running Shake on workstations with limited RAM (512 MB, for example) or when running both the Shake interface session and a background render on workstations with 1 GB (or less) of RAM, you should reduce diskCache.cacheMemory to 32 MB.

diskCache.cacheMemoryLimit
Internally, Shake sets an upper limit of 512 MB on the size of the RAM-based component of the image cache. This global plug allows users to override this limit. However, it is only recommended that you override (increase) this value when working with scripts that have large image resolutions (greater than 2K) and higher bit depths (float) on computers with greater than 2 GB of RAM. Increase at your own risk.

diskCache.cacheSize
This global plug controls the size of the on-disk component of the image cache. Larger values enable Shake to keep more "high priority" images around that have been pushed out of the RAM-based component of the image cache. Remember that this component of the image cache is inactive during background or command line renders.

- Now that workstations routinely have disk drives with hundreds of gigabytes of capacity, it's safe to increase the diskCache.cacheSize to 1 GB or more. This improves interactivity in large scripts, as well as scripts with high bit depth images.
- Time Bar playback also results in images being cached to disk. If you scrub or play through the Time Bar frequently, or play long sequences, increasing the diskCache.cacheSize to 1 GB or more allows multiple sequences to reside on disk.
- The only reason to reduce diskCache.cacheSize is if a computer has very limited disk space, or the very unlikely scenario that the workstation is using a remote disk mounted over a network as its cache drive—under these circumstances, the latency in retrieving cached images over the network may offset the computational advantages.

diskCache.cacheMaxFile
This global plug sets the maximum number of files that are stored in the disk-based component of the image cache. Larger values allow Shake to store more images, since each cached image is stored as a separate file. However, some file systems have limits on both the maximum number of open files allowed and the maximum size of those files, so you can use this parameter to reduce the number of files being used in the image cache if a particular system's file limit is being exceeded.

diskCache.cacheMaxFileSize
The global plug sets the maximum file size (in bytes) that can be stored in the disk-based component of the image cache. Greater values allow Shake to store larger images, since each cached image is stored in a separate file. However, some file systems have limits on both the maximum number of open files allowed and the maximum size of those files, so you can use this parameter to reduce the size of the files being used in the image cache if a system's file limit is being exceeded.

diskCache.cacheLocation
The directory to which disk cache files are written. By default, the cache is written to:

```
/var/tmp/Shake/cache
```

Note: Shake automatically creates cache directories if they do not already exist.

To free up disk space, you can remove this directory, but all caching information will be lost. This is not vital to a script, it simply forces Shake to completely rerender the compositing tree.

Customizing Shake

14

Shake's graphical interface can be highly customized. This chapter covers how to create preference files, and explains the different variables and settings that can be modified by the user.

Setting Preferences and Customizing Shake

This chapter explains how to customize the appearance of Shake, macro interactivity, and performance parameters. It also lists environment variables you can set to improve Shake's performance.

There are several other sections in the Shake documentation that cover similar information:

- For information on creating Viewer scripts, see "Viewer Lookups, Viewer Scripts, and the Viewer DOD" on page 61.
- For information on creating custom kernels for filters, see "*Convolve*" on page 865.
- For more information about creating macros, see Chapter 30, "Installing and Creating Macros," on page 905. For a tutorial on creating a macro, see Tutorial 8, "Working With Macros," in the *Shake 4 Tutorials*.
- For information on scripting, see Chapter 31, "Expressions and Scripting," on page 935.

Creating and Saving .h Preference Files

Unlike many applications that control user customizable settings with a preferences window, Shake provides access to a wide variety of functionality using a system of user-created preference files. This section discusses where to find the uneditable files that contain Shake's default functions and settings, and how to create and store your own separate preference files, to overwrite these settings and customize Shake's functionality.

Finding Shake's Default Settings

Shake uses two important files to set the original default settings. These files are named *nreal.h* and *nrui.h,* in the following directory:

On Mac OS X
<ShakeInstall>/Contents/Resources

On Linux
<ShakeDir>/include

> ***Warning:*** You should never modify either of these files. Doing so risks damaging your Shake installation, requiring you to reinstall Shake.

The first file, *nreal.h*, lists every function and default setting in Shake. Although this file should *never* be modified by the user, you can open it, and copy functions to use as a formatting reference for creating your own custom .h preference files. The commands found in the *nreal.h* file can be copied and saved in .h files within your own *startup* directory.

The second file, *nrui.h*, contains the data Shake uses to build the interface. It assigns menu names and contents, tabs, buttons, slider settings, and all of the default settings used by the interface in its default state. The commands in the *nrui.h* file should be copied and saved in .h files within your *ui* directory.

Note: To open these files for the Mac OS X version of Shake, Control-click or right-click the Shake icon, then choose Show Package Contents from the shortcut menu to view the Shake package contents (which include the *nreal.h* and *nrui.h* files).

Creating Your Own Preference Files

You can create your own preference files (.h files) to change Shake's default settings, add functionality, or change Shake's performance.

To add your own .h files to customize Shake:

1. Using your text editor of choice, create a new file, then enter the variables and settings you wish it to modify.

   These variables are covered in depth later in this chapter, and are referenced throughout the documentation.

2. Save the file as a plain text file.

   Preference files can be given any name, (*except "*nreal.h" or "nrui.h," which are reserved files used by Shake as the standard list of functions and settings), and they must have the .h extension to be recognized by Shake.

   Note: A fast way to disable a preference file is to remove its .h extension.

3. Place the new .h file within one of the directories discussed in the next section.

Possible Preference File Locations

Shake .h preference files can be saved in one of several locations (.h files in each of these locations are read by Shake in the order listed):

- In the Shake directory, *Contents/Resources/* and *Contents/Plugins/startup* (*<ShakeDir>/include/startup/ui* for Linux installs). These directories are scanned every time Shake is launched by any user that is using Shake called from this directory.
- In any directory listed in the $NR_INCLUDE_PATH list. Set the NR_INCLUDE_PATH environment variable to point to a list of directories. This is usually done when sharing a large project among many users.

  Note: For information on setting environment variables in Mac OS X, see "Environment Variables for Shake" on page 393.

  Add the following line to a *.cshrc* or *.tcshrc* file in your $HOME directory:

  setenv NR_INCLUDE_PATH " //MyMachine/proj/include:/Documents/shake_settings/include"

  Use the above for facility-wide or machine macros and settings that are read by all users. Because you can add multiple directories in the path list, you can have several locations of files.

- In the User directory. This is for settings for your own personal use. Shake automatically scans the *$HOME/nreal/include* subtree. A typical way to manage .h files is to create a directory named *ui* in the following location:

  $HOME/nreal/include/startup/ui

  The User directory can have the following subdirectories:

  - *Include/startup/ui*: For macros, machine settings, and interface settings.
  - *Settings*: For window layout settings.
  - *Icons*: For personal icons for the interface (also for the icons of macros).
  - *Fonts*: For personal fonts.
  - *Autosave*: For scripts saved automatically every 60 seconds (by default) by Shake.

Installing Custom Settings and Macros

Custom files that change default settings or add macros (see below) all have a .h file extension, and are located in:

$HOME/nreal/include/startup

For example:

/Users/my_account/nreal/include/startup/memory_settings.h

This is referred to as the *startup* directory. Files in this location are referred to as *startup* .h files.

Installing Custom Interface Settings

Settings that change the interface in some way (including macro interface files) are usually located in:

<somewhere>/include/startup/ui

These also have a .h file extension, for example:

/Users/my_account/nreal/include/startup/ui/slider_settings.h

This is referred to as the *ui* directory or sometimes *startup/ui* directory. Files inside it are referred to as *ui* .h files.

Files that change additional default settings or add extra controls are in the *templates* directory, which is always within a *ui* directory:

/Users/me/nreal/include/startup/ui/templates/defaultfilter.h

Installing Custom Icons

Just as you can create preference files, you can create your own icons. The description of the actual icons can be found in "Using the Alternative Icons" on page 375. Icons can be found in one of three locations:

- *<ShakeDir>/Contents/Resources/icons* (*<ShakeDir>/icons on non MacOS*): a directory, not to be confused with the important *icons.pak* file)
- *$HOME/nreal/icons*
- In any directory pointed to by $NR_ICON_PATH, set the same way $NR_INCLUDE_PATH is set

Preference File Load Order

Within a *startup* or *startup/ui* directory, files are loaded in no specific order. If it is important that a file is loaded before another file, this can be accomplished in a variety of ways.

To explicitly control preference file load order, do one of the following:

- Add an *include* statement at the beginning of the file. For example, if *macros.h* relies on *common.h* being loaded before, start *macros.h* with:

```
#include <common.h>
```

- Put all the files you want to load in a directory (for example *include/myprefs*) and create a .h file in *startup* that contains only *include* statements:

```
#include <myprefs/loadmefirst.h>
#include <myprefs/nowloadthis.h>
#include <myprefs/andthistoo.h>
```

Include files are never loaded twice, so it is okay if two .h files contain the same *#include <somefile.h>* statement.

Troubleshooting Preference Files

If your custom preference files do not appear to be working, check the following:

- Does the file have a .h extension?
- Is the file in a *startup* directory in one of the three possible locations (as described above)?
- If using a tcsh, and the file is in what you think is the NR_INCLUDE_PATH, is NR_INCLUDE_PATH actually set for that window? To test this, type the following in a tcsh window:

  echo $NR_INCLUDE_PATH

- Have you checked the text window from which you launched Shake? This is where syntax problems are shown.

Customizing Interface Controls in Shake

Two forward slashes (//) indicate that a line is commented out and inactive.

Color Settings for Various Interface Items

The following settings let you customize the colors of different interface controls.

Setting Tab Colors

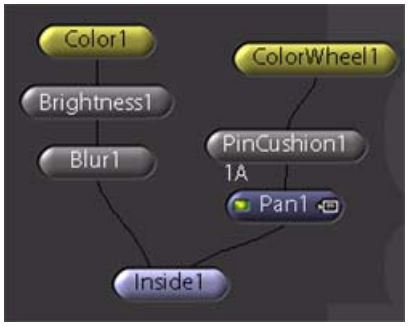
In the *ui* directory:


```
nuiPushToolBox("Image");
nuiSetObjectColor("ImageToolBox",
  tabRed, tabGreen, tabBlue,
  textRed, textGreen, textBlue);
nuiPopToolBox();
```

This is an excerpt from the *include/nrui.h* file. The Image tab is opened and assigned a color for both the tab and the text on the tab. Instead of numbers for the color values, variables are used here to indicate the parameter. Search for the variable names above or enter your own explicit values. Doing this does not automatically assign color to nodes within the tab.

Setting Colors for the Nodes in the Node View

In the *ui* directory:



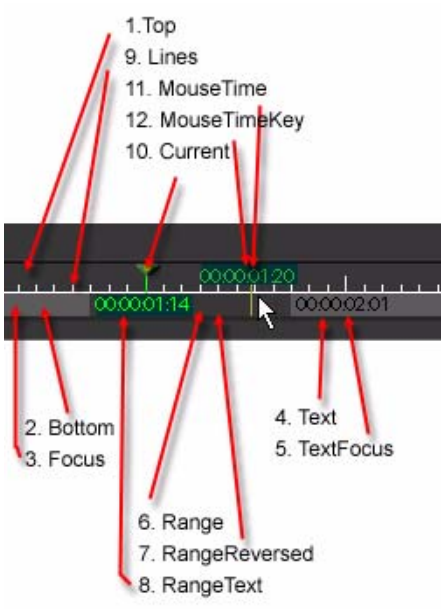

```
nuiSetMultipleObjectsColor(
  nodeRed, nodeGreen, nodeBlue,
  textRed, textGreen, textBlue,

  “DisplaceX”,
  “IDisplace”,
  “PinCushion”,
  “Randomize”,
  “Turbulate”,
  “Twirl”,
  “WarpX”
);
```

This command assigns colors to nodes in the Node View. The *nodeRed*, *green*, etc., and *textRed*, *green*, etc., are supposed to be float values. When coloring the nodes, keep in mind that the default artwork is a medium gray, so you can have numbers above 1 for the node color parameters to multiply it up.

Setting Colors for the Time Bar

In the *ui* directory:




```
gui.color.timeSliderTop = 0x373737FF;
gui.color.timeSliderBottom = 0x4B4B4BFF;
gui.color.timeSliderFocus = 0x5B5B5BFF;
gui.color.timeSliderText = 0x0A0A0AFF;
gui.color.timeSliderTextFocus = 0x000000FF;
gui.color.timeSliderRange = 0x373737FF;
gui.color.timeSliderRangeReversed = 0x505037FF;
gui.color.timeSliderRangeText = 0x0A0A0AFF;
gui.color.timeSliderLines = 0xFFFFFFFF;
gui.color.timeSliderCurrent = 0x00FF00FF;
gui.color.timeSliderMouseTime = 0xACAC33FF;
gui.color.timeSliderMouseTimeKey = 0xFCFC65FF;
```

These are just a few plugs to change the coloring of the text in all time-based windows, such as the Curve Editor, Time Bar, and so on. The numbers are, obviously, in hexadecimal format, just to make things more difficult. Ignore the 0x and the last FFs. Note you often have control over a basic color and its mouse-focused variation.

Setting Colors for Groups in the Node View

In the *ui* directory:

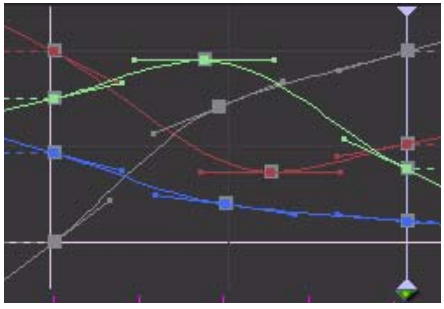

```
nuiSetObjectColor("Group", .75, .75, .75);
```

This sets the color of collapsed groups. If you set them to 1, the group takes on the color set in the Group's color setting:

```
nuiSetObjectColor("Group", 1., 1., 1.);
```

Setting Colors for the Curves in the Editor

In the *ui* directory:



```
gui.color.curveDef = 0x658a61;
gui.color.curveDefFoc = 0xcccc26;
gui.color.curveDefSel = 0xcccc26;
gui.color.curveDefFocSel = 0xffff26;
//Curves starting with 'r' or 'R'
gui.color.curveR = 0xa74044;
gui.color.curveRFoc = 0xff0000;
gui.color.curveRSel = 0xff0000;
gui.color.curveRFocSel = 0xff8888;
//Curves starting with 'g' or 'G'
gui.color.curveG = 0x8de48d;
```

```
gui.color.curveGFoc = 0x00ff00;
gui.color.curveGSel = 0x00ff00;
gui.color.curveGFocSel = 0xaaffaa;
//Curves starting with 'b' or 'B'
gui.color.curveB = 0x406bf7;
gui.color.curveBFoc = 0x1818ff;
gui.color.curveBSel = 0x1818ff;
gui.color.curveBFocSel = 0x8888ff;
//Curves starting with 'a' or 'A'
gui.color.curveA = 0x888888;
gui.color.curveAFoc = 0xbbbbbb;
gui.color.curveASel = 0xbbbbbb;
gui.color.curveAFocSel = 0xeeeeee;
```

There are really only four basic curve types, the normal curve (*Def*), the focused curve (*DefFoc*), the selected curve (*DefSel*), and the focused, selected curve (*DefFocSel*). You then also have additional controls over curves that start with the letters r, g, b, and a.

Setting Colors for Text

In the *ui* directory:


```
gui.fontColor = 0xFFFFFF;
gui.textField.fontColor = 0xFFFFFF;
gui.textField.tempKeyBackClr = 0xFFFFFF;

//the color of text on an active tab
gui.tabber.activeTint.red = .9;
gui.tabber.activeTint.green = .9;
gui.tabber.activeTint.blue = .87;

//the color of text on an inactive tab
gui.tabber.tint.red = .65;
gui.tabber.tint.green = .65;
gui.tabber.tint.blue = .63;
```

This colors the text in hexadecimal format. There are a series of expressions near the very end of the *nrui.h* file that allow you to put in normalized RGB values that are then fed into the hex number, but you can also determine your color using the Color Picker.

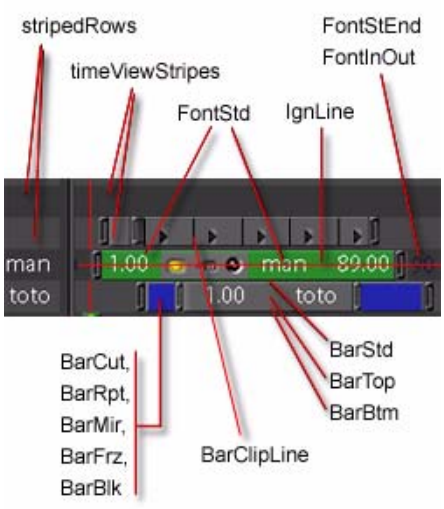
- *fontColor:* The color of the actual parameter name, messages, and also of macros without declared coloring.

- *textfield.fontColor:* The color of the values within the value field.
- *tempKeyBackClr:* A warning color for values entered when not in autosave mode for animated parameters. The value is not saved until the Autokey button is enabled.

Some text colors can also be interactively modified in the Globals tab. These are saved into *<UserDir>/nreal/settings* when you choose File > Save Interface Settings.

Setting Time View Colors

In the *startup* or *ui* directory:

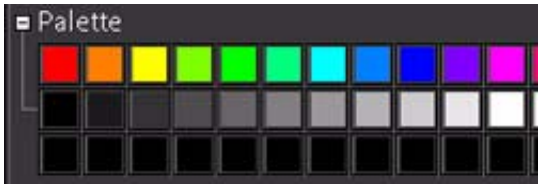



```
gui.color.timeViewBarStd = 0x737373;
gui.color.timeViewBarTop = 0x909090;
gui.color.timeViewBarBtm = 0x303030;
gui.color.timeViewBarCut = 0x101010;
gui.color.timeViewBarRpt = 0x5a5a5a;
gui.color.timeViewBarMir = 0x5a5a5a;
gui.color.timeViewBarFrz = 0x424242;
gui.color.timeViewBarBlk = 0x0;
gui.color.timeViewBarClpLine = 0x0;
gui.color.timeViewFontInOut = 0x111144;
gui.color.timeViewFontStEnd = 0x441111;
gui.color.timeViewFontStd = 0xFFFFFF;
gui.color.timeViewIgnLine = 0xFF0000;
gui.color.stripedRowColLight = 0x373737;
gui.color.stripedRowColDark = 0x474747;
gui.color.timeViewDarkStripe = 0x373737;
gui.color.timeViewLightStripe = 0x474747;
```

The *BarCut*, *BarRpt*, *BarMir*, *BarFrz*, and *BarBlk* refer to the repeat modes, so each one has a different color.

Creating a Custom Palette

In the *ui* directory:




```
nuiSetColor(1,1,0,0);
nuiSetColor(2,1,0.5,0);
nuiSetColor(3,1,1,0);
etc.
```

This assigns default colors to the palette icons, with the first number as the button number.

Custom Stipple Patterns in the Enhanced Node View

Different stipple patterns can be set in a .h preference file. Each stipple pattern is defined by a four-byte hex number that, when converted to binary, provides the pattern of the line drawn for each bit depth—each 1 corresponds to a dot, and each 0 corresponds to blank space.

For example, 0xFFFFFFFF is the hex equivalent of 1111111111, which creates a solid line. 0xF0F0F0 is the hex equivalent of 1111000011110000, which creates a dashed line.

The default settings are:

```
gui.nodeView.stipple8Bit = 0x33333333;
gui.nodeView.stipple16Bit = 0x0FFF0FFF;
gui.nodeView.stipple32Bit = 0xFFFFFFFF;
```

Adding Custom Media Formats to the Format Menu

You can create your own custom entries in the format pop-up menu in the Globals tab by adding the following declarations to a .h file in the *startup* directory.

Custom formats take the following form:

```
DefFormatType(
  “string”,
  width,
  height,
  aspectRatio,
  viewerAspectRatio,
```

```
  framesPerSecond,
  fieldRendering
);
DefFormatType(
  “Academy”, 1828, 1556, 1,1,24,“24 FPS”
);
```

Setting the Default Format Whenever Shake Is Launched

Add the following:

```
script.format = “FormatName”;
```

Setting Format Defaults

In the *startup* directory:

```
script.defaultWidth = 720;
script.defaultHeight = 486;
script.defaultAspect = 1;
script.defaultBytes = 1;
script.format = “Full”;
```

Using the *script.format* overrides the other settings—you either set the first four or the format settings, as shown above.

Assigning Default Width and Height to a Parameter in a Macro

In either *startup* or *ui* (typically inside of a macro’s parameter setting):

```
image MyGenerator(
  int width=GetDefaultWidth(),
  int height=GetDefaultHeight(),
  float aspectRatio=GetDefaultAspect(),
  int bytes = GetDefaultBytes()
)
```

These four commands check the default global settings and return the value at the time of node creation; they are not dynamically linked. Therefore, if you change the default parameters, the node’s values do not change.

Setting Maximum Viewer Resolution in the Interface

In the *ui* directory:

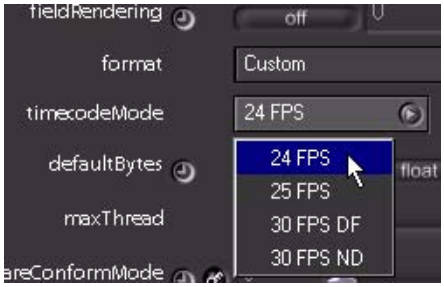
```
gui.viewer.maxWidth = 4096;
gui.viewer.maxHeight = 4096;
```

By default, Shake protects the user from test rendering an enormous image by limiting the resolution of the Viewer to 4K. If the user accidentally puts a *Zoom* set to 200 on the composite, it does not try to render an enormous file, but instead only renders the lower-left corner of the image cropped at 4K. To change this behavior, set a higher or lower pixel resolution. These assignments have no effect on files written to disk.

> ***Warning:*** Setting maxWidth and maxHeight to excessively high values may result in Shake unexpectedly quitting during certain functions.

Creating Custom Listings for the Format Pop-Up Menu

In the *startup* directory:




```
DefTimecodeMode(
  “Name”,
  fps,
  numFramesToDrop,
  numSecondsDropIntervals,
  numSecondsDropException
);

DefTimecodeMode(“24 FPS”, 24);
DefTimecodeMode(“30 FPS DF”, 30, 2, 60, 600);
```

These define the timecode modes for the timecodeMode pop-up menu in the Globals tab.

To set the default timecodeMode, use:

```
script.timecodeMode = “24 FPS”;
```

Default Timecode Modes and Displays

In the *startup* or *ui* directory:

```
script.framesPerSecond = 24;
script.timecodeMode = “24 FPS”;
gui.timecodeDisplay = 0;
```

Set one or the other. Setting the timecodeMode allows you to use drop-frame settings. See above to set the timecode modes. The third line is to display the frames in the Curve Editor and Time Bar as frames or timecode. 1 = timecode; 0 = frames. The other timecode modes are: “25 FPS,”“30 FPS DF,” and “30 FPS ND.”

Autosave Settings

The following declarations let you modify Shake’s autosave behavior.

Autosave Frequency

In the *startup* directory:

```
script.autoSaveDelay = 60;
```

This shows, in seconds, how often the autoSave script is performed. The script is saved automatically in your *User* directory as *autoSave1.shk*, *autoSave2.shk*, and so on, up to *autoSave5.shk*. It then recycles back to 1. If you lose a script because Shake unexpectedly quits, you can load in the autoSave version.

Four other autosave behaviors can be customized within a .h preference file.

Autosave Directory

In the *startup* directory:

```
script.autoSaveDirectory = "//myMachine/myAccount/myDirectory/";
```

Setting a directory with this declaration overrides the default behavior of placing autosave scripts in *~/nreal/autosave/*..

Autosave Prefix

In the *startup* directory:

```
script.autoSavePrefix = "MySweetScripts";
```

Defines text to be prepended to autosave script names. This is blank by default.

Autosave File Count

In the *startup* directory:

```
script.autoSaveNumSaves = 20;
```

Sets the total number of autosave scripts to be saved. Files are discarded on a first in, first out basis. The default autoSaveNumSaves value is 5.

Undo Level Number

In the *ui* directory:

```
gui.numUndoLevels= 100;
```

This determines how many steps of undo are available. Undo scripts are stored in the TEMP directory.

Amount of Processors to Assign to the Interface

In the *ui* directory:

```
sys.maxThread = nrcGetAvailableProcessors();
```

This sets the number of processors when using the interface. The nrcGetAvailableProcessors automatically calculates and assigns all of them. If you only want to use a limited number of processors, assign that number here.

You can assign the number of processors to be used when batch processing with the *-cpus* flag. The default is 1. For example:

```
shake -exec my_script.shk -cpus 2
```

Font Size for Menus and Pop-Up Menus

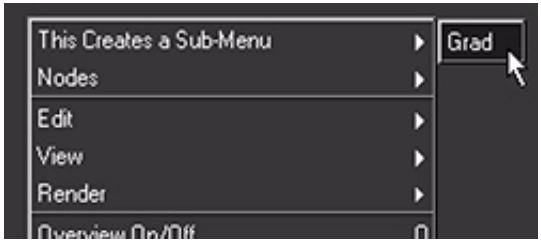
In the *startup* directory:


```
// It can take the following values:
//tiny, small, medium, big, std
gui.menu.fontSize= "std";
```

This should be in a *ui* .h file, but it must be set before the interface is built, so it goes in a *startup* file. The example is "tiny." The default is "std."

Adding Functions to the Right-Click Menu

In the *ui* directory:

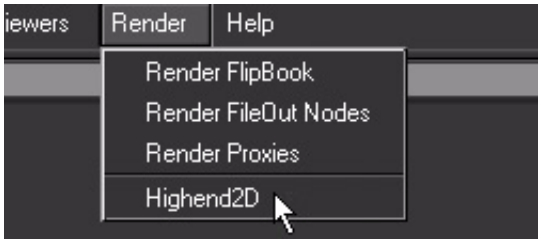



```
nuiPushMenu("NRiNodeViewPopup",1);
  nuiPushMenu(
     "This Creates a Sub-Menu",0
     );
     nuiMenuItem(
          "Grad",
          nuiToolBoxMenuCall({{Grad()}})
);
nuiPopMenu();
```

This is an example that creates a subtab called "This Creates a Sub-Menu" in the Node View, and attaches the *Grad* function to its list. This is just one example. Take a look at the *nrui.h* file, where all right-click menus are built. The first line declares under what menu it is built, so typically these commands are added directly into the *nrui.h* file.

Adding Functions Into a Menu

In the *ui* directory:




```
nuiOpenMenu(“Render”);
nuiMenuSeparator();
nuiMenuItem(
  “Highend2D”,
  LaunchBrowser(
     “http://www.highend2d.com”,1
  )
);
nuiPopMenu();
```

This creates an entry in the Render menu, split from the other entries by a separator.

Opening Scripts With Missing Macros

If you open a Shake script that contains one or more macros that you do not have on your system, you have the option to load the script using a substitute node, or to not load the script at all using the macroCheck parameter in the renderControls subtree of the Globals tab. To set the default macroCheck behavior to substitute a *MissingMacro* node, include the following in a .h file:

```
sys.useAltOnMissingFunc = 2
```

For information on the macroCheck parameter, see “renderControls” on page 96.

Linking an HTML Help Page to a Custom Node

To link a node’s HTML Help button to your own custom page, enter the following line into its ui.h file:

```
nuiRegisterNodeHelpURL ("MyCustomFunction", "http://www.apple.com/
shake/");
```

The Curve Editor and Time Bar

The following settings let you customize the Time Bar.

Setting the Time Bar Frame Range

In the *ui* directory:


```
gui.timeRangeMin = 1;
gui.timeRangeMax = 100;
```

That pretty much says it all, doesn't it?

Default Timecode Modes and Displays

In the *startup* or *ui* directory:

```
script.framesPerSecond = 24;
script.timecodeMode = "24 FPS";
gui.timecodeDisplay = 0;
```

Set one or the other. Setting the timecodeMode allows you to use drop-frame settings. See above to set the timecode modes. The third line specifies whether the frames in the Curve Editor and Time Bar are displayed as frames or timecode. 1 = timecode; 0 = frames.

Customizing File Path and Browser Controls

This section lists ways of customizing the File Browser.

Setting Default Browser Directories

In the *ui* directory:

```
gui.fileBrowser.lastScriptDir = "$MYPROJ/shakeScripts/" ;
gui.fileBrowser.lastExprDir = "//Server/shakeExpressions/" ;
gui.fileBrowser.lastTrackerDir = "$MYPROJ/tracks/" ;
gui.fileBrowser.lastAnyDir = "C:/Jojo/" ;
```

You can assign specific directories for the Browser to scan when you start the interface. You can assign different directories to different types of files, such as scripts, images, trackers, and expressions.

Important: There must be a slash at the end of the path.

Using the UNC File Name Convention

In the *startup* directory:


```
script.uncFileNames = 1;
```

Shake automatically assigns the UNC file name, that is, the entire file path name using the network address starting with *//MachineName//DriveName/path*. This ensures proper network rendering. However, you can turn this off by assigning the *uncFileNames* to 0, at which point local file paths are maintained. You can use local paths in either case, but they get converted when UNC is on.

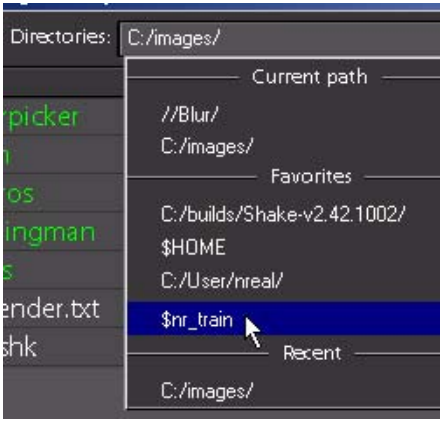
Using Relative Path Conventions

In the *startup* directory:

```
gui.fileBrowser.keepRelativePaths = 1;
```

Adding Personal Favorites to the Browser

In the *ui* directory:




```
nuiFileBrowserAddFavorite(
  “D:/icons/scr/”
);
nuiFileBrowserAddFavorite(
  “$nr_train/”
);
```

All directories assigned here appear in your Favorites area of the Directories pop-up menu in the Browser.

To also bookmark a directory in the Browser, click the Bookmark button and then choose File > Save Interface Settings. This saves a setting in your *$HOME/nreal/settings* directory.

Assigning a Browser Pop-Up Menu to a Parameter

In the *ui* directory:


```
nuxDefBrowseControl(
  “Macro.imageName”,
  kImageIn
);
nuxDefBrowseControl(
  “Macro.imageName”,
  kImageOut
);
nuxDefBrowseControl(
  “Macro.fileName”,
  kAnyIn
);
nuxDefBrowseControl(
  “Macro.lookupFile”,
  kExprIn
);
nuxDefBrowseControl(
  “Macro.scriptName”,
  kScriptIn
);
nuxDefBrowseControl(
  “Macro.renderPath”,
  kAnyOut
);
```

This assigns a folder button to a string so that you can relaunch the File Browser. The Browser remembers the last directories you used for any different type, so you can assign the type of file the Browser should look for as well with *kImageIn/Out*, and so on. For example, if you have a macro that browses an image to be read in, use *kImageIn*, so when you click that button, it jumps to the last directory from which you read in an image.

- *kImageIn*: Directory of the last image input directory.
- *kImageOut*: Directory of the last image output directory.

- *kAnyIn*: Directory of the last input directory of any type.
- *kAnyOut*: Directory of the last output directory of any type.
- *kScriptIn*: Directory of the last script input directory.
- *kScriptOut*: Directory of the last script output directory.
- *kExprIn*: Directory of the last expression input directory.
- *kExprOut*: Directory of the last expression output directory.

Automatic Launching of the Browser When Creating a Node

In the *ui* directory:

```
nuiToolBoxItem("ProxyFileIn",
  {{
  const char *filename = getFileInName();
  filename ? ProxyFileIn(filename,0,2) :
  (image) 0
  }}
);
```

In this example, the Browser is called for the parameter file name in the *ProxyFileIn* macro. The macro has three parameters: Filename and two numbers (0 and 2). The *getFileInName* function automatically launches the Browser when the user creates this node in the interface. You can use:

- *getFileInName()*
- *getFileOutName()*
- *getScriptInName()*
- *getScriptOutName()*

Automatic Browser File Filters

In the *ui* directory:


```
gui.fileBrowser.lastImageRegexp = "*.tif" ;
gui.fileBrowser.lastScriptRegexp = "*.shk" ;
gui.fileBrowser.lastExprRegexp = "*.txt" ;
gui.fileBrowser.lastTrackerRegexp = "*.txt";
gui.fileBrowser.lastAnyRegexp = "*";
```

You can assign specific filters for the Browser for different types of Browser activity. For example, if you only use Cineon files, you may want to use an assignment such as:

```
gui.fileBrowser.lastImageRegexp= "*.cin" ;
```

Tool Tabs

There are a number of ways you can customize the available Tool tabs.

Setting the Number of Node Columns in a Tool Tab

In the *<ShakeDir>/include/nrui.h* or a *startup* file:


```
gui.doBoxColumns = 8;
```

This sets the number of columns for the nodes in the Tool tab, which is sometimes called the "Do Box." Unlike the other *ui.* h files, this must go in *<ShakeDir>/include/nrui.h*, placed right before the call to start building the Image tab. To activate it, uncomment the bold line in the *nrui.h* file:

```
//These control the color of text on an inactive tab
gui.tabber.tint.red = .65;
gui.tabber.tint.green = .65;
gui.tabber.tint.blue = .63;
//gui.doBoxAltFxIcons = 1;
//gui.doBoxColumns = 5;

nuiPushMenu("Tools");
nuiPushToolBox("Image");
  nuiToolBoxItem("Average",   "const char *fileName = blah
  nuiToolBoxItem("Checker", Checker());
  nuiToolBoxItem("Color", Color());
  nuiToolBoxItem("ColorWheel", ColorWheel());
```

Using the Alternative Icons

In *startup* or the *<ShakeDir>/include/nrui.h* file:


```
gui.doBoxAltFxIcons = 1;
gui.doBoxColumns = 8;
```

This calls the alternative icon set, which concentrates more on the name of the function. The alternative icons are stored in *icons/fxAlt,* with the same name as the normal icons set, for example, *Image.Average.nri*, and so on. The dimensions for these icons are 130 x 26. Because they are wider, you typically limit the columns to five in a normal Shake environment. For a macro on generating these icons, see "MakeNodeIcon Macro" on page 998. You can activate the icons in two places, either a *startup* file, or by uncommenting the following two bold lines in the *nrui.h* file:

```
//These control the color of text on an inactive tab
gui.tabber.tint.red = .65;
gui.tabber.tint.green = .65;
gui.tabber.tint.blue = .63;
//gui.doBoxAltFxIcons = 1;
//gui.doBoxColumns = 5;

nuiPushMenu("Tools");
nuiPushToolBox("Image");
  nuiToolBoxItem("Average", " const char *fileName = blah blah blah
  nuiToolBoxItem("Checker", Checker());
...
```

Attaching a Function to a Button in the Tabs

In the *ui* directory:


```
nuiPushToolBox("Image");
nuiToolBoxItem("Flock", Flock(0,0,0));
nuiPopToolBox();
```

This places an icon that you have created into a tab that you assign. In this example, the icon is placed in the Image tab. If you use a custom name, such as *My_Macros*, it creates that tab. The second line first attaches the icon, and then assigns the function with its default arguments to that button. They do not have to be the same name, but both are case sensitive. The icon is either found in *<ShakeDirectory>/icons*, your *$HOME/nreal/icons*, or in any directory pointed to with $NR_ICON_PATH. The icons have the following characteristics:

- Although they can be any size, the standard resolution is 75 x 40 pixels.
- Do not use an alpha channel. Assign a *SetAlpha* (set to 0) or *Reorder* (set to rgbn) to remove the alpha channel.

- The file name is *TabName.Whatever.nri*. This example is therefore called *Image.Flock.nri*.
- The icon border is added automatically by Shake.

The section that says Flock(0,0,0) is the function of what that button actually does. You can assign any function to these—read in scripts, call multiple nodes, and so on. If the function does not have default values for its parameters, they must be provided here.

Attaching a Function to a Button Without an Icon

In the *ui* directory:


```
nuiPushToolBox("Image");
nuiToolBoxItem("@Flock", Flock(0,0,0));
nuiPopToolBox();
```

Note the @ sign before the icon name. This creates a button with whatever text you supply.

Creating Multiple Nodes With One Function

In the *ui* directory:


```
nuiToolBoxItem(
  "QuickShape",
  Blur(QuickShape())
);
```

You can create multiple nodes with one button click when you call up a function. For example, if you always attach a *Blur* node to a *QuickShape*, you can do this by embedding one function within another. The first argument for a function is usually the image input. By substituting the value (usually 0) with a different function, that function feeds into your first function. In the above example, *QuickShape* is fed into *Blur*.

Light Hardware Mode

In the *ui* directory:

```
sys.hardwareType = 1;
```

This command opens Shake without any borders on buttons, making the interactivity a little faster for graphically slower machines. A value of 0 is the normal mode; a value of 1 is the lightweight mode. Note the artwork isn't updated for the darker interface, so it looks a bit odd.

Customizing the Node View

The Node View defaults can be customized in many ways.

Setting Default Node View Zoom Level

In the *ui* directory:

```
gui.nodeView.defaultZoom=1;
```

These are plugs specifically for the Time View. They use the same hex syntax as the other color plugs.

Using Parameters Controls Within Macros

These are commands typically assigned to help lay out your macros by setting slider ranges, assigning buttons, and so on. These behaviors are typically assigned to specific parameters. They can be applied either globally (all occurrences of those parameters) or to a specific function. For example, if there is a trio of parameters named red, green, blue, Shake automatically assigns a Color control to it. However, for a parameter such as depth, you want to specify actions based on whether it is a bit depth-related function (and therefore assign a button choice of 8-, 16-, or float-bit depth) or a Z-depth related function (in which case you probably want some sort of slider). To assign a parameter to a specific function, preface the parameter name with the function name, such as *MyFunction.depth*.

All parameters, unless overridden by Shake's factory-installed rules, are assigned a slider with a range of 0 to 1.

Assigning a Color Control

In the *ui* directory:


```
nuiPushControlGroup("Color");
  nuiGroupControl("Func.red");
  nuiGroupControl("Func.green");
  nuiGroupControl("Func.blue");
nuiPopControlGroup();
nuiPushControlWidget(
  "Color",
  nuiConnectColorTriplet(
     kRGBToggle,
     kCurrentColor,
     1
  )
);
```

This assigns a button to three sliders so that you can scrub across an image and retrieve color information. You can select the current color, the average color, the minimum color, or the maximum color values. You can also assign a toggle switch to select the input node's color or the current node's color. For example, for pulling keys, you probably want to use the input node color since you are scrubbing (usually) blue pixels, rather than the keyed pixels. You can also choose to return different color spaces other than RGB. Assigning a Color control creates a subtree of those parameters.

Notice that you must first group the parameters into a subtree (the first five lines of the above example).

Color controls automatically appear if you name your trio *red*, *green*, *blue* or *red1*, *green1*, *blue1*, or *red2*, *green2*, *blue2*.

There are three parameters for the *nuiConnectColorPControl* function. The first one is the color space, which can be declared with either a string (for clarity) or an integer:

```
kRGBToggle   0
kHSVToggle   1
kHLSToggle   2
kCMYToggle   3
```

The second parameter describes the type of value to be scrubbed—the current, average, minimum, or maximum. Again, you can use either the word or the integer.

```
kCurrentColor   0
kAverageColor   1
kMinColor    2
kMaxColor   3
```

The last parameter is a toggle to declare whether you use the current node's pixel values or the input node's pixel values. You use either 0 or 1:

- 0 = current node
- 1 = input node

Use of the current node may possibly cause a feedback loop. Typically, for color corrections, you use current node; for keyers, the input node.

Therefore, the above example creates a subtree called *Color* for the function called *MyFunction*. The scrubber returns RGB values, of which only the current value is returned. When the Color control is called, the Use Source Buffer is turned on.

Assigning the Old Color Control

In the *ui* directory:


```
nuiPushControlGroup("Func.Color");
  nuiGroupControl("Func.red");
  nuiGroupControl("Func.green");
  nuiGroupControl("Func.blue");
```

```
nuiPopControlGroup();
nuiPushControlWidget(
  “MyFunction.Color”,
  nuiConnectColorPControl(
     kRGBToggle,
     kCurrentColor,
     1
  )
);
```

This is an older version of the Color control without the cool extra controls.

Changing Default Values

In the *<ShakeDir>/include/nrui.h* file:

```
nuiPushToolBox(“Color”);
  nuiToolBoxItem(“Add”, Add(0,0,0,0,0,0));
  nuiToolBoxItem(“AdjustHSV”, AdjustHSV(0));
  nuiToolBoxItem(“Brightness”,    Brightness(0,1));
  nuiToolBoxItem(“Clamp”, Clamp(0));
  nuiToolBoxItem(“ColorCorrect”, ColorCorrect(0));
...
```

In the *include/nreal.h* file, most functions have their default values declared, but not all of them. To override the default values when you call the function, modify the line that loads the function in the interface. If every parameter in a function has a default value in a function, you can call the function with something like:

```
nuiToolBoxItem(“Clamp”, Clamp(0));
```

Normally, *Clamp* has about 8 values. Here, the 0 represents in the first argument, the input image. 0 is used to indicate that no images are expected to be inserted, so you can attach it to the active node. However, you can add additional parameters. For example, the *Brightness* line above it has (0,1), 0 for the image input (no input) and 1 for the brightness value. Change the 1 to a different value to override it. You only need to supply the parameters up to the one you want. For example, the following is the call for the *Text* function:

```
nuiToolBoxItem(“Text”, Text());
```

To override the default font for the *Text* function, you have to supply the width, height, bytes, text, and finally the font. The rest you can ignore afterward:

```
nuiToolBoxItem(“Text”, Text(
  GetDefaultWidth(),
  GetDefaultHeight(),
  GetDefaultBytes(),
  “Yadda Yadda”,
  “Courier”
  )
);
```

Grouping Parameters in a Subtree

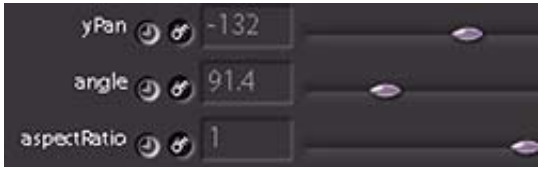
In the *ui* directory:


```
nuiPushControlGroup("Func.timeRange");
  nuiGroupControl("Func.inPoint");
  nuiGroupControl("Func.outPoint");
  nuiGroupControl("Func.timeShift");
  nuiGroupControl("Func.inMode");
  nuiGroupControl("Func.outMode");
nuiPopControlGroup();
```

This groups parameters into a subtree that can be opened and closed by the user. This example, although it says "Func," is for the *FileIn* node.

Setting Slider Ranges

In the *ui* directory:




```
nuiDefSlider(
  "Funct.yPan", 0, height
);
nuiDefSlider(
  "Funct.angle", -360, 360
);
nuiDefSlider(
  "Funct.aspect", 0, 2, .2, .01
);
```

Even though the sliders are in relatively the same position, there are different numbers in the value fields. You can set slider ranges and precision with this function. The first line assigns a slider range just for the yPan parameter of the *Move2D* function. Note the use of the height variable so the range adjusts according to the input image. The second line assigns a range for the angle parameter in any node. The third line also has optional precision parameters, which are granularity and notch spacing.

- *granularity* represents the truncation point. Shake cuts off all values smaller than your truncation value, that is, if your granularity is .01, a value of .2344 becomes .23. Granularity is a "hard" snap—if granularity is set to 0.001, you cannot get anything but multiples of 0.001 when you slide. Also, granularity cannot be anything but a multiple of 10 (+ or -).
- *notch spacing* represents at what value interval the magnets appear. A value of .1 means the magnets are at .1, .2, .3, and so on. The default value is .1. Notch spacing is a "soft" snap—the slider tends to stick longer to multiples of notch spacing, but does not prevent the selection of other values. Think of it as a tiny notch in a flat line where a ball rolls: The ball tends to get stuck in the notch, but if you keep pushing, it eventually gets out.

Adding Pop-Up Menus

In the *ui* directory:


```
nuxDefMultiChoice("Defocus.shape",
"fast gaussian|fast box|circle"
);
```

This pop-up menu, from the *Defocus* function, allows you to use a pop-up menu for strings. Note this only supplies strings and not numbers, so you have to do some tricky math inside the macro itself. For more information, see Chapter 31, "Expressions and Scripting," on page 935.

Creating Radio Buttons

In the *ui* directory:


```
nuxDefRadioBtnControl(
  “Text.xAlign”,
  1, 1, 0,
  “1|ux/radio/radio_left”,
  “2|ux/radio/radio_center”,
  “3|ux/radio/radio_right”
);
```

This example is for the *Text* node. This code creates a series of radio buttons that are mutually exclusive. The naming convention assumes that you have four icons for each name, with the icon names *name.on.nri*, *name.on.focus.nri*, *name.off.nri*, and *name.off.focus.nri*. If no icons exist, you can choose to not use icons, which then gives a label with an on/off radio button instead. The code has these parameters:

```
nuxDefRadioBtnControl(
  const char *name,
  int useIcon,
  int useLabel,
  int animatable,
  curve string state0, ....
);
```

You can place as many icons as you want. The height of Shake’s standard parameters icons is 19 pixels, though this can change. The output parameter for the *Primatte* and *Keylight* nodes is a good example.

You can make your own radio buttons with the *RadioButton* function. This function is discussed in “RadioButton Macro” on page 999.

Creating Push-Button Toggles

In the *ui* directory:

```
nuxDefExprToggle("Func.parameter",
  "repl.nri|repl.focus.nri",
  "interp.nri|interp.focus.nri",
  "blur.nri|blur.focus.nri"
);
```

This assigns a series of buttons to toggle through integers starting at 0. The first line is assigned a value of 0, the second line assigned a value of 1, the third assigned a value of 2, and so on. You can place as many toggles as you want. There are two buttons for each assignment, the normal button, and a second button for when the pointer passes over the button to signify that you can press it. Note the standard buttons are all in the subdirectory *ux*, but this is not a requirement. Shake includes a series of precreated icons that are packed into the *icons.pak* file and are inaccessible to the user, but are understood by this code. Your custom icons can be any size, but the default height is 19 pixels. You cannot have an alpha channel attached to an icon. Use *SetAlpha* (set to 0) or *Reorder* (set to rgbn) to remove the alpha channel. They can be placed in *<ShakeDirectory>/icons*, the *$HOME/nreal/icons*, or $NR_ICON_PATH.

Creating On/Off Buttons

In the *ui* directory:


```
nuxDefExprToggle("Func.param");
```

This is similar to the push-button toggles, but you only have two values, on and off—off a value of 0, and on a value of 1. The icon assignment is automatic.

Making a Parameter Non-Animateable

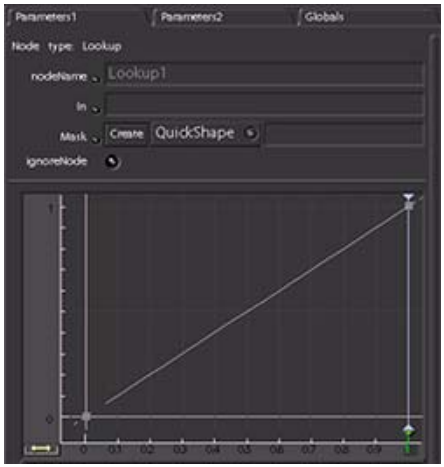
In the *ui* directory:

```
nriDefNoKeyPControl("DilateErode.soften");
```

This designates that no Autokey buttons appear.

Placing a Curve Editor Into a Parameters Tab

In the *ui* directory:




```
nuiPushControlGroup("colorExpr");
  nuiGroupControl("Lookup.rExpr");
  nuiGroupControl("Lookup.gExpr");
  nuiGroupControl("Lookup.bExpr");
  nuiGroupControl("Lookup.aExpr");
nuiPopControlGroup();

//Makes all curves invisible by default
registerCurveFunc("colorExpr");
//This makes all curves visible by default
registerCurveFuncVisible("colorExpr");

gui.colorControl.curveEditorDirection = 0;
//When it is 1, the layout is vertical
//When this equals 0, the layout is
//horizontal
```

This code loads a Curve Editor embedded inside the Parameters tab. The first six lines of code simply group parameters together. The last line then attaches the parameters to the Curve Editor embedded in the Parameters tab.

Viewer Controls

This section discusses Viewer settings and onscreen controls.

Setting Maximum Viewer Resolution in the Interface

In the *ui* directory:

```
gui.viewer.maxWidth = 4096;
gui.viewer.maxHeight = 4096;
```

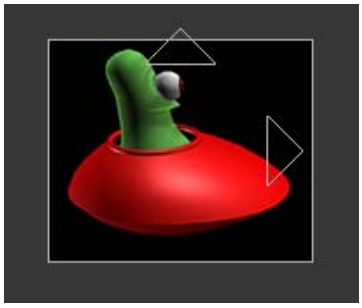
By default, Shake protects the user from test rendering an enormous image by limiting the resolution of the Viewer to 4K. If the user accidentally puts a *Zoom* set to 200 on the composite, it does not try to render an enormous file, but instead only renders the lower-left corner of the image cropped at 4K. To change this behavior, set a higher or lower pixel resolution. These assignments have no effect on files written to disk.

Onscreen Controls

Onscreen controls are automatically built according to how you name your parameters in your macro, with one exception—to make a cross-hair control. The following is the list of parameters it takes to make certain controls. For the illustrations, the controls are attached to their appropriate functions. For example, the pan controls are attached to a *Pan* function and scaling to a *Scale* function. Simply naming the parameters does not necessarily give you the functionality you want.

Panning Controls

In the *startup* macro file:

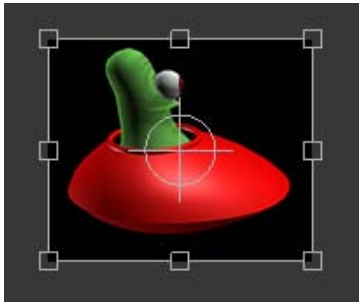


```
float xPan = 0,
float yPan = 0
```

This gives you the lattice to pan around. You can grab anywhere on the cross bars.

Scaling Controls

In the *startup* macro file:

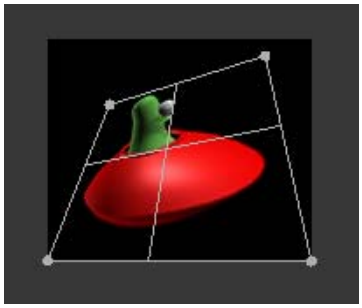


```
float xScale = 1,
float yScale = 1,
float xCenter = width/2,
float yCenter = height/2
```

This gives you the border and center controls to change the scale center. You can grab a corner to scale X and Y, or an edge to scale X or Y.

CornerPin Controls

In the *startup* macro file:



```
float x0 = 0,
float y0 = 0,
float x1 = width,
float y1 = 0,
float x2 = width,
float y2 = height,
float x3 = 0,
float y3 = height
```

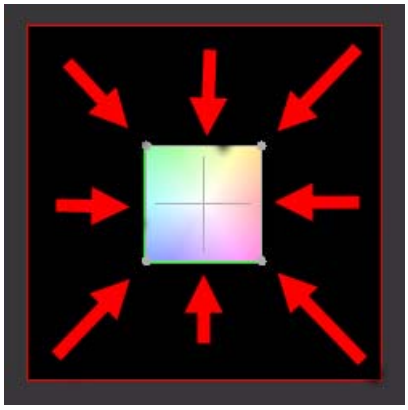
In the ui file:

```
nuiPushControlGroup("Func.Corner Controls");
  nuiGroupControl("Func.x0");
  nuiGroupControl("Func.y0");
  nuiGroupControl("Func.x1");
  nuiGroupControl("Func.y1");
  nuiGroupControl("Func.x2");
  nuiGroupControl("Func.y2");
  nuiGroupControl("Func.x3");
  nuiGroupControl("Func.y3");
nuiPopControlGroup();
```

Grab any corner or the crosshairs in the middle to adjust the position of your image. The grouping code for the ui file is included, so you do not have to look at all eight parameters in your list.

Box Controls

In the *startup* macro file:



```
int left = width/3,
int right = width*.66,
int bottom = height/3,
int top = height*.66
```

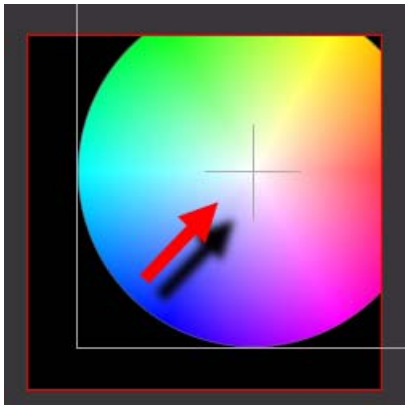
In the ui file:

```
nuiPushControlGroup("MyFunction.Box Controls");
  nuiGroupControl("MyFunction.left");
  nuiGroupControl("MyFunction.right");
  nuiGroupControl("MyFunction.bottom");
  nuiGroupControl("MyFunction.top");
nuiPopControlGroup();
```

This creates a movable box. You can grab corners or edges, or the inside crosshairs. This example is applied to a *SetDOD* function. The Layer–*Constraint* and Transform–*Crop* nodes also use these controls. In this example, integers are used for values that assume you are cutting off pixels, but you can also use float values.

Offset Controls

In the *startup* macro file:

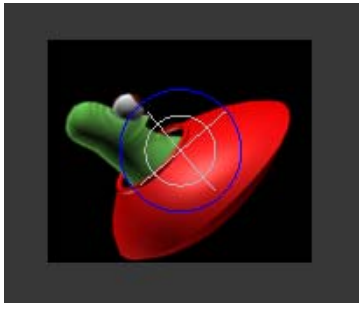


```
float xOffset = 0,
float yOffset = 0
```

This is similar to the *Pan* controls, but with center crosshairs. This control is available in the Other–*DropShadow* node.

Rotate Controls

In the *startup* macro file:

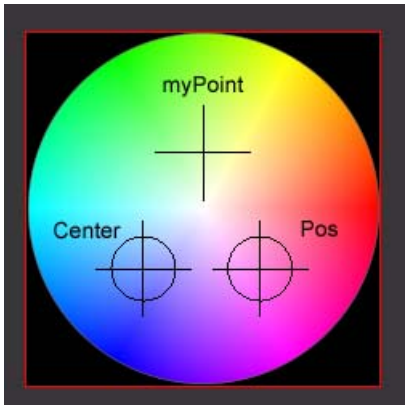


```
float angle = 0,
float xCenter = width/2,
float yCenter = height/2,
```

This gives you a rotation dial and a center control. This example is plugged into a *Rotate* function.

Point Controls

In the *startup* macro file:




```
float xCenter = width*.33,
float yCenter = height*.33,
float xPos = width*.66,
float yPos = height*.33,
float myPointX = width/2,
float myPointY = height*.66
```

In the UI file:

```
nuiAddPointOsc(“Func.myPoint”);
```

These three sets of parameters create a crosshairs control. *Center* and *Pos* are default names—the *Center* pair is also associated with the angle and the scale parameters. However, the last point is completely arbitrary, as long as it ends in an uppercase X and Y. In the *ui* file, you must also declare that these are an XY pair.

Radius Controls

In the *startup* macro file:

```
float radius = width/6,
float falloffRadius = width/6,
float xCenter = width/2,
float yCenter = height/2
```

This is basically for *RGrad*, but maybe you can do some more with it.

Template Preference Files

You can add additional parameters and default settings by adding files into the *startup/ui/templates* directory. Each time Shake is launched, it adds these extra parameters. For example, if you always want the Proxy Filter to be "box" instead of "default," and you always want a slider in the Globals tab called *switch1*, create a .h file under the templates directory with:

```
SetProxyFilter("box");
curve int switch1 = 0;
```

Basically, take a saved script and strip out the lines you want to set as defaults, and save it as a .h file into *templates*.

Changing the Default QuickTime Configuration

You can change the default QuickTime configuration that appears when you set the fileFormat parameter of a *FileOut* node to QuickTime. The default QuickTime configuration is also the configuration that Shake falls back on when a script is opened with a *FileOut* node that's set to use a QuickTime codec that's not available on that computer.

The default settings in Shake are limited to the ones you find in the standard QuickTime Compression Settings dialog.

To change the default QuickTime configuration:

1. Create a script with a *FileOut* node.
2. Select the *FileOut* node, then choose QuickTime from the fileFormat pop-up menu in the Parameters tab.
3. Click codecOptions, then set the codec options in the Compression Settings dialog.
4. Save the script.
5. Open the script in a text editor and find the definition of the *FileOut* node you created. After the name of the codec, you'll see a 0, then a long, seemingly nonsensical string of text in quotes. Copy the long nonsensical string, but not the quotes.
6. Create a .h file in your *include/startup* directory. Type:

```
sys.QTDefaultSettings = " x ";
```

where x is the long string you copied, in quotes with a semicolon at the very end of the line. Here is an example:

```
sys.QTDefaultSettings =
    "100W@u3000WDcsuHA#M000E@4J3In8q5CBRZ0VY2LKKPgATB3A9KSC7gXaC30q8v
    W16OG5Koq10h2A5HIvi00ieKA9WT6a1rS9hH8dqIEiBqOHT0SJEZ8HHc8qtf4rlxS
    AP9WYwcYJHfMMCKpWXYn2W893LCsk080000@00000000000;";
```

Note: The above declaration sets the Uncompressed 10-bit 4:2:2 codec as the default.

The default settings in Shake are limited to the ones you find in the standard QuickTime Compression Settings dialog.

Note: If the default codec you've specified is not available, a message is sent to the console, and the default codec reverts to Animation.

Environment Variables for Shake

This section discusses two ways to set environment variables, and the variables recognized by Shake. At the end of the section, some examples of aliases are provided.

> ***Warning:*** Incorrectly setting environment variables can lead to problems running Shake and with your operating system. If you are not comfortable with changing these types of settings, consult your system administrator for guidance.

Environment variables are strings of information, such as a specific hard drive, file name, or file path, set through a shell (for example, in Terminal on a Mac OS X system) that is associated with a symbolic name (that you determine). This information is stored in a hidden file. Each time you launch Shake, the operating system and the Shake application look at the hidden file to set the environment variables. In other words, defining environment variables is the equivalent of setting user-defined system preferences.

As a simple example, you can set an environment variable that specifies a folder that Shake scans (on launch) for additional fonts used by the *Text* or *AddText* node.

To set environment variables on a Mac OS X system, create and edit a ".plist," or property list, file. Using the .plist sets variables for Shake whether it is launched from the Terminal or from the Shake icon.

Using the above example of a font folder, to instruct Shake to read the */System/Library/Fonts* folder, set the following environment variable in your .plist file:

```
<key>NR_FONT_PATH</key>
<string>/System/Library/Fonts</string>
```

Another way to define environment variables is to use the *setenv* command in a *.tcshrc* (enhanced C shell resource) file. Each time the Terminal is launched, the *.tcshrc* file is read. The environment variables defined by the *.tcshrc* file are only read by Shake when launched from the Terminal.

Using the above example of a font folder, to instruct Shake to read the */System/Library/Fonts* folder, set the following environment variable in your *.tcshrc* file:

```
setenv NR_FONT_PATH /System/Library/Fonts
```

Note: The *.tcshrc* file can be used on all Shake platforms (Mac OS X and Linux).

A common use for a user's personal *.plist* or *.tcshrc* file is to define commonly used aliases for commands. As a simple example, you can set an environment variable to launch Shake from the Terminal.

An alias in the command line is not the same as an alias on the Macintosh operating system. In the OS, an alias merely points to another file. In the command line, you create an alias to assign your own name to a command.

Note: If you do not have environment variables set on your Mac OS X system, you can still launch Shake from the Terminal by typing the complete path to Shake:

```
/Applications/Shake4/shake.app/Contents/MacOS/shake
```

To set the Shake path in the Terminal, do the following:

1. Launch Terminal.
2. In the Finder, navigate to the Shake application (usually located in the *Shake4* folder in the *Applications* folder).
3. Drag the Shake icon to the Terminal.

   The Shake path is automatically entered in the Terminal.
4. Set environment variables for Shake. For example, you can specify the location of important files that your Shake script needs when opened.
5. Specify the Shake directory.

Creating the .plist Environment File

Each time you log in, the system searches for an environment file, named *environment.plist*. This file sets an environment for all processes (launched by the logged-in user). In the Terminal, you create a directory called *.MacOSX* that contains the environment file. You also create the environment file (using a text editor), and move the file into the *.MacOSX* directory.

To set environment variables in Shake on Mac OS X using the .plist file:

1. Log in using your personal login.
2. Launch Terminal.

   By default, you should be in your Home ($HOME) directory. Your Home directory is your own directory in the Users folder. For example, if John Smith logs in and launches the Terminal, the following message is displayed in the Terminal:

   ```
   john-smiths-Computer:~] john%
   ```
3. In the Terminal, type the following command to create a directory in your Home directory called *.MacOSX*:

   ```
   mkdir $HOME/.MacOSX
   ```
4. Press Return.

   An invisible directory (indicated by the " . " in front of the directory name) is created in your Home directory.
5. To ensure the *.MacOSX* directory was created, type:

   ```
   ls -als
   ```
6. Press Return.

   All files, including the new invisible *.MacOSX* directory, are listed.
7. Next, launch TextEdit (or another text editor) to create a file to set your variables.

   Note: If you have installed and are familiar with the Apple Developer tools, you can use the Property List Editor application to create or edit variables. The Property List Editor application is located in *Developer/Applications*.

8 In the text document, create the following file (if you're reading this in the PDF version of the user manual, you can copy the following and paste it into the text document) and edit the information.

Note: The following is an example file for instructional purposes only.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist SYSTEM "file://localhost/System/Library/DTDs
     PropertyList.dtd">
<plist version="0.9">
<dict>
<key>MyProject</key>
<string>/Documents/MyBigFilm</string>
<key>NR_INCLUDE_PATH</key>
<string>/Documents/MyBigFilm/macros</string>
<key>NR_ICON_PATH</key>
<string>/Documents/MyBigFilm/icons</string>
</dict>
</plist>
```

This sets the variable *MyProject* to */Documents/MyBigFilm*. This tells Shake that all files associated with *MyProject* (that could be a script, directory, and so on) are located in */Documents/MyBigFilm*. As a result, if you type *MyProject* in the browser, it returns */Documents/MyBigFilm*, and can then be set as a favorite. This file also sets the *NR_INCLUDE_PATH* (points to the directory or directories that you want Shake to scan for macros and personal machine or user interface settings), and *NR_ICON_PATH* (points to a directory where you can save your own icons for Shake functions).

9 In TextEdit, choose Format > Make Plain Text.

The document is converted to a *.txt* file.

10 Choose File > Save.

11 In the "Save as" field of the Untitled.txt window, enter:

```
environment.plist
```

Be sure to remove the *.txt* file extension.

12 Save the file to your Home directory (in TextEdit, choose Home from the Where pop-up menu), then click Save.

13 In the Save Plain Text window, click "Don't append."

The file is saved in your Home directory with the extension *.plist*.

14 Quit TextEdit.

In order for Shake and your system to access the environment variables, the *environment.plist* file must be saved to the *.MacOSX* directory (created in step 3).

15 To save the *environment.plist* file to your *.MacOSX* directory, move the file (using the Terminal) from your Home directory to the *.MacOSX* directory. In the Terminal, do the following:

a To ensure you are still in your Home directory, type the "present working directory" command:

```
pwd
```

Using the example from step 2, this should return:

```
/Users/john
```

b Enter the following:

```
mv environment.plist .MacOSX
```

The *environment.plist* file is moved into the *.MacOSX* directory.

c To confirm the *environment.plist* file is located in the *.MacOSX* directory, enter:

```
cd .MacOSX
```

This command moves you into the *.MacOSX* directory.

d Enter:

```
ls
```

The content of the *.MacOSX* directory, the *environment.plist*, is listed.

16 Log out and then log in again.

To edit the .plist file:

1 In the Finder, choose Go > Go to Folder (or press Command-Shift-G).

2 In the "Go to the folder:" text field, enter the path to the invisible *.MacOSX* folder:

```
/Users/john/.MacOSX
```

3 Click Go.

The *environment.plist* file appears in the folder.

4 Open the *.plist* file in TextEdit (or another text editor).

5 Once your changes are made, choose File > Save.

6 Quit TextEdit.

Using the .tcshrc Environment File

You can also set environment variables (or aliases) using a *.tcshrc* file. Like the above *.plist* file example, you can create the *.tcshrc* file in a text editor, or directly in a shell using vi, pico, or another shell editor. Unlike the *.plist* file, however, you do not save the *.tcshrc* file to the *.MacOSX* directory. Instead, the *.tcshrc* file is saved into your Home ($HOME) directory.

Usually, you define environment variables in tsch with the *setenv* command, for example:

```
setenv audio /Volumes/shared/footage/audio_files/
```

This variable instructs Shake to automatically look in */Volumes/shared/footage/audio_files/* when you import an audio file into Shake.

At login, your computer runs the default */etc/csh.cshrc*, followed by any *.tcshrc* files in your login directory. This sequence is repeated whenever a new tsch is spawned—for example, when you launch Terminal.

Note: As mentioned above, Shake only reads the *.tcshrc* environment file when Shake is run from the Terminal (the file is not applied when Shake is launched from the application icon).

To add a variable for Terminal commands, enter the following formatting (edit to suit your own project) into *$HOME/.cshrc* or *$HOME/.tcshrc*:

```
setenv NR_INCLUDE_PATH " //MyMachine/proj/include;/Documents/shake_settings/
     include"
```

The following is an example of a *.tcshrc* file for illustration purposes only:

```
setenv shake_dir /Applications/Shake4/shake.app/
     Contents/MacOS/shake
setenv shk_demo /Documents/project_03
set path = (. $shake_dir $path)
setenv NR_INCLUDE_PATH /Documents/project_03
setenv NR_FONT_PATH /System/Library/Fonts
alias ese vi $HOME/.tcshrc
alias s. source $HOME/.tcshrc
alias lt ls -latr
alias ui cd $HOME/nreal/startup/ui
alias st cd $HOME/nreal/include/startup
alias shake $shake_dir
```

This file sets the Shake directory, as well as points to the directories that you want Shake to scan for macros, user interface settings, and so on. (*/Documents/project_03*), and fonts (*/System/Library/Fonts*).

Note: Alias definitions or environment variables saved in a *.tcshrc* file are read the next time you log in. To make the alias or environment variable effective immediately, update your alias definition by sourcing out *.tcshrc*. Type the following:

```
source .tcshrc
```

To edit the *.tcshrc* file, use pico or vi (or another shell editor). Once your changes are made, save the *.tcshrc* file.

Shake Variables

Shake recognizes the following variables:

- *shell variables*: The File Browser recognizes an environment variable, for example, *$pix* in the Browser if Shake is run with that environment setting.
- *NR_CINEON_TOPDOWN*: When set, that is,

  setenv NR_CINEON_TOPDOWN

Cineon frames are written in the slower top-down method for compatibility with other, less protocol-observant, software.

- *NR_FONT_PATH*: Points to a directory where you want Shake to scan for additional fonts used by the *Text/AddText* functions. In Mac OS X, fonts are stored in *<mycomputer>/Library/Fonts* and *$HOME/Library/Fonts*. On Linux systems, fonts are typically stored in */usr/lib/DPS/AFM*.
- *NR_ICON_PATH*: Points to a directory where you can save your own icons for Shake functions. Typically, this would be an *nreal/include/startup* directory that you create.
- *NR_INCLUDE_PATH*: Points to the directory or directories that you want Shake to scan for macros and personal machine or Shake interface settings. These directories should have *startup/ui* as subdirectories. For example:

  setenv NR_INCLUDE_PATH /shots/show1/shake_settings

  should have */shots/show1/shake_settings/include/startup/ui*
- *NR_SHAKE_LOCATION*: Points Shake to a nonstandard installation area. Default installation is */usr/nreal/<ShakeDir>*.
- *NR_TIFF_TOPDOWN*: This is identical for *NR_CINEON_TOPDOWN*, except it applies to TIFF files.
- *TMPDIR*: Points to the directory you want to use as your temporary disk space directory.
- *NR_GLINFO*: Information is printed for Flipbooks.

To Test Your Environment Variable

There is a simple way to test if your environment variable exists. In Terminal, type "echo," followed by the environment variable, for example:

```
echo $myproj
```

and the proper value should be returned.

Using Aliases

An alias is a pseudonym or shorthand for a command or series of commands, for example, a convenient macro for a frequently used command or a series of commands. You can define as many aliases as you want (or, as many as you can remember) in a *.tcshrc* file.

To see a current list of aliases, type the following in a shell:

```
alias
```

To start Shake from the Terminal window:

```
Alias shake /Applications/Shake4/shake.app/Contents/MacOS/shake
```

To determine how many users are currently working on the system:

```
Alias census 'who | wc -l'
```

To display the day of the week:

```
alias day date +"%A"
```

To display all Shake processes that are running:

```
alias howmany 'ps -aux | grep shake'
```

Interface Devices and Styles

This section discusses considerations when using a stylus, setting mouse behavior, using a two-monitor system, and setting the monitor resolution.

Using a Stylus

1. In the Globals tab, open the guiControls subtree.
2. Set the virtualSliderSpeed parameter to 0.

When virtualSliderMode is enabled, dragging left or right in a value field decreases or increases the parameter value beyond the normal slider limits.

Note: The stylus does not allow you to use your desktop space the same way as with a mouse, so you have to enable virtualSliderMode.

Dual-Head Monitors

Choose View > Spawn Viewer Desktop to create a new Viewer window that floats above the normal Shake interface. You can then move this Viewer to a second monitor, clearing up space on the first for node-editing operations.

Important: This only works when both monitors are driven by the same graphics card.

The following is a handy *ui Directory* command to automatically create the second Viewer Desktop:

```
gui.dualHead= 1;
// This is an example of what you can do to open a second
// viewer desktop on the other monitor.
if(gui.dualHead) spawnViewerDesktop(1290,10,1260,960);
```

For information on using a broadcast video monitor, see "Viewing on an External Monitor" on page 330.

Customizing the Flipbook

The following arguments have been added to the Flipbook executable as global plugs, allowing you to specify an external Flipbook as the default. Specify these plugs using a .h file in the *startup* directory. The global plugs and their default values are:

```
gui.externalFlipbookPath = "shkv"; // the flipbooks name -- this
     should include the full path
gui.flipbookStdInArg = "-"; // instructs the flipbook to take data
     from StdIn
gui.flipbookExtraArgs = ""; // allows you to enter any extra
     arguments the flipbook needs.
gui.flipbookZoomArg = "-z"; // sets the zoom of the flipbook
gui.flipbookTimeArg = "-t"; // the time range argument
gui.flipbookFPSArg = "-fps"; // the frames per second argument
```

Note: If the specified external Flipbook doesn't support one of these arguments, setting its value to an empty string ("") prevents that value from being passed to it.

Configuring Additional Support for Apple Qmaster

You can enable additional support for Apple Qmaster by adding the following global plug to a .h file in the *startup* directory:

```
sys.useRenderQueue = "Qmaster";
```

This setting causes additional options to appear in the Render Parameters window when you choose Render > FileOut Nodes. These options become visible when you open the renderQueue subtree.

If Apple Qmaster isn't installed but the sys.useRenderQueue plug is declared, a message is sent to the console upon startup, and the following options do not appear.

RenderQueue Options

- *queueName:* The name of the render queue software being used. If Apple Qmaster is installed, "Qmaster" appears here.
- *useQueue:* When useQueue is turned on, the *FileOut* nodes specified by the renderFileOuts parameter are sent to the render queue when you click Render. By default, useQueue is turned off. Setting renderFileOuts to All sends all *FileOut* nodes to the render queue software. Setting renderFileOuts to Selected only sends the selected *FileOut* nodes to the render queue software.
- *jobTitle:* Enter the name you want to use to keep track of this job here.
- *workingDir:* The directory in which you want to store the temp script used by the render queue. The temp script is a temporary duplicate of your script that the computers in the specified cluster can access to perform the job.
- *cluster:* A pop-up menu that allows you to choose which cluster you want to use to perform the job. All clusters set up in your render queue software will appear here.

- *minFrames:* Use this field to specify the minimum number of frames you want to be processed by each computer in the cluster.
- *timeout:* The time, in seconds, a computer on a cluster can be idle before that part of the job is re-routed to another computer.
- *priority:* A pop-up menu that allows you to choose the priority of the job. The options delay: A pop-up menu that allows you to delay when the render queue software starts the job you're submitting. The options are 15 minutes, 30 minutes, 1 hour, or 2 hours.
- *batchMonitor button:* Click batchMonitor to launch the Apple Qmaster Batch Monitor application.

Part II: Compositing With Shake

II

Part II contains detailed information on how to perform compositing tasks using all the tools and functions Shake provides.

Image Processing Basics

15

Shake gives you explicit control over every aspect of image processing. This chapter covers the basics of image processing, and how to control the flow of image data in your own Shake scripts.

About This Chapter

This chapter covers important information about topics that are fundamental to compositing within Shake.

These topics include:

- Resolution handling via the Infinite Workspace
- Bit depths
- Alpha channel information
- Premultiplication
- Logarithmic colorspace support

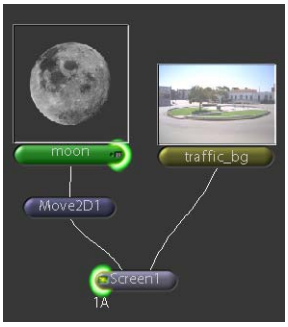
If you're a new user, or an experienced user who has always wondered why some things don't seem to turn out like you'd expect, it is well worth your time to review this information to better understand and control how images are processed in your scripts.

Taking Advantage of the Infinite Workspace

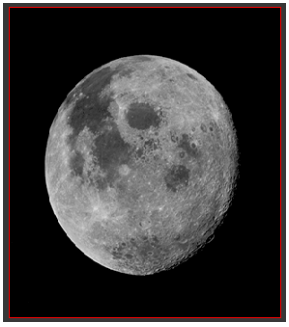
One of the most powerful features of Shake is the Infinite Workspace. Shake optimizes image processing by rendering only the portions of each image that are currently exposed within the frame, no matter what the original resolution.

This means that if, for example, you have a very small element that is 100 x 100 pixels, and you pan it 50 pixels in the X axis and 50 pixels in the Y axis, three-fourths of the image will extend outside of the 100 x 100-pixel frame. Although Shake does not calculate the "unseen area," the portion of the image outside the boundary of the frame is preserved. If, later in your script, you composite that image over another image with a 400 x 400 pixel frame, the parts of the image that were previously outside of the frame reappear. Because of the Infinite Workspace, you never lose image data as a result of transformations or resolution changes.

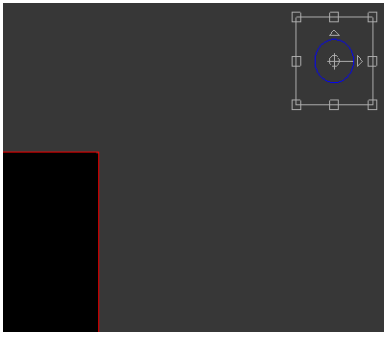
In the following example, the *moon* image is scaled, and panned up and to the right, resulting in the image moving completely offscreen. When the *traffic* image is later composited with the *moon* image using a *Screen* node, the hidden *moon* image appears in its entirety.



Tree



Moon image



Move2D1

Screen1

Even though Shake calculates only the visible parts of the image within the frame, the image information outside of the frame is preserved for later use. This powerful feature optimizes the operations within your script, has almost no memory or calculation cost, and eliminates many potential difficulties when combining and transforming images of varying resolutions.

Nodes that modify image resolution also take advantage of the Shake Infinite Workspace. For example, if you apply a *Crop* node to a 20,000 x 20,000-pixel image, Shake calculates only the area of the image specified in the node. This is true even if you employ other nodes prior to the *Crop*. The Infinite Workspace allows Shake to limit the processing power needed by your script, because only the contents of the *Crop* window are calculated. This makes Shake ideally suited for high-resolution functions such as scrolling a large background image under lower-resolution foreground elements.

When working with the Infinite Workspace, bear in mind the following:

- You do not need to crop a small image before it is composited with a larger image when you are panning the image. Simply read in your image, apply the pan, and composite it over or under the larger image.
- The *Blur* node gives you the option to blur pixels outside of the frame using the spread parameter. When set to 0, only pixels inside the frame are considered. When set to 1, outside pixels are calculated into the blur as well. When you read in an image and then blur the image, set the spread to 0. Otherwise, black ringing occurs around the edge because Shake adds the empty black region beyond the image border into its blur calculation. If you read in the image, scale it up, and then blur, you should set spread to 1, since there are now non-black pixels outside of the frame.

Clipped Images

If an image is clipped, it is usually because a *Crop* node has been applied, or because you have applied a *Blur* with a spread set to 0, which is including black outside the image area. Set spread to 1 in the *Blur* node's parameters.

spread = 1

spread = 0

- If you transform an object, apply a color correction, then transform the object back to its original state, the entire image retains the result of the color correction, not just the portion that was in the frame when you applied the color correction.

Note: You must be careful when pulling a bluescreen matte with the *ChromaKey* node. The outside black pixels are considered invisible because the node is keying a non-black color.

To disable the effect of the Infinite Workspace, insert a *Crop* node and don't modify its default values (which does not change the resolution). This cuts off the area outside of the frame, replacing it with black pixels. The *Viewport* node is similar to *Crop*, but it does not disable the Infinite Workspace.

Note: Be very careful when scaling elements up, applying an operation, then scaling back down. When you apply an operation to the scaled element, even though your frame is small, Shake will calculate everything outside of the frame when you scale the image back down to fit in the frame.

For more information on the Infinite Workspace, see "Color Correction and the Infinite Workspace" on page 617. You can also see "The Domain of Definition (DOD)" on page 82.

Bit Depth

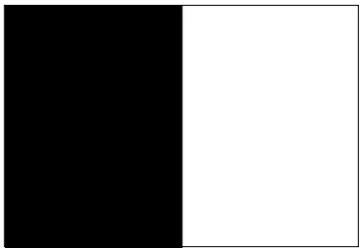
Bit depth describes how many values are used to describe the range of colors in an image. The number of steps in a range of color is calculated by taking 2 to the *n*th power, where *n* represents the number of bits. For example, a 1-bit image gives you two values—black and white. A 2-bit image gives you 2^2, or 4 color values per channel. Bit depth directly affects image quality in several ways.

Comparing Different Bit Depths

Higher bit depths allow you to more realistically represent a wider range of color, by ensuring that the gradients between similar colors are smooth. Using a bit depth that's too low results in what is sometimes described as color banding—where, for example, you can actually see the limited number of colors in between two shades of blue.

For a better understanding of how this happens, you can look at how a range of color is represented at varying bit depths on a graph. In a simplification, the following charts display a grayscale ramp in 1-bit, 2-bit, 3-bit, and 8-bit depths.

Note: These examples of 1-bit, 2-bit, and 3-bit images are not supported by Shake, but are used for demonstration purposes. In Shake, you ordinarily work with 8-bit, 16-bit, or 32-bit float (floating point) images.



1 bit, 2 values total

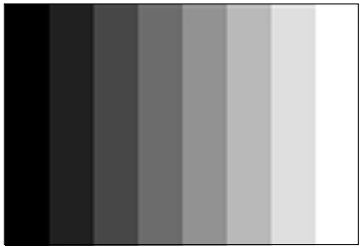
Graph of 1-bit image

At 1-bit resolution, the graph shows the harsh difference between black and white.

2 bits, 4 values total

Graph of 2-bit image

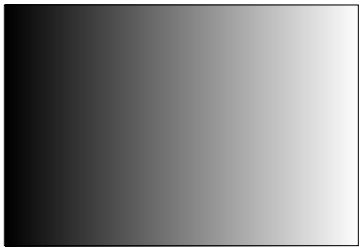
At 2-bit resolution, the graph is still harsh, but there are more colors between.



3 bits, 8 values total

Graph of 3-bit image

At 3-bit resolution, you begin to see a gradient from black to white, although the graph is still choppy.



8 bits, 256 values total

Graph of 8-bit image

Finally, at 8 bits, you can see a smooth transition, and the graph line is almost straight.

These graphs demonstrate that more bits used to represent an image results in finer color transitions. Digital film compositing refers to bit depth on a per-channel basis, so 8 bits refers to 8 bits per channel, or 32 bits total for an RGBA (Red, Green, Blue, and Alpha) image. Shake can calculate up to 32 bits per channel, or 128 bits total in an RGBA image. To complicate things, because 8 bits equals 1 byte, images in Shake are set with a byte value of 1 (8 bits), 2 (16 bits), or 4 (32 bits, or *float*).

Ultimately, the color depth you decide to work in depends on the destination of the end result. For example, because of the responsiveness of film and the size of the screen, an image that looks fine at 8 bits on video can look terrible on film. On the other hand, higher bit depths are more processor-intensive. You need to strike a balance between quality and speed.

Avoiding Color Banding

Most non-film composites work fine at 8 bits, which is typically the standard output of most 3D renderers and paint packages. However, there are times when you need to use a higher bit depth to process your images—requiring you to increase an image's bit depth to 16 bits, or a whopping 65,000 (more or less) values per channel.

A typical example of when higher bit depths are better is whenever you process a ramp of similar color values (for example, the light-to-medium blue found within an image of the sky) across a wide screen space. An 8-bit image, though sometimes indiscernible from 16 bits on a computer monitor, will probably exhibit color banding when printed to film. If your sky image is generated in 8 bits in a different software package, there is no immediate improvement if you bump it up to 16 bits in Shake. In this example, the ramp needs to be generated in 16 bits to take advantage of the extra precision of 16 bits (for example, using the Shake *Ramp* or *RGrad* node).

Note: In 8-bit images there is no 50 percent point—you have a smidgen less than 50 percent gray and a smidgen more than 50 percent, but you cannot get an exact 50 percent value. This occasionally becomes an issue when creating and using macros.

If this is the case, then why not always work at 16-bit resolution? Most film houses do, but it comes at the expense of slower calculations, more memory required, and larger-sized image files requiring significantly more hard disk space.

Float

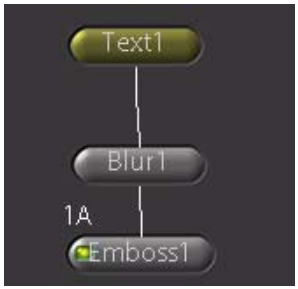
The Shake 32-bit representation is "float"—values can go above 1 or below 0. In all of the above examples, the ramp ranges from 0 to 1. If you add two 8-bit ramps together, the white values are added together (1+1), but clipped at 1. This is fine visually, but you may later do other mathematical computations in which it is important to realize that 1+1 is 2, not 1. A good example is the Z channel, which is always in float. The Z channel is usually generated by a 3D render, and supplies the distance on a per-pixel basis from the object to the "camera." Therefore, values could go from 0 to infinity. If you swap your Z channel into your red channel, you do not want it clipped off at 1, because you could not tell the difference between the pixels that are 2 units away and the pixels that are 1000 units away. A float representation, however, maintains these values.

Bit Depth Independence

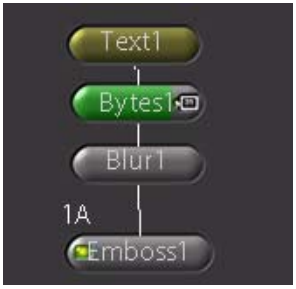
Shake recognizes and maintains the bit depth of incoming images—except for 10-bit Cineon files, which are automatically boosted to 16 bits. Because Shake concatenates color corrections, in Shake you are penalized less frequently when working at 8 bits than you are in other software. This is because adjacent color corrections are collapsed into a single mathematical lookup table, enabling Shake to perform the overall computation in float. The resulting image is returned to its source bit depth.

With the use of the *Bytes* node, you have the option of modifying your image to a higher or lower bit depth. As the name implies, the *Bytes* node takes bytes as its argument, so a value of 1 equals 8 bits, 2 equals 16 bits, and 4 equals 32 bits (or float). (There is no "3 bytes" setting.) For information on the *Bytes* node, see "The Bytes Node" on page 413.

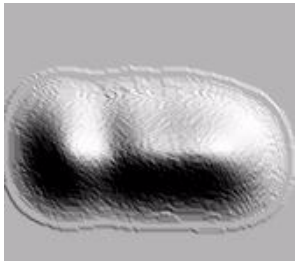
You might need to use a higher bit depth when employing certain nodes, such as *Emboss* and *Blur*, since they naturally create smooth gradations. In the following example, the image on the left has a *Blur* node and an *Emboss* node applied. At 8 bits, terracing appears. By inserting an Other—*Bytes* node at the beginning of the node tree set to 2 bytes (16 bits), the *Emboss* effect is smoothed.



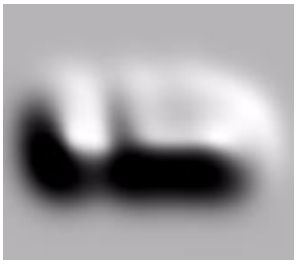
8 bits



16 bits



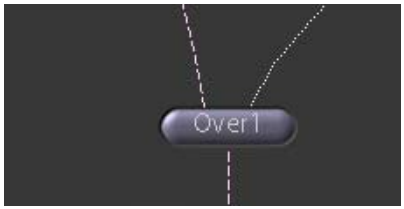
2-bit image, 4 values per channel possible



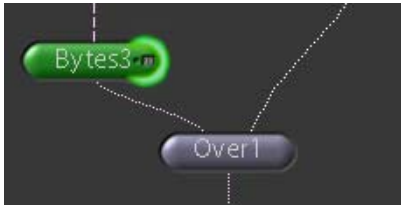
16-bitimage, 65,535 values per channel possible

Be sure to increase the bit depth of the image before the *Blur* node. This does not mean you need to insert a *Bytes* node before every *Blur*. Rather, use the *Bytes* node when you plan to apply numerous operations to a blurred image. Why? Because blurred images are likely to display unwanted banding after multiple operations are applied.

You can seamlessly layer images of different bit depths together. This results in the lower bit-depth image being automatically promoted to the higher of the two bit depths. (For example, an *Over* node compositing an 8-bit image with a 16-bit image results in a 16-bit image.) This is an automatic operation, invisible to the user.



To reverse this behavior, insert a *Bytes* node before the *Over* node on the 16-bit image to reduce the image to 8 bits.



Bit-depth level is calculated locally in the node tree. In the previous example mixing 8-bit and 16-bit images, only the sections of the node tree that come after the 8-bit to 16-bit conversion in the *Over* node are calculated at 16 bits.

Cineon File Bit Depth

10-bit Cineon files are automatically promoted to 16 bits when read in or written by Shake, so you don't have to worry about any data loss. However, the linearization of the log files may result in loss unless you first promote your images to float. For more information, see "The Logarithmic Cineon File" on page 437.

The Bytes Node

The *Bytes* node converts the input image to a different bit depth. The bit depth is counted in bytes per channel. To view the current bit depth of an image, you can look at the title bar of the Viewer, or look at the output of a Shake *-info* in the Command Line field at the bottom of the interface. 10-bit Cineon files are automatically pushed to 16 bits when read by Shake.

Note: When compositing images of different bit depths, you do not need to force the images to conform; Shake automatically pushes the lower bit-depth image to the higher bit depth.

Parameters

This node displays the following control in the Parameters tab:

outBytes

Forces the incoming image into a new bit depth. There are three buttons, corresponding to three values in the outBytes parameter field.

- 1 = 1 byte per channel, or 8 bits per channel.
- 2 = 2 bytes per channel, or 16 bits per channel.
- 4 = 4 bytes per channel, or 32 bits per channel (float).

Channels Explained

Shake supports and tracks different numbers of channels in an image in your composition, giving you channel independence as well as bit-depth and resolution independence.

For information on displaying different channels in the Viewer, see "Using and Customizing Viewers" on page 45.

Code	Description
BW (Black and White)	1-channel grayscale image.
A (alpha)	1-channel matte image.
Z (depth)	1-channel depth image, always in float.
BWA	2-channel grayscale image with matte channel.
RGB	3-channel color image, representing Red, Green, and Blue information.
RGBA	4-channel color image with matte channel.

An additional Z channel can be added to any of the above, so the maximum number of channels you can use to represent a single image is five: RGBAZ. Unfortunately, the Z channel does not show up in the Viewer unless you use the View Z script button. (Whether or not the View Z script is active, you can always check the Viewer title bar to see whether currently loaded image contains a Z channel.)

Combining Images With Different Channels

In Shake, you can combine images that use different channels. For example, you can composite a 2-channel image over a 4-channel image.

Shake is optimized to work on a per-channel basis—a 1-channel image usually calculates about three times faster than a 3-channel image. For this reason, if you read in masks from a different package, you are encouraged to make them 1-channel images to reduce disk space usage and processing time.

If you apply an operation that changes channel information, Shake automatically updates which channels are used. For example, if you place a Color–*Monochrome* or Filter–*Emboss* node on an RGB image, that image becomes a BW image at that point, speeding up the calculation of the subsequent nodes. If you then composite the image over an RGB image or change its color (for example, via a *Mult* node with values of 1.01, 1, 1), the BW image becomes an RGB image again.

In certain situations, this behavior may seem nonintuitive. For example, a 3-channel image composited with an *Inside* node to a matte image still results in a 3-channel image—no matte channel is added to the result. This eliminates the need to add an alpha channel to the 3-channel image just to combine it. If, however, you want to add or remove channels at some point, you can use the *Copy*, *SwitchMatte*, Color–*Set*, or Color–*Reorder* node.

Viewing the Number of Image Channels

There are several ways you can see how many image channels are being used by the current node.

To determine the number of channels in an image, do one of the following:

- Load the image into the Viewer, then the look in Viewer title bar display.
- Position the pointer over the node in the Node View and look at the Info field at the bottom of the interface.
- In the Command Line field, type:

  shake my_image -info
- To view the Z channel, use the View Z script button.

For more information on displaying channels in the Viewer, see "Using and Customizing Viewers" on page 45.

Displaying Individual Channels in the Viewer

If necessary, you can display each individual channel in the Viewer to help you fine-tune your composite.

To view the alpha channel of an image, do one of the following:

- Position the pointer in the Viewer (or the Flipbook), then press A.
- Click the View Channel button, then choose the alpha channel option from the pop-up menu.

To return to viewing the RGB channels, do one of the following:

- Position the pointer in the Viewer, then press C.
- Click the View Channel button, then choose the RGB channel option from the pop-up menu.

To display the R, G, or B channels individually, do one of the following:

- Press R to display the Red channel.
- Press G to display the Green channel.
- Press B to display the Blue channel.
- Click the View Channel button, then choose a color channel from the pop-up menu.
- Right-click the View Channel button, then choose a color channel from the shortcut menu.

Changing the Number of Image Channels

As stated above, certain operations automatically add or remove channels. For example, the *Emboss* and *Monochrome* nodes change an RGB image to a BW image, and a non-uniform *Add* node changes a BW image to an RGB image. You can also explicitly change the number of channels in an image with specific nodes.

The following nodes also potentially modify image channels:

Node	Effect	Operation
Color–*Add*	Adds R, G, B, A, or Z.	A value raised above 0 creates the specified channel.
Color–*Brightness*	Removes RGB.	A brightness value set to 0 removes the RGB channels.
Layer–*Copy*	Adds R, G, B, A, or Z.	Copies a channel from the second input to the example. If you copy Z, the second image must have the Z channel.
Filter–*Emboss*	Turns an RGB image to a BW image.	This, of course, radically alters your image.
Color–*Monochrome*	Turns an RGB image to a BW image.	Uses a luminance balance, but you can adjust this to push specific channels.
Color–*Mult*	Removes R, G, B, A, or Z.	Setting the R, G, B, A, or Z to 0 removes the specified channels.
Color–*Reorder*	Adds or removes R, G, B, A, or Z.	By using *n* or 0, you remove the specified channel: • *rgbn* or *rgb0* removes the alpha channel. • *rgbal* adds the luminance into the Z channel, thereby creating a Z channel. • *rrra* creates a 2-channel image, assuming the "a" channel is not black. • *000a* or *nnna* turns the image into a 1-channel alpha image.

Node	Effect	Operation
Color–*Set*	Adds or removes R, G, B, A, or Z.	A value set to 0 removes the specified channel. A channel parameter set to something other than 0 adds that channel.
Layer–*SwitchMatte*	Adds A.	Copies in any channel from the second input for use as the new alpha channel for the first input.

Many operations allow you to select which channel is used as the modifying channel. For example, the *SwitchMatte*, *KeyMix*, and *IBlur* nodes give you the option to select the R,G,B, or A channel as your control or alpha channel. This often removes the need to swap your channels before you do many operations. Two exceptions to this are the *Inside* and *Outside* nodes, which always depend on the second image's alpha channel.

To convert a BW (or BWA) image into an RGB (or RGBA) image without changing its values, use the Command Line field to specify the following operation:

```
shake myBlackAndWhiteImage.iff -forcergb -fo myRGBImage
```

For more information on channel/compositing functions, see Chapter 16, "Compositing With Layer Nodes," on page 451.

Compositing Basics and the Alpha Channel

The Shake compositing nodes are located in the Layer tab. The primary compositing nodes are the *Over* and *KeyMix* nodes. For more information on how to use these nodes, see Chapter 16, "Compositing With Layer Nodes," on page 451.

Example 1: Compositing Using the Over Node

The following images are used in this discussion to demonstrate how the primary compositing nodes work:

Foreground (with mask)

Foreground alpha channel (part of foreground image)

Background

As its name implies, the *Over* node places a foreground image with an alpha channel over a background image. The foreground RGB channels should be premultiplied by the alpha channel. In a premultiplied image, the RGB channels are multiplied by the alpha channel.

Important: Premultiplication plays a vital role in compositing, and Shake gives you explicit control over premultiplying and unpremultiplying your images. For more information, see "About Premultiplication and Compositing" on page 421.

Tree with *Over* node

Result

If the image is not premultiplied, it can be premultiplied in one of two ways:

- Add a Color–*MMult* node before the *Over* node in the process tree.
- Use the preMultiply toggle in the *Over* node's parameters.

Example 2: Compositing Using the KeyMix Node

KeyMix, the second most important compositing node, mixes a foreground input image and a background input image through a third, separate, input image—the mask. You can select which channel of the third image works as the mask. You can also invert the mask and control its intensity.

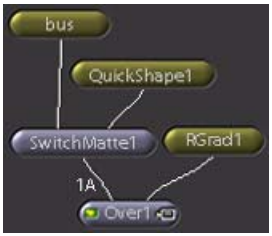
As mentioned previously, a successful *Over* composite requires an alpha channel for the foreground and foreground RGB channels that are premultiplied by that alpha channel. 3D-rendered elements are almost always premultiplied. Scanned elements or other 2D-generated plates require an added alpha channel (also called the matte or mask channel) that is used to premultiply that image with the Color–*MMult* node.

To get the necessary alpha channel, you have several options:

- Pull a key with a Shake keying node (or combination of nodes).
- Pull a key in a different software package and read the images into Shake. Copy the key into the alpha channel of the foreground image (with the *Copy* or *SwitchMatte* node). Finally, apply an *MMult*, and then composite.
- Draw a mask with an Image–*RotoShape* node.
- Paint a mask with an Image–*QuickPaint* node.
- All of the above, combining masks with the *IAdd*, *Max*, *Inside*, or *Outside* node.

Example 3; Assigning an Alpha Channel With the SwitchMatte Node

In the following example, the mask is drawn using the *QuickShape* node and copied in as the alpha channel for the bus via the *SwitchMatte* node. Because no color corrections have been made, the MatteMult toggle is used in *SwitchMatte* to premultiply the foreground.



Tree with *SwitchMatte* node

Matte bus (no alpha)

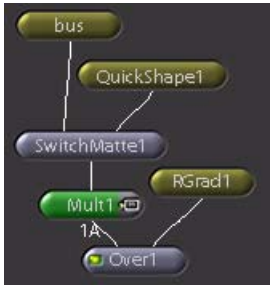
QuickPaint

Background

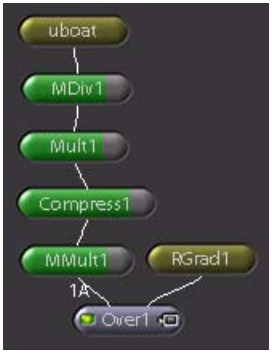
Premultiplied result of *SwitchMatte* node

Over result

The bus is color corrected in the next example, so preMultiply is disabled in the *SwitchMatte* node, and enabled in the *Over* node. Or, you can also insert a Color–*MMult* node between *SwitchMatte1* and *Over2*.



In the following example, *MDiv* and *MMult* nodes are added to color correct a 3D-rendered element. Again, you can alternatively omit the *MMult*, and enable preMultiply in the Layer–*Over* node.



Color correcting

Result

For an example of color correcting premultiplied elements, see Tutorial 3, "Depth Compositing" in the *Shake 4 Tutorials*.

Combining Images in Other Ways

Shake also has a set of mathematical and Boolean layering operators. The *IAdd*, *IDiv*, *IMult*, *ISub*, and *ISubA* nodes add, divide, multiply, or subtract two images together. The "I" stands for "Image." The second subtracting node, *ISubA*, returns the absolute value of the image. If you place a dark gray image in the first input (value .2, .2, .2), and then a white image (1, 1, 1) in the second input, the *ISubA* operation, it returns a light gray image (.2 - 1 = -.8, take the absolute value of, returning .8, .8, .8). This is a quick way to compare two images.

The more flexible tool for generating difference mattes is the *Common* node, which is used to isolate common or different elements between two images. The other mathematical operators are *Min* and *Max*, which return the lower or higher values of two images, respectively. The Boolean operators *Inside*, *Outside,* and *Xor* are also useful for masking images:

- *Inside* places the foreground image inside of the background alpha.
- *Outside* places the foreground image outside of the background alpha.
- *Xor* reveals only areas without a common alpha area.

Shake contains several effect operators:

- *ZCompose* for Z-based compositing
- *Screen*, to composite light elements and preserve highlights
- *Atop*, to place an element only on the foreground image. (For example, you can use the *Atop* node to place a smoke element over a CG character, matching smoke in the background plate.)

Using the ClipMode Parameter of Layer Nodes

You can easily composite elements of any resolution. To set the output resolution of a composite that contains images of multiple resolutions, go to the compositing node's parameters and use clipMode to toggle between the foreground or background (as the output resolution). This applies to all layering commands. An element composited over a differently sized background is one way to set your output resolution. For more information on setting resolution, see Chapter 3, "Adding Media, Retiming, and Remastering," on page 107.

As outlined in the "About Channels" section above, you can easily combine 1-channel, 2-channel, 3-channel, 4-channel, or 5-channel images. For example, you can combine a luminance image with an alpha channel (2 channels) over a 5-channel image, using an *Over* node.

For more information on compositing math and the individual layer functions, see Chapter 16, "Compositing With Layer Nodes," on page 451.

About Premultiplication and Compositing

An understanding of *premultiplication* in compositing is essential to understanding how to combine the tasks of compositing and color correction. This section details the process that makes standard compositing functions work, and defines what premultiplication actually is. Regardless of your compositing software, the concept of premultiplication is important for understanding what can go wrong, and how to fix it.

The definition of a premultiplied image is simple—an image that has its RGB channels multiplied by its alpha channel. Typically, images from a 3D renderer are premultiplied. This means that the transparent areas are black in both the RGB channels *and* in the the alpha channel. In premultiplied images, the RGB channels never have a higher value than the alpha channel.

Premultiplication should always be considered whenever you have to modify a foreground element and composite it over a background image. The premultiplication of two or more composited images should be considered whenever you do one of the following things:

- Perform color correction—in particular using nodes that raise the black level such as *Add*, *Clamp*, *ColorMatch*, *ColorCorrect*, *Compress*, and *Contrast*
- When using filtering nodes

Some software packages take care of image premultiplication for you—they hide the state of premultiplication. This works fine nine times out of ten, but for that last problematic 10 percent, there is no practical solution. Still other compositing packages pretend premultiplication doesn't exist and encourage bad habits, such as chewing on your matte, or modifying foreground/background alpha multiplication curves.

Shake takes a third approach, giving you explicit control over premultiplication for every image in your composition. Although this can be inconvenient, it helps you get around the problems that are typical in other software packages.

In a Nutshell—the Rules of Premultiplication

If you don't read the full explanation of the mathematics of premultiplication, here are the two rules you must *always* follow when creating a composition in Shake:

- *Rule Number 1:* Always color correct unpremultiplied images. To unpremultiply an image, use a Color–*MDiv* node.
- *Rule Number 2:* Always filter and transform premultiplied images. To premultiply an image, use a Color–*MMult* node.

Problems Caused When Premultiplication Is Ignored

There are two typical problems that occur when the premultiplied state of an image is ignored:

- Unwanted fringing around a masked subject
- Unwanted side effects that occur when a node affects parts of the image that ought not to be affected

The following exercise demonstrates these problems.

To experience the heartbreak of premultiplication errors:

1. Open the Image tab, and click the *FileIn* node.

   The File Browser opens.

2. In the File Browser, navigate to the directory where you copied the tutorial media.

   The default path is *$HOME/nreal/Tutorial_Media/Tutorial_Misc/premult*.

3. Select the *bg.jpg* and *munch_pre.sgi* images, then click OK.

   The nodes appear in the Node View.

   The image *munch_pre.sgi*, rendered in Maya, is premultiplied (the blacks in the alpha match the blacks in the RGB), and has an embedded alpha channel.

munch_pre.sgi

munch_pre.sgi, alpha channel

bg.jpg

Images from *Munch's Oddysee* courtesy of Oddworld Inhabitants.

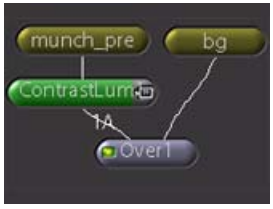
4 In the Node View, select the *munch_pre* node, click the Layer tab, then click the *Over* node.

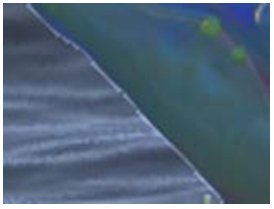
An *Over* node is added to the *munch_pre* node.

5 Connect the *bg* node to Background input (the second input) of the *Over* node.

6 Now, insert a Color–*ContrastLum* node between the *munch_pre* node and the *Over* node, and set the *ContrastLum* value to .5.

If you zoom into the edges, a white edge appears around Munch. This unwanted fringe is problem number one.





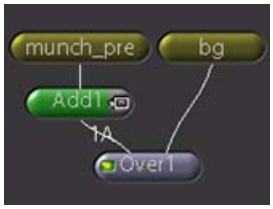
7 To replace the *ContrastLum* node, select it in the Node View and Control-click the Color–*Add* node.

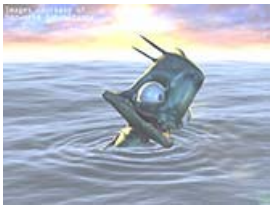
The *ContrastLum* node is replaced by an *Add* node.

8 In the *Add* parameters, boost the Color (red/green/blue) values to approximately .4.

Note: To adjust the color values, you can also press O (for Offset) and drag the pointer to the right over the *Add* color control.

Although the *Add* node is only applied to the *munch_pre* node, the entire image is brightened. A node affecting more of the image than it's supposed to is problem number two.





If you ignore the premultiplication of your composites, you may have problems with edges, or with raised global levels. Many people see these types of errors, and assume it is a mask problem, so they pinch in the mask a bit. Even more extreme people have erroneously assumed you cannot color correct premultiplied images. These problems are easily solved through proper management of premultiplication.

The Math of Over and KeyMix

To understand why premultiplication problems occur, it is best to cut straight to the heart of compositing by understanding how a standard composite operator (foreground through a mask over a background) works. This has nothing to do with Shake, but in fact was worked out in the 1970s by two clever fellows named Porter and Duff.

The following is the math equation they developed. In this equation, A is the foreground's alpha channel:

```
Comp = (Fg * A) + ((1-A)*Bg)
```

Rather than go into this too deeply, look briefly at the significant part, `(Fg*A)`. This is the definition of premultiplication—"RGB multiplied by its alpha." To avoid the math, you can build the composite with other nodes, in effect constructing an *Over* node from scratch.

The following example (continued from the above section) shows how to build the compositing equation.

To build the compositing equation:

1 Read in the *munch_unpremult.jpg* and *munch_mask.iff* images from the */Tutorial_Misc/premult* directory.

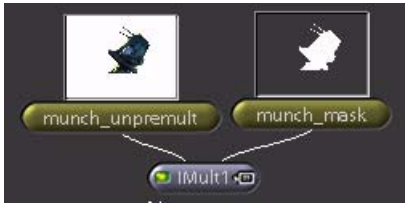
The *munch_unpremult* image is an unpremultiplied image. It has no mask, so there is no correspondence between black pixels in the RGB channels and black pixels in the mask to describe transparency in the image.

munch_unpremult.jpg

munch_mask.iff

2 The first step in the formula is `(Fg*A)`. To duplicate this using nodes, attach an *IMult* node to *munch_unpremult* and connect the *munch_mask node* to the *IMult* background input.

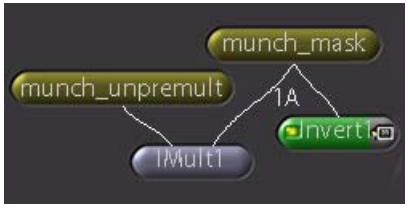
The result is identical to *munch_premult*—the RGB is multiplied by the mask.



Next, invert the foreground alpha channel (`1-A`).

3 To do this, select the *munch_mask* node, then Shift-click the Color–*Invert* node.

The *Invert* node is attached to the *munch_mask* node as a separate branch.

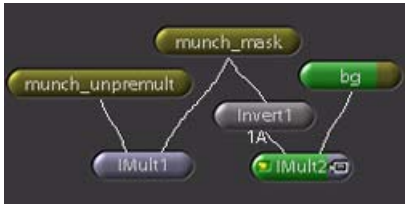


The next step in the formula (`(1-A)*Bg`) calls for you to multiply the background by the inverted alpha.

4 In the Node View, select the *Invert* node, then click Layer–*IMult*.

An *IMult* node is added to the *Invert* node.

5 Connect the *bg* node to the Background input of the *IMult2* node.



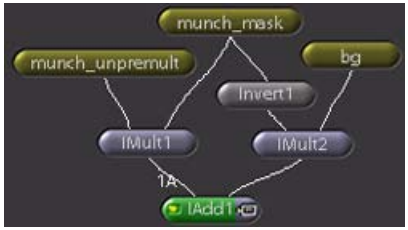
Tree **IMult2**

Next, the crucial step—add the two results together.

6 In the Node View, select *IMult1*, then click Layer–*IAdd*. Connect the *IMult2* node to the background input of the *IAdd* node.

Comp = `(Fg * A) + ((1-A)*Bg)`

The result is exactly the same as the *Over* node.



By punching a hole in the background, the alpha determines what is transparent when the two plates are added together. The key concept is that because you are adding, anything that is black (a value of 0) does not appear in the composite.

IMult1

IMult2

The *KeyMix* and *Over* nodes do this math for you, saving you from having to create four nodes to do a simple composite. The difference between *Over* and *KeyMix* is that *Over* assumes that the foreground is premultiplied (identical to *IMult1*). *KeyMix* is only for non-premultiplied images (identical to *munch_unpremult*). Strictly speaking, the math breaks down like this:

KeyMix = `(Fg * A) + ((1-A)*Bg)`

Over = `Fg + ((1-FgA)*Bg)`

In these formulas, the foreground of *Over* is already multiplied by the foreground alpha channel, thus the term *premultiplied*—it isn't multiplied in the composite because it was previously multiplied, usually by the 3D renderer.

Unpremultiplying an Image

With this knowledge, you can go back and start to understand the errors that occur when the *ContrastLum* and *Add* functions are used with the *Over* node.

An *Add* node was originally attached to the premultiplied Munch.

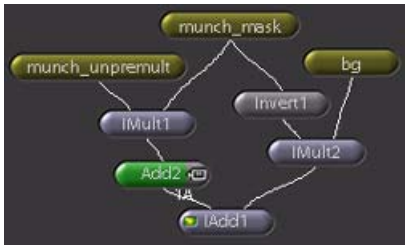
To create the same error here, continue with the previous node tree and do the following:

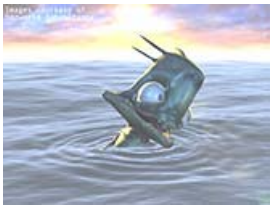
1. In the Node View, select the *IMult1* node, then click Color–*Add*.

   An *Add* node is inserted into the tree, between the *IMult1* node and the *IAdd1* node.

2. In the *Add* node parameters (click the right side of the node to load its parameters into the Parameters tab), set Color to .5.

   The same error occurs—the entire image brightens, not just Munch. This is not what we want to happen.



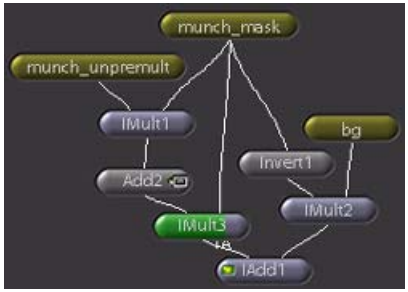


For a clue, you can double-click the *Add* (*Add2*) node. There is no longer an exact correlation between the black areas of *munch_mask* and the black areas of the RGB channels.

Add2

munch_mask

Now things get a little odd. To reassert the mask, you might be tempted to insert another *IMult* node after *Add2*, and connect the *munch_mask* node to the *IMult3* background input. The image is premultiplied again and *appears* to work.



Tree with IMult3 inserted

IAdd2

Although this looks fine, mathematically speaking, there is an error. You need to have the correct compositing equation:

Comp = `(Fg * A) + ((1–A)*Bg)`

But, in fact, the above example multiplies the foreground twice (there are two *IMult* nodes), so you have this:

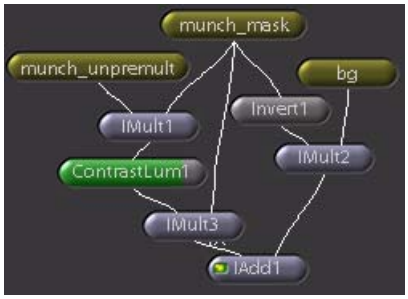
Comp = `(Fg * A * A) + ((1–A)*Bg)`

While this solution appears to have worked in this example, further manipulation of the resulting image data can reveal problems further down the tree. To test to see if this solution looks fine in all cases, switch the *Add* node to a *ContrastLum* node.

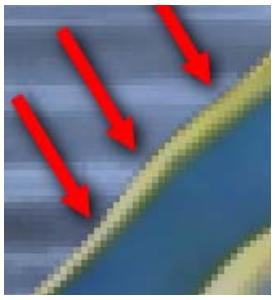
To test the equation:

1. In the Node View, select the *Add* (*Add2)* node in the tree, then Control-click Color–*ContrastLum*.
2. In the *ContrastLum* parameters, set the contrast value to .5.

If you zoom into the Viewer and look very closely, you'll notice a dark rim appears around the edge.



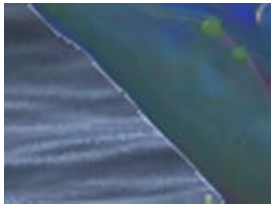
Add switched to ContrastLum



IAdd1 detail

3 In the Node View, select the *IMult3* node and press I.

The node is ignored, and the same nasty edges appear as before.



IAdd1 detail with IMult3 ignored

4 Press I again so the node is no longer ignored.

So, what is the solution? Simple math will help resolve this issue. Dividing something by a number and then multiplying by the same number is the same as multiplying by 1—the equivalent of no change at all. If you multiply the foreground by the mask one too many times, divide it by the alpha to balance it out.

Mathematically, here's how it looks:

Comp = `(Fg * A / A * A) + ((1–A)*Bg)`

Or, abbreviating the equation, you return to the proper formula, since the two extra alphas (A) cancel out:

Comp = `(Fg * A) + ((1–A)*Bg)`

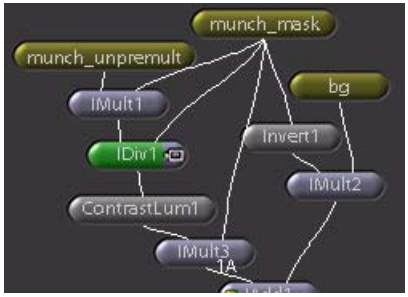
To translate this into the node tree:

1 In the Node View, select the *IMult1* node, then click Layer–*IDiv*.

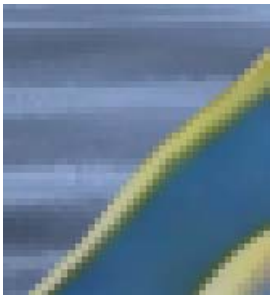
An *IDiv1* node is attached to the *IMult1* node.

2 Connect the *munch_mask* node to the *IDiv1* background input.

The edge appears clean.



IDiv inserted



IAdd1 detail

3 To test the result, select the *IDiv1* node and press I several times to ignore and show the node.

By dividing by the mask, the image is unpremultiplied and ready for color correction. Therefore, you can conclude that color correction should be applied to an unpremultiplied image.

After going through all this, you may be surprised to find out that the insertion of *IDiv1* and *IMult3* can be bypassed entirely.

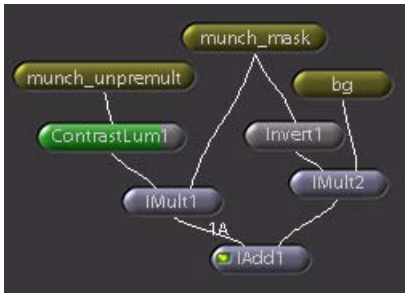
To bypass the insertion of IDiv1 and IMult3:

1 Delete the *IDiv1* and *IMult3* nodes.

2 Select the *ContrastLum1* node, then press E to extract it from the tree.

3 Drag the *ContrastLum1* node between the *munch_unpremult* node and the *IMult1* node to snap it back into the tree.

Since *munch_unpremult* is already an unpremultiplied image, you get the same clean result.



IDiv and IMult3 deleted, ContrastLum moved up

IAdd1 detail

Remember This

Rule Number 1: Only color correct unpremultiplied images.

Managing Premultiplication

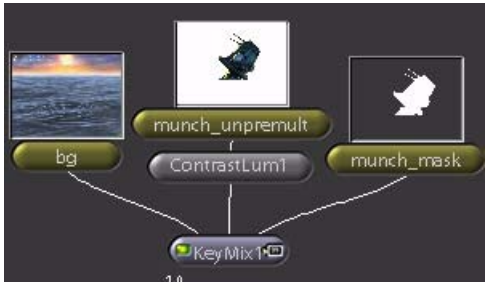
The above steps are an elaborate illustration to explain *Over*, *KeyMix*, and premultiplication. This explanation can be simplified a bit.

The basic difference between the *KeyMix* and *Over* nodes is:

- *KeyMix* is used for unpremultiplied foreground images. The alpha channel can be anywhere, including in the foreground image.
- *Over* is used for premultiplied foreground images, but can also enable premultiplication if necessary for unpremultiplied images. The alpha channel is in the foreground element.

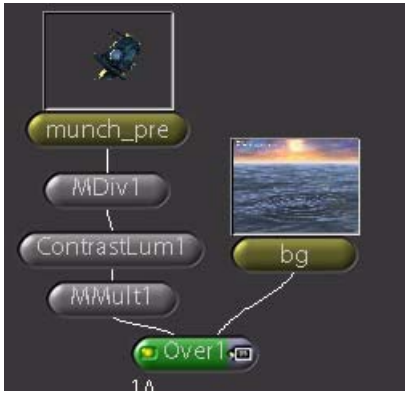
Shake does not require you to constantly separate your alpha channel and to perform *IMult* and *IDiv* operations with the second image. Instead, there are nodes to do this for you. The following examples duplicate the current complex tree with simplified versions.

This example uses the *KeyMix* node, which handles unpremultiplied foreground elements, the background, and a mask to split between the two. Enable invert in the *KeyMix* parameters (you could also switch the foreground and background inputs).



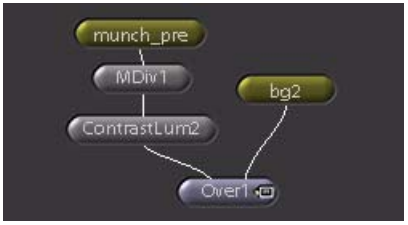
Identical tree using KeyMix

If you are using a premultiplied image that was rendered straight out of a 3D animation package, there are special tools that unpremultiply and premultiply the RGB channels by the alpha. These tools are Color–*MDiv* and Color–*MMult*. "MDiv" is "Mask Divide," and "MMult" is "Mask Multiply." Used in a node tree, *MDiv* unpremultiplies the image, then you color correct, and then premultiply using *MMult*. When using a premultiplied image as the foreground, the tree should look something like the following example.



Identical tree with a premultiplied foreground

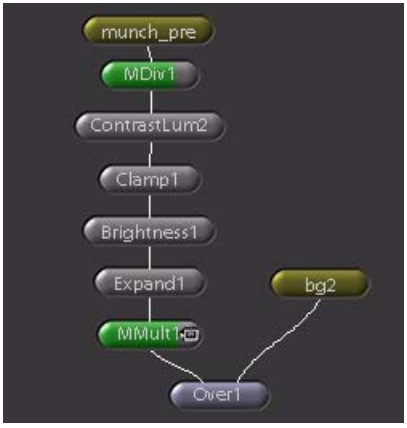
The *Over* node has a premultiplication parameter built into it. In the following tree, the preMultiply flag in the *Over* parameters is turned on, which allows you to omit the *MMult* node entirely.



Identical tree with premultiplication handled by Over1

Over1 parameters

You are not required to apply an *MDiv* for each color correction. You only need to add a single *MDiv* to the beginning of the branch of your node tree where you need to perform color correction. After the first *MDiv*, you can stack as many correctors up as you want.



Multiple corrections on one MDiv

After you've added all the color-correction nodes you require, add an *MMult* node to the end, and the image is ready for further compositing.

Filters and Premultiplication

Now to the second rule of premultiplication—using filters (such as *Blur*).

Remember This

Rule Number 2: Only apply *Filter* and *Transform* nodes to premultiplied images.

In the next example, an unpremultiplied image is accidentally filtered, to show you what artifacts to look for.

Adding a filter to an unpremultiplied image—the wrong way:

1 In a preexisting node tree, add a *Blur* node to the *munch_unpremult* node, in an attempt to blur it against the background.

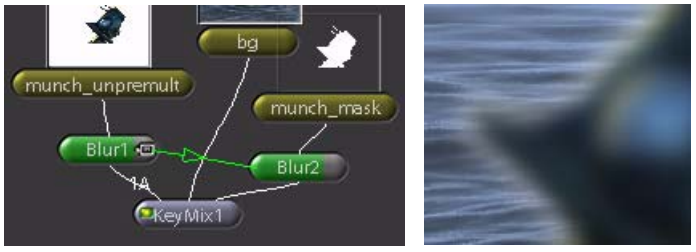
The mask clips any soft gradation.

2 Copy and clone the *Blur1* node. (Copy the node, then press Control-Shift-V.)

Note: Press Command-E or Control-E to activate the enhanced Node View and see that the cloned *Blur* node is linked to the *Blur1* node.

3 Connect the *Blur1_Clone1* node to the *munch_mask* node.

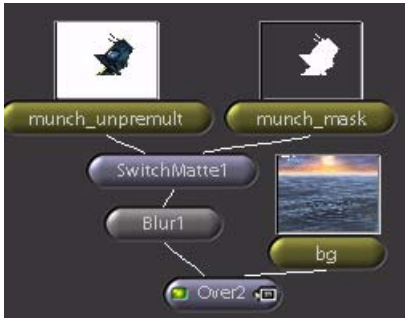
You should notice that an unwanted glowing highlight has appeared around the edge of the image.



To eliminate this glow, you should apply the blur to an image that has nothing added to the rim (against a black background)—since black has a value of 0 and therefore does not add to the filtering.

Adding a filter to an unpremultiplied image—the right way:

- To fundamentally change the compositing order, you should premultiply the foreground with the mask with a *SwitchMatte* node, then apply an *Over* node to create the composite. (*KeyMix* is only used for unpremultiplied images.)



The following image shows the same tree with a single premultiplied image. The preMultiply parameter is disabled in the *Over1* node (otherwise a black edge appears around the image).

Blur on a premultiplied image

Over1 detail

Because transforms do a bit of filtering, technically speaking, you should perform transforms on a premultiplied image as well, although the tolerances are a little more forgiving.

To Sum Up

Make sure your image is unpremultiplied before you do any color correction, by applying an *MDiv* to premultiplied images. Then, apply an *MMult* prior to applying transforms and filters. Finally, use any of the layering nodes to composite the images together.

Nodes That Affect Premultiplication

The following nodes can change the premultiplication status of an image, although anything that operates an image's mask differently from the RGB channels changes the effect.

Node	Description
Color–*ColorCorrect*	This color corrector has an option in its Misc tab to specify when the incoming image is premultiplied. If it is, set the premultiplied parameter to yes. *MDiv* and *MMult* are internally inserted into the computation.
Color–*Reorder*	This node allows you to switch a channel into the alpha channel, and may disrupt your premultiplication status. However, it isn't often used on images in the normal chain of color correct, position, and composite operations—but it is more of a utility node.
Filter–*DilateErode*	This node is typically used to add to or chew the matte by a few pixels. Set your channels to just "a." Because you then modify the matte separately from the RGB, you make the image unpremultiplied.
Key–*LumaKey, DepthKey, DepthSlice*	These keyers all have the option to immediately set your image to a premultiplied state.
Key–*KeyLight*	This keyer can output either a premultiplied or an unpremultiplied version, or just the alpha channel with the RGB untouched.
Key–*Primatte*	This keyer can output either just the alpha or a premultiplied version. Setting it to unpremultiplied requires an external *MDiv*.
Layer–*AddMix*	This modified *Over* node allows you to manually break the premultiplied relationship by tweaking curves that specify how the matte multiplies the foreground and background images. It was inherited from another package, where it was used extensively because this particular system had no control over premultiplication.
Layer–*SwitchMatte*	This node copies a channel from the second image to the alpha channel of the first image. You have the option to immediately premultiply the image (enabled by default).
Other–*DropShadow*	This node should be applied to premultiplied images only.

Non-Black Premultiplied Images

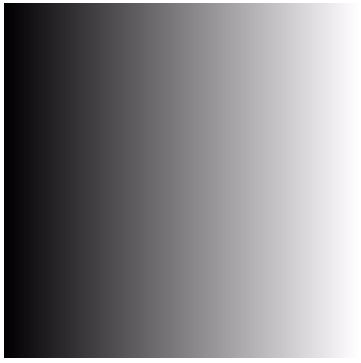
Some packages output images that are considered premultiplied, but in which the background is not black. This is mathematically bizarre. To work with these images, try the *AEPremult* macro included Chapter 32, "The Cookbook," on page 989.

The Logarithmic Cineon File

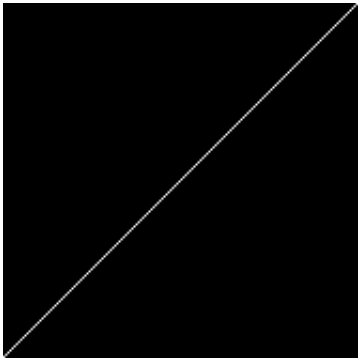
Kodak created the Cineon file format to support their line of scanners and recorders. Two things are typically associated with the Cineon file: the Cineon file itself (an uncompressed 10-bit image file) and the file's particular color representation. When graphed, this representation takes the form of a logarithmic curve, hence the term "logarithmic color" (or "log" for short). Although some refer to it as a "log space," it is not actually a "space" such as YUV, RGB, or HSV.

Note: (A caveat, if you will.) This section is to be treated only as an overview of a subject that causes flame wars of such staggering proportions that everybody walks away dazed and filled with fear and loathing. You have been warned.

The image set below represents linear space—every value has an equal mathematical distance in brightness to its neighbor.

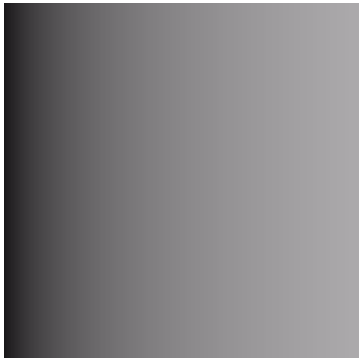


Linear image

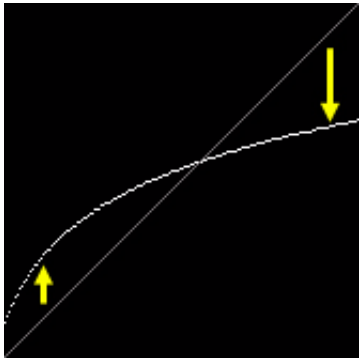


Graph

The following image set represents a ramp converted to logarithmic color, where the brightness is weighted.



Logarithmic image



Graph

The logarithmic image does not appear to have a pure black or white. Its graph shows that the highlights are flattened (compressed), and that the blacks have more attention. This is why Cineon frames typically seem to have a very low contrast, mostly in the highlights. Because the Y axis represents the output data, it contains less data than the X axis. You can therefore store this in a smaller file than you might otherwise use, to which the Cineon 10-bit format is nicely adapted as good balance between color range and speed.

Log compression is not inherent or unique to the Cineon file format. You can store logarithmic color data in any file type, and you can store linear color data into a *.cin* file.

The easiest way to think about a logarithmic file is as a "digital negative," a concept encouraged by Kodak literature. Unfortunately, Kodak decided to not actually invert the image. While the idea of a negative piece of film is easy to understand, this concept may cause some confusion. To see it properly, you must invert the film. In the same manner, you cannot work with a "digital negative" in your composite. You must first convert the image, in this case from the filmic log color to the digital linear color. This is called a *log to lin* color correction, and is performed with the Color–*LogLin* node.

It is important to remember that log to lin (or lin to log) conversion is simply a color correction. This workflow is explained below. However, it is first necessary to discuss how this color correction works.

A piece of film negative has up to approximately 13 stops of latitude in exposure. A stop is defined as a doubling of the brightness. The "digital negative," the Cineon file, contains approximately 11 stops. A positive print cannot display as much range as a negative. The same rule applies to a computer monitor—it can only display about 7 stops of latitude. To be properly handled by the computer, information must be discarded. In a simplified example, an image contains a rounded-off range of 0 to 11 stops for black to white. Since the monitor can only display about 7 stops (for example, steps 1 through 8), the rest of the image (below 1 and above 8) is thrown away. These 7 steps, only a portion of the original negative, are then stretched back to 0 (black) and 11 (white). Because what used to be dark gray to medium-bright gray is now black to white, the image appears to have a higher contrast. However, information has been thrown away in the process. This loss is permanent when working in 8 or 16 bits, but can be retrieved when working in 32-bit float. For more information, see "Logarithmic Color and Float Bit Depth" on page 444.

The extraction process is one way to control exposure. If you select the higher portions of the image (for example, 4 to 11 rather than 1 to 8), the image appears darker because you are remapping the brighter parts of the image down to black. This is the same as selecting to expose your camera for the brightest portion of your scene. The opposite occurs when selecting the lower portions of the image brightness.

These types of controls are paralleled in the *LogLin* node with the black and white point parameters. Every 90 points represents one stop of exposure. You can therefore control exposure with the *LogLin* node.

Note: Some Kodak documents state that 95 points represents one stop of exposure.

Once images are converted to linear color, they can be treated like other "normal" linear images. When a linear image is broken down to its steps of brightness, there is equal distance between two steps in the dark areas and two steps in the light areas. In a logarithmic file, however, the digital negative sees light areas differently than it sees dark areas, as described by the logarithmic curve that it is stored in. The distance between two steps in the dark areas is different than the distance between two steps in the bright areas. This is why color corrections and compositing should be done in linear color. This is explained in "The Hazards of Color Correcting in Logarithmic Color" on page 439.

Once you have finished your compositing, the images must be converted back to logarithmic representation (another *LogLin* node set to "lin to log") and then rendered to disk. These images are then properly treated by the film recorder.

A Little Further Reading

Two websites are recommended for more information on this subject. The first is the specification for the logarithmic conversion and all of the parameters. It is somewhat dense, but contains useful information:
http://www.cineon.com/conv_10to8bit.php

The second recommended site contains a nice discussion of the film negative's response to light:

http://www.slonet.org/~mhd/2photo/film/how.htm

The Hazards of Color Correcting in Logarithmic Color

If the logarithmic format is so great, why bother to convert back to the linear color? First, logarithmic color is unnatural to the eye—you have to convert back to linear color to see it properly. More importantly, compression also means that any color corrections applied in log color produce unpredictable results, since shadows, midtones, and highlights have an uneven application in many color correctors.

In the example below, the first image is the original plate in log color. The second image has been converted back to linear color, and therefore looks more natural to the eye.

Plate in log color

Plate in linear color

A Color–*Mult* node is applied to the log image and to the linear version with an extremely slight red color.

Note: You are invited to view the online PDF documentation to see the color images.

It is difficult to gauge the result in the first image, since it is still in log color. The second image lends a nice tint to the pink clouds (assuming you want pink clouds).

Mult in log color

Mult in linear color

When the log image is converted to the more natural linear space, the "slight" red multiplier applied before the conversion has completely blown the image into the red range. This is bad.

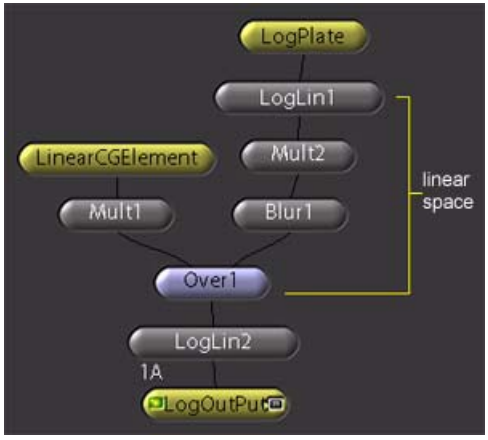
Mult in log color viewed in linear representation

Therefore, you are mathematically urged to color correct in linear color, or view a conversion to linear using a VLUT while you adjust the color correction.

Converting Between Logarithmic and Linear Color

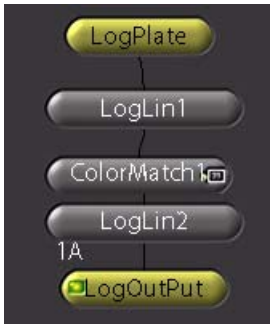
As previously mentioned, logarithmic images are simply a result of a color correction. This correction has been standardized by Kodak. Every conversion function is essentially the same. The Shake node that handles this correction is the *LogLin* node, and is located in the Color Tool tab. Using this node, you can convert from log to linear color, or linear to log color.

Typically, a *LogLin* node is applied after a *FileIn* node to convert the image from log to linear color. You do your composite in linear color, convert back to log at the very end, and attach the *FileOut* node. In the following node tree, *LogLin1* has a conversion setting of log to lin, and *LogLin2* has lin to log.



This example also includes an imaginary 3D-rendered element, read in by the *FileIn* node named *LinearCGElement*. These elements are almost always rendered in linear color, so no conversion is necessary.

If you are simply doing color timing, as in the following example, you have an added benefit: All the color corrections concatenate into one internal operation.



Wedging and Color Timing

So, why have numbers in the *LogLin* node? These numbers are used to color correct your plate as a sort of calibration of your entire input/output system. There are (at least) two fundamental ways to do this: *wedging* and *color timing*. The following two methods are only suggestions. Every facility probably has its own system—there is no standard.

When wedging a shot, you ensure that the files output from your entire digital process match the original film print. There are multiple places that color changes can intentionally or accidentally occur—exposing film in the camera, printing the workprint, scanning the data into digital format, digitally compositing, recording the digital plate onto film, and developing the print. You are saved on the one hand in that much of this process is out of your hands—the print lab guy was cranky because his delivered coffee was too cold, so what can you do? To limit discrepancies, use the *Wedge* macro in Chapter 32, "The Cookbook," on page 1000. This is a macro for *LogLin* that puts in preset variations of the offset values and contrast values, and steps up and down to try to find a match of the original scan.

The process usually involves the following steps:

1. Make a workprint of your original negative. This is the standard to which you compare your digital output.
2. Scan the negative into a digital format (which usually creates log Cineon files).
3. Create a *FileIn* node for a single frame from the scanned image files, then attach the *Wedge* macro.
4. Visually, take your best shot at the exposure level. (You may have to boost the blue up or down 22 points to see what may work.) Remember that 90 points is 1 f-stop of exposure and 45 points is half a stop.

5 Render out 48 frames. The *Wedge* macro automatically brackets your initial pick up and down by whatever value you set as the colorStep. For a wide bracket, use a high number (such as 90). For a narrow bracket, use a lower number (such as 22). This process prints 48 different color, brightness, and contrast tests, then automatically returns the image to log color with the default settings. The internals of the *Wedge* macro are basically *LogLin* (log to lin) > *ContrastRGB* > *LogLin* (lin to log).

6 Record and print your 48 frames.

7 Compare the actual physical pieces of film to the frame of the original workprint on a light box using a loupe. The exposure numbers are printed on the frame by the *Wedge* macro. Select the frame that exactly duplicates the original print. If no frames match up, adjust your starting points for red, green, and blue, narrow your colorStep, and wedge again until you have a frame that looks correct.

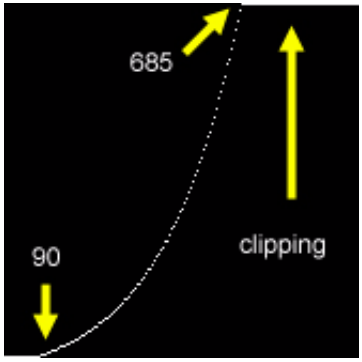
8 For all composites that use this plate, use the values you selected in the *Wedge* macro in your *LogLin* node, converting from log to linear color. Keep your default values when returning to log color, with the exception of the conversion setting, which is linear to log.

This process is generally done for every shot. At a larger studio, there is likely to be an entire department to do the color timing for you. Count yourself lucky to be in such a wise and far-sighted studio.

The second technique to handle the color process is to not handle it at all—let the Color Timer for the production or the developing lab handle it. While this is easier, you do surrender some control over the process. However, this is a perfectly acceptable technique used in many large effects houses. When you employ this technique, you use the same values in your *LogLin* in and out of your color. You may adjust the numbers slightly, but make sure the same numbers are used in both operators. For example, because a physical piece of negative has a slight orange cast, your positive scan may have a slight green cast. You may want to adjust for this in the *LogLin* node. A good technique is to adjust your log to lin *LogLin*, copy the node (Command-C or Control-C), and then paste a linked copy back in (Shift-Command-V or Shift-Control-V). Next, adjust the conversion setting of the second *LogLin* node to lin to log. If you adjust the original *LogLin* node, the copy takes the same values since it is a linked copy.

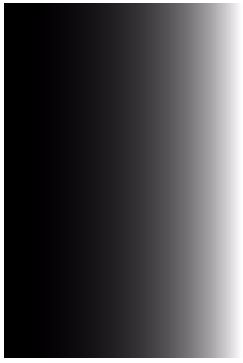
Logarithmic Color and Float Bit Depth

Here is where it gets tricky. If you examine the following logarithmic-to-linear conversion curve, you can see that clipping occurs above 685.

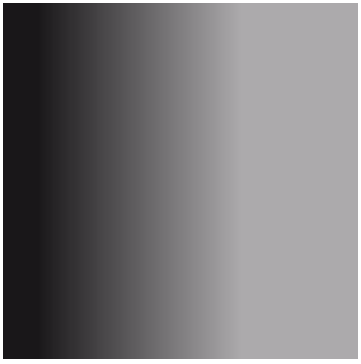


The *LogLin* node does have a roll-off operator, which helps alleviate the sharp corner at the 685 point, but this is inherently a compromise—you are throwing data away. You do not really see this data disposal in linear color. However, once you convert back to logarithmic, the color clipping is evident.

In the following grayscale examples, the left image is a ramp with an applied log to lin conversion. It is now in linear color. The right image is the left image converted back into log color representation. Quite a bit of clipping has occurred.



Log grayscale converted to linear color



Previous example converted back to log color

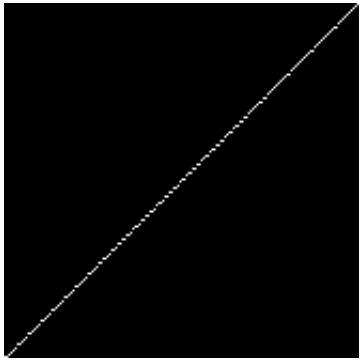
The following images are from a log plate. The left image is the original plate. The right image is the output plate, also in log color, that has been passed through the log-to-lin-to-log conversion process.

Original log image

Output log image

Notice that the highlights in the hair detail are lost.

The following images are graphs that represent the log plates in the above illustrations. The left graph represents the entire range of the log image. Keep in mind that this represents all of the potential values—few log plates have this entire range. The right image displays the result of the log-to-lin-to-log conversion process.

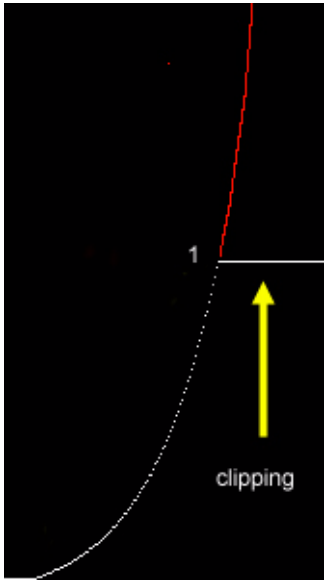


Graph of original ramp

Graph of output ramp

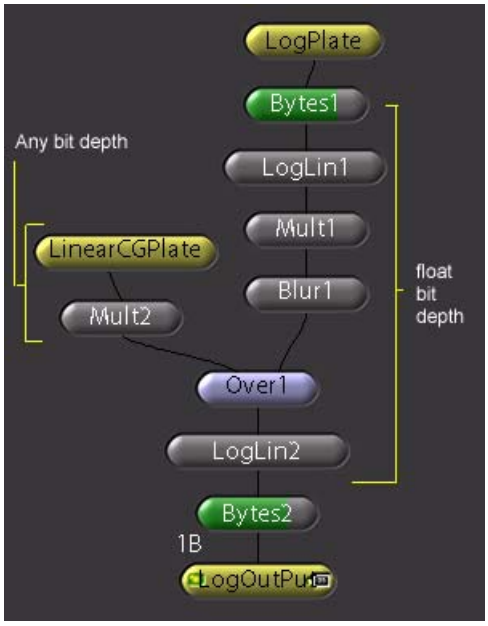
So, why is this happening? In the first part of this chapter ("The Logarithmic Cineon File" section), using an 11-stop example image, 7 stops were extracted and the rest thrown away in order to view and work on the image in the computer. These same 7 stops can be thought of as the stops between 95 and 685. As mentioned earlier, the rest are thrown away, or clipped.

If you look back at the original log-to-lin conversion graph, the curve suggests that it should continue past 1, but so far, the curve has been clipped at 1. In the following illustration, the red line represents the potential information derived from the color conversion. The curve is clipped at 1 because values can only be stored from 0 to 1 in 8 or 16 bits.



As the illustration suggests, if data could be preserved above 1, you could always access the data, and life would be happy. Fortunately (by design), you can convert your images to a higher bit depth called "float." Whereas 8 and 16 bits are always in a 0 to 1 range, float can describe any number and is ideal for working with logarithmic images when converting to linear representation and back to log representation. If you keep your images in linear color because you are reading out to a linear format, float is not necessary.

The following image shows a modification of the compositing tree shown on page 441. An Other–*Bytes* node is inserted, and the image is bumped up to 4 bytes (32 bits, or float).



Notice that you are not obligated to promote your 3D-rendered element (here represented with *LinearCGPlate*) to float, since it is already in linear color. The second *Bytes* node at the end of the node tree is included in case you render to an .iff format—you may want to convert it down to 16 bits for smaller storage costs. Cineon is always stored at 10 bits, and therefore does not need the second *Bytes* node.

The following is a table of file sizes for 2K plates. Draw your own conclusions.

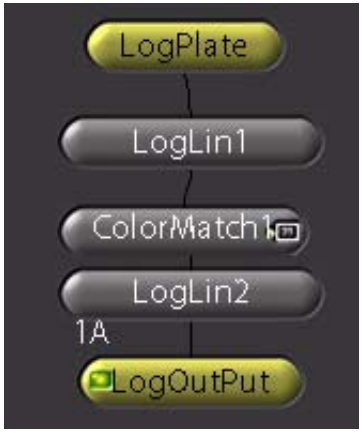
2K RGB Plate	Size
Cineon, 10 bits	12 MB
IFF, 8 bits	9 MB
IFF, 16 bits	18 MB
IFF, float	36 MB

In addition to storage requirements, working in float costs you render time, a *minimum* of 20 percent, but usually significantly higher than that.

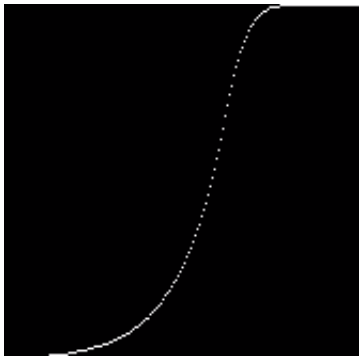
Looking at Float Values

You can use the Float View ViewerScript to view values that are above 1 or below 0. To look at these values, create a *Ramp* node, and set depth to float. Then, apply a Color–*LogLin* node. When you turn on the Float View, the top 35 percent turns white, indicating values above 1, and the lower 9 percent turns black, indicating values below 0. Everything else turns gray, indicating values between 0 and 1.

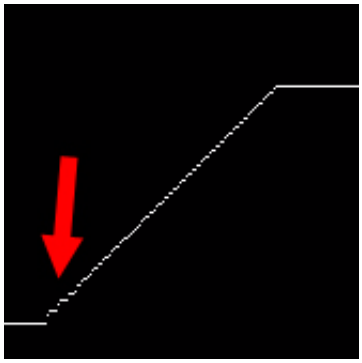
If you are just color timing (doing color corrections that concatenate), there is no need to convert to float. All concatenating color correctors, including the two correctors to convert in and out of linear representation, are calculated simultaneously. Therefore, the following tree is still accurate.



As an alternative to float, you can use the roll-off parameter in the *LogLin* node. However, this involves inherent compression and banding of your data. The roll-off parameter gives your curve a roll-off (compared to the standard log curves), which can help preserve some highlights, and allows you to stay in 16 bits. However, as shown in the next example, when the same image is converted back to log representation, there is still some cutoff in the upper and lower ranges, but the lower ranges also band. Use at your own risk.



Graph of linearization With roll-off



Graph of relog With roll-off

Float Bit Depth and Third-Party Plug-Ins

Most third-party plug-ins, including *Primatte* and *Keylight*, do not support float (rather, only 8 and 16 bits).

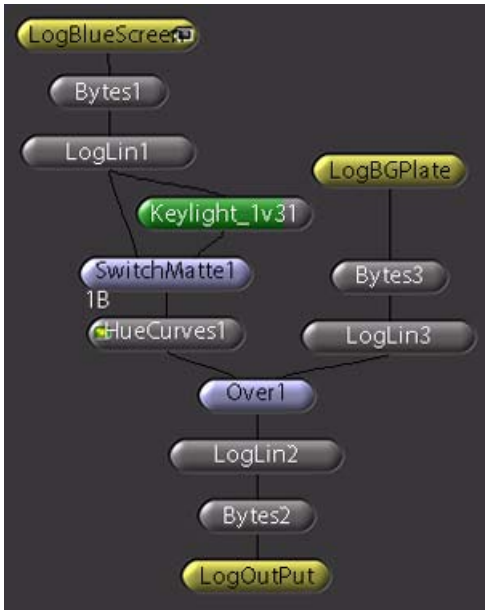
To ensure that your highlights are maintained:

1. Convert the images back to log color.
2. Execute the third-party plug-in (assuming it is still accurate on non-linear images).
3. Convert the images to linear representation.
4. Continue with your composite.

If the plug-in works on a single channel (for example, both *Primatte* and *Keylight* can isolate their effect on the alpha channel), do the following:

1. Create one branch for color modifications, and keep that branch in float.
2. Create a second branch from the *FileIn* node and pull the key.
3. Copy the alpha back to the float RGB chain with a *SwitchMatte* node.
4. Since the keyers also do color correction, you have to compensate by using a Color–*HueCurves* or Key–*SpillSuppress* node to do your spill suppression.

The following is a sample tree:



There are two exceptions:

- *Keylight* allows you to key, color correct, and composite in log color. Simply toggle the colourSpace control to log.
- *Ultimatte* preserves float data. No tricks necessary.

Applying gentle pressure to the plug-in manufacturer to support float images is also a nice alternative, but less productive in the short run.

Compositing With Layer Nodes

16

Layer nodes form the foundation for compositing two or more images together in Shake. This chapter covers the basic Shake compositing nodes—how they work, and how to use them.

Layering Node Essentials

The Shake compositing nodes are located in the Layer Tool tab.

There are three types of layering nodes:

- *Atomic nodes:* Atomic nodes (*Over*, *IAdd*, *Atop*, and so on) do one thing—combine two images according to a fixed mathematical algorithm. (Hence the term *atomic*: they apply a single, elemental operation to a pair of images.) They are useful for command-line compositing, scripting, and are also convenient for the Node View in that you can quickly see which type of operation is occurring.
- *More flexible nodes:* The second node type are the more flexible *MultiLayer, MultiPlane,* and *Select* nodes. *MultiLayer* allows you to duplicate most of the atomic nodes (with the exception of the *AddMix*, *AddText*, *Interlace,* and *KeyMix* nodes). These nodes have the additional benefit of allowing unlimited inputs to the node.
- *LayerX node:* The third category is the unique node *LayerX*, which allows you to enter your own compositing math.

Before getting into the layer nodes in more detail, here are some important rules to keep in mind when compositing in Shake.

Don't Mask Layer Nodes

The side-input masks for layer nodes should not be used, as they behave counter-intuitively and will not produce the result you might expect. If you want to mask a layering node, mask the input nodes, or use the *KeyMix* node.

Remember the Rules of Premultiplication

There are two unbreakable rules that you must *always* follow when creating a composition in Shake:

- *Rule Number 1:* Only color correct unpremultiplied images. To unpremultiply an image, use the Color–*MDiv* node.
- *Rule Number 2:* Only apply *Filter* and *Transform* nodes to premultiplied images. To premultiply an image, use the Color–*MMult* node.

For more detailed information on premultiplication, see "About Premultiplication and Compositing" on page 421.

Using the clipMode Parameter of Layer Nodes

You can easily composite elements of any resolution. To set the output resolution of a composite that contains images of multiple resolutions, go to the compositing node's parameters and use the clipMode button to toggle between foreground or background (as the output resolution). This applies to all layering commands. An element composited over a differently sized background is one way to set your output resolution. For more information on setting resolution, see Chapter 3, "Adding Media, Retiming, and Remastering," on page 107.

Compositing Math Overview

If Shake had only one layer node, it would have to be *LayerX*, since it can be used to mimic all of the other compositing nodes. The math for most of the operators is included in this node, both in general notation and in *LayerX* syntax. The *LayerX* syntax has expressions for each channel.

The following table provides a quick reference to the Shake layer nodes and their common uses, math, and *LayerX* syntax. For specific node descriptions, see "The Layer Nodes" on page 453.

Layer Node	Common Uses	Math	LayerX Syntax
Atop	Add effects to foreground elements, like smoke over a CG character to match the background.	`A*Ba+(B*(1-Aa))`	`r2+(r*a*a2)` or `(a2==0 || (r2==0 && g2==0 && b2==0 && a2==0) ? r2 : (a2*r+(1-a2*a)*r2))`
Common	Create difference masks.		
IAdd	Add fire effects, adding mattes together.	`A+B`	`r+r2`
IDiv		`A/B`	`r2==0?1:r/r2`
IMult	Mask elements.	`A*B`	`r*r2`
Inside	Mask elements.	`A*Ba`	`r*a2`
Interlace	Interlace two images, pulling one field from one image, and the second field from the other image.		

Layer Node	Common Uses	Math	LayerX Syntax
ISub		`A-B`	`r-r2`
ISubA	Find the difference between elements.	`absolute(A-B)`	`fabs(r-r2)`
KeyMix	Mix two images through a third mask.	`A*(1-M*C)+ (B*M*C)`	M represents the percentage mix.
Max	Combine masks.	`If (A > B) then A, else B`	`r>r2?r:r2 or max(r,r2)`
Min		`If (A < B) then A, else B`	`r<r2?r:r2 or min(r,r2)`
Mix	Cross-fade.	`A*(1-Mix)+(B*Mix)`	`r*(1-mix)+r2*mix`
Outside	Mask elements.	`A*(1-Ba)`	`r*(1-a2)`
Over	Primary compositor	`A + (B*(1-Aa))`	`r+(1-a)*r2`
Screen	Add reflections and light elements without losing highlight detail.	`1-(1-A)*(1-B)`	`1-(1-r)*(1-r2)`
Under	A reverse of *Over*	`B + (A*(1-Ba))`	`r2+r*(1-a2)`
Xor	Find areas that are mutually exclusive.	`A*(1-Ba) + B*(1-Aa)`	`r*(1-a2) + r2*(1-a)`

ZCompose	Create Z-depth composite.
	
	If foreground alpha does not equal 1, an *Over* is performed. Otherwise, if the Z values are not equal, the lower Z is taken. If the Z values are the same, the result is a 50 percent mix of an *Over* and an *Under*.
		`If(Aa == 1) then`
		`  If (Az == Bz) then`
		`((A + (B*(1-Aa))) +`
		`B + (A*(1-Ba)) ) / 2`
		`  else`
		`If(Az < Bz) then A`
		`  else B`
		`else A + (B*(1-Aa))`	

The Layer Nodes

This section provides a detailed description of each of the layer nodes.

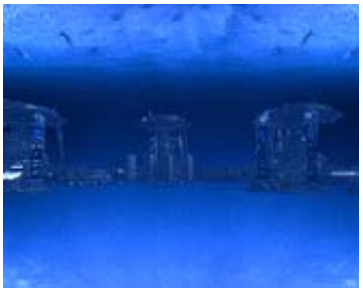
AddMix

The *AddMix* node is similar to the *Over* node, except that you have control over curves to help blend the edges together. For more information on the *Over* node, see "Over" on page 466.

AddMix Example

In the following node tree, a key is pulled from the foreground using a *KeyLight* node. The foreground is then attached to the background tree. The first "Closeup" illustration (on the left) shows the default result, a result identical to an *Over* node. Ringing problems occur along the hair (a bright ring), and along the shoulder (a dark ring). When the curves are adjusted, the ringing is dramatically reduced, resulting in a much cleaner composite.

Node tree



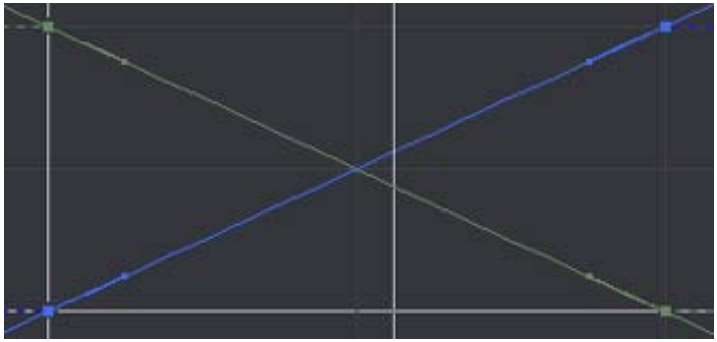
Over1

Bluescreen

AddMix1

Closeup, default curves

Closeup, modified curves



Default curves

Modified curves

Note: Because the curves can dramatically alter opacity of the foreground, you may need to ensure that the interior of the matte is 100-percent opaque through the use of additional masking, garbage mattes, and so on.

Parameters

This node displays the following controls in the Parameters tab:

clipMode

Toggles between the foreground (0) or the background (1) image to set the output resolution.

preMultiplied

Set this parameter to Yes if the foreground is premultiplied. This is the opposite of the standard *Over*. If enabled, the foreground curve does nothing.

foreground curve

A graphical control appearing within a curve editor in the Parameters tab. This curve controls the way the edge of the foreground image is blended.

background curve
A graphical control appearing within a curve editor in the Parameters tab. This curve controls the way the edge of the background image is blended.

AddText

The *AddText* node differs from the *Text* image node only in that it places text over a pre-existing background. See "Text" on page 604 for more information on this extremely flexible feature. *AddText* is also a bit nicer than *Text* for the command line because you do not have to actually enter all of the parameters for the image width and height.

Parameters

This node displays the following controls in the Parameters tab:

text
The text you want to display in your composite. This defaults to the word *Text*. This parameter correctly interprets ASCII codes preceded by a backslash. For example, entering `\12` into the text field results in a line feed that begins a new line of text.

font
The font you want to use. You can use any TrueType or PostScript font. The default font is Utopia Regular.

xFontScale
The horizontal size of the font. The default is 100 pixels.

yFontScale
The vertical size of the font. By default, this parameter is set to the *xFontScale* parameter, locking both of these parameters together to maintain the aspect ratio of the selected font.

leading
The space between multiple lines of text. The default is1.

xPos
The horizontal position of the text. By default, this parameter is set to *width/2*, which is the center of the frame.

yPos
The vertical position of the text. By default, this parameter is set to *height/2*, which is the center of the frame.

zPos
The Z depth of the text. The default is 0.

xAlign
The horizontal alignment of the text. The default is Centered (2).
- 0 = left/top-justified
- 1 = right/bottom-justified
- 2 = centered

yAlign
The vertical alignment of the text. The default is Centered (2).
- 0 = left/top-justified
- 1 = right/bottom-justified
- 2 = centered

Color
A color control defining the color of the text. The default is white (1,1,1).

alpha
The transparency of the text. The default is 1 (solid).

xAngle
Rotates the text about a horizontal axis, so that it appears to tilt toward or away in 3D space.

yAngle
Rotates the text about a vertical axis, so that it appears to tilt toward or away in 3D space.

zAngle
Rotates the text around its center, so that it appears to spin around in the Viewer.

fieldOfView
Degrees of view, which is half the total vertical field of view, from the perpendicular out to the edge. The default is 45.

kerning
The spacing between each letter. This parameter can be set to negative values.

fontQuality
The polygonalization factor of the font splines. This parameter is conservatively set to a high value. For flat artwork, you can probably get away with a value of 0. When you have extreme perspective, you want to keep the value high.

Atop

The *Atop* node is similar to *Over*, except that the background matte is used; the foreground only appears where there is background matte. Unlike the *Inside* node, *Atop* also keeps the background information. In other words, *Atop* is the equivalent of applying an *Inside* node followed by an *Over* node. Use *Atop* for applying atmosphere effects to a foreground element, such as compositing smoke over a foreground character.

Parameters

This node displays the following control in the Parameters tab:

clipMode
Toggles between the foreground (0) or the background (1) image to set the output resolution.

Common

The *Common* node compares two images and extracts or hides common elements. Use *Common* to create difference mattes. The extracted elements are taken if the difference between the two images is less than the set tolerances. A similar node is *ISubA*, which subtracts two images and returns the absolute value of these images.

Parameters

This node displays the following controls in the Parameters tab:

clipMode
Toggles between the foreground (0) or the background (1) image to set the output resolution.

mode
Tells Shake what to do with common elements.

- *0:* "show" keeps common elements only, turning the rest black.
- *1:* "hide" hides common elements, leaving different elements untouched.
- *2:* "white" turns the image to black and white, keeping common elements white.
- *3:* "black" is the inverse of "white."
- *4:* "proportion" takes the difference between the values, averages the values, and then inverts the values.

rTol
The red color channel tolerance. If pixels between the two images are less than the tolerance value, they are considered common.

gTol
The green color channel tolerance. If pixels between the two images are less than the tolerance value, they are considered common.

bTol
The blue color channel tolerance. If pixels between the two images are less than the tolerance value, they are considered common.

aTol
The alpha channel tolerance. If pixels between the two images are less than the tolerance value, they are considered common.

Constraint

Constraint is a multifunctional node that restricts the effect of nodes to limited areas, channels, tolerances, or fields. Toggle the type buttons to select the constraint type. When you do so, certain parameters then become active; others no longer have any effect. This is similar to the *KeyMix* node in that you mix two images according to a third constraint. *KeyMix* expects an image to be the constraint. *Constraint* allows you to set other types of constraints.

The *Constraint* node also speeds calculation times considerably in many cases. The speed increase always occurs when using the ROI (Region of Interest) or field mode, and for many functions when using channel mode. Channel mode decreases calculation time when the output is a result of examining channels, such as layer operations. Calculation time is not decreased, however, when it must examine pixels, such as warps and many filters. The tolerance mode may in fact increase calculation times, as it must resolve both input images to calculate the difference between the images.

Parameters

This node displays the following controls in the Parameters tab:

clipMode

Toggles between the foreground (0) or the background (1) image to set the output resolution.

type

Selects the type of constraint you use.

- *AOI - Area of Interest (1):* Draws a mixing box.
- *Threshold (2):* Only changes within a tolerance are passed on.
- *Channel (4):* Only specific channels are modified.
- *Field (8):* Only a selected field is modified.

Because of the labeling, you can do multiple types of constraining in the script by adding the numbers together. For example, 7 = AOI (1) + Threshold (2) + Channel (4); in other words, AOI, Threshold, and Channel are all active.

AOI Controls

These controls are active only if the *type* parameter is set to AOI. Opening the AOI controls subtree reveals left, right, bottom, and top subparameters, which create a cropping box for the effect.

rTol

Controls the red color channel tolerance if the type parameter is set to Threshold (2). If pixels between the two images are less than the tolerance value, they are considered common.

gTol
Controls the green color channel tolerance if the type parameter is set to Threshold (2). If pixels between the two images are less than the tolerance value, they are considered common.

bTol
Controls the blue color channel tolerance if the type parameter is set to Threshold (2). If pixels between the two images are less than the tolerance value, they are considered common.

aTol
Controls the alpha channel tolerance if the type parameter is set to Threshold (2). If pixels between the two images are less than the tolerance value, they are considered common.

thresholdType
Active only when the type parameter is set to Threshold (2). This sets the tolerance to "lo" or "hi."

- 0 = lo. Changes are made only if the difference between image1 and image2 is less than the tolerance values you set.
- 1 = hi. Changes are made only if the difference between FG and BG is greater than the tolerance values.

tReplace
Active when the type parameter is set to Threshold (2). Toggles whether the entire pixel is replaced, or just the channel meeting the Tolerance criteria.

Channel and Field Controls
Opening this parameter subtree reveals two subparameters.

channels: If the type parameter is set to Channel (4), the operation applies only to the specified channels.

field: If the type parameter is set to Field (8), the effect applies only to one field.

- 0 = even field
- 1 = odd field

invert
Inverts the selection. For example, everything beyond the specified color tolerance is included, rather than everything below the specified color tolerance.

Copy

The *Copy* node copies selected channels from BG to FG (replacing the images entirely). It is common to copy over the alpha channel. To copy channels within the same image, use the *Reorder* node. *SwitchMatte* depends on *Copy*, but is a macro with *MMult* and *Invert* placed inside of the node.

Parameters

This node displays the following controls in the Parameters tab:

clipMode
Toggles between the foreground (0) or the background (1) image to set the output resolution.

channels
Tells Shake which channels to copy from BG to FG. You can use r, g, b, a, and/or z. To use multiple channels, list them out. For example, *rgz* copies the red, green, and Z channels.

copyBData
If you're using an image with metadata within the blind data container, you can assign the blind header data from the foreground input image to the background input image by turning on copyBData. This can be useful if you have a complex node tree that uses many layer nodes combining several images, yet you want the final file to render out with specific header data taken from one of the *FileIn* nodes.

For more information about custom file header metadata, see "Support for Custom File Header Metadata" on page 178.

IAdd

The *IAdd* node adds one image to another. The strength of the second image can be raised and lowered with the percent parameter.

Parameters

This node displays the following controls in the Parameters tab:

clipMode
Toggles between the foreground (0) or the background (1) image to set the output resolution.

percent
A built-in fade parameter. This is the amount of the second image that is taken into consideration. Full strength is 100. If set to 50, the added amount is reduced by 50 percent.

IDiv

The *IDiv* node divides one input image by another input image. The strength of the second image can be raised and lowered with the percent parameter. To divide an image by its own matte channel, use the *MDiv* node.

Parameters

This node displays the following controls in the Parameters tab:

clipMode
Toggles between the foreground (0) or the background (1) image to set the output resolution.

percent
A built-in fade parameter. This is the amount of the second image that is taken into consideration. Full strength is 100. If set to 50, the added amount is reduced by 50 percent.

ignoreZero
Tells Shake to ignore black (zero) values.

IMult

The *IMult* node multiplies one input image by another input image. The second image's strength can be raised and lowered with the percent parameter. To multiply an image by its own matte channel, use *MMult*.

Parameters

This node displays the following controls in the Parameters tab:

clipMode
Toggles between the foreground (0) or the background (1) image to set the output resolution.

percent
A built-in fade parameter. This is the amount of the second image that is taken into consideration. Full strength is 100. If set to 50, the added amount is reduced by 50 percent.

ignoreZero
Tells Shake to ignore black (zero) values.

Inside

The *Inside* node places one image inside only the mask of a second image, so that only the mask of the second image is considered in the composite; the rest comes from the color of the foreground image. This is an extremely useful node for masking images.

Parameters

This node displays the following control in the Parameters tab:

clipMode
Toggles between the foreground (0) or the background (1) image to set the output resolution.

Interlace

This node interlaces two images. You can control field dominance, whether the input images are themselves separate field images (for example, half Y resolution), or if the fields are extracted from every other line. For information on the *Interlace* node, see "Interlace" on page 205.

ISub

The *ISub* image math node subtracts one image from another. The strength of the second image can be raised and lowered with the percent parameter. If you want to detect the difference between two images, use *ISubA* instead, because it takes the absolute value of the difference.

Parameters

This node displays the following controls in the Parameters tab:

clipMode

Toggles between the foreground (0) or the background (1) image to set the output resolution.

percent

A built-in fade parameter. This is the amount of the second image that is taken into consideration. Full strength is 100. If set to 50, the added amount is reduced by 50 percent.

ISubA

The *ISubA* node is identical to *ISub*, except that it takes the absolute value of the result. This is to see if similar images are different by subtracting one image from another. If the second image is brighter, *ISub* does not reveal anything, but *ISubA* does (for example, .5 - .8 = -.3, absolute value of -.3 = .3).

Parameters

This node displays the following controls in the Parameters tab:

clipMode

Toggles between the foreground (0) or the background (1) image to set the output resolution.

percent

A built-in fade parameter. This is the amount of the second image that is taken into consideration. Full strength is 100. If set to 50, the added amount is reduced by 50 percent.

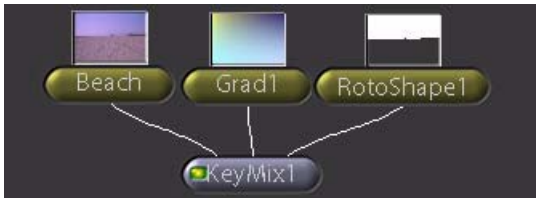
KeyMix

The *KeyMix* node mixes two images together through the specified channel of a third image. You can control the mix percentage, and also invert the masking image. *KeyMix* and *Over* are the two primary compositing nodes—*KeyMix* for unpremultiplied images and *Over* for premultiplied images.

For more information on premultiplication, see "About Premultiplication and Compositing" on page 421.

Example

In the following node tree, by mixing through the *RotoShape* node, you can color correct the beach and position the sky separately.



Grad1

RotoShape1

Beach

KeyMix1

Parameters

This node displays the following controls in the Parameters tab:

clipMode

Toggles between the foreground (0) or the background (1) image to set the output resolution.

channel
Lets you pick the channel from the third image that you want to use as the mask. You can pick the r, g, b, or a channel.

mixPercent
The percentage of the second image that's mixed in.

invert
Inverts the mask created from the third image.

LayerX

The *LayerX* node is exactly like *ColorX*, except that you also have access to a second image. This allows you to composite with expressions. The values of the second image are accessed with the variables *r2*, *g2*, *b2*, *a2*, and *z2*. For more information, see "ColorX" on page 647.

Parameters

This node displays the following control in the Parameters tab:

r, g, b, a, and zExpr
Each parameter allows you to enter a separate expression specific to that channel. By default, the parameters default to r, g, b, a, and z.

Max

The *Max* node compares two images and takes the pixel with the higher value. This is a good tool to merge alpha masks.

Parameters

This node displays the following controls in the Parameters tab:

clipMode
Toggles between the foreground (0) or the background (1) image to set the output resolution.

percent
A gain on the second image, which is effectively the same as a brightness node applied to the second image.

Min

The *Min* node compares two images and takes the pixel with the lower value. Like the *Max* node, this is a good tool to merge alpha masks.

Parameters

This node displays the following controls in the Parameters tab:

clipMode
Toggles between the foreground (0) or the background (1) image to set the output resolution.

percent
Adjusts the gain on the second image.

Mix

The *Mix* node mixes two images together according to a percentage. Animate the mixPercent parameter to create a dissolve between two clips.

Parameters

This node displays the following controls in the Parameters tab:

clipMode
Toggles between the foreground (0) or the background (1) image to set the output resolution.

percent
The percentage of each image used in the final result. The default is 50.

- 0 = nothing of background
- 100 = nothing of foreground

channels
Which channels are used in the final result. The default is "rgba."

MultiLayer

For complete information about using the MultiLayer node, see "Using the MultiLayer Node" on page 478.

Outside

The *Outside* node places only one image outside of the mask of a second image. Therefore, only the mask of the second image is considered in the composite; the color comes from the foreground image. This is a great tool for mask layers.

Parameters

This node displays the following control in the Parameters tab:

clipMode
Toggles between the foreground (0) or the background (1) image to set the output resolution.

Over

The *Over* node is the main compositing node of Shake. This node places one image over another, according to the matte of the foreground image. Images are assumed to be premultiplied. You can use an *MMult* on an input node if it is not premultiplied, or you can toggle the preMultiply parameter to 1.

Parameters

This node displays the following controls in the Parameters tab:

clipMode
Toggles between the foreground (0) or the background (1) image to set the output resolution.

preMultiply
When this parameter is on (1), the foreground image is multiplied by its alpha mask. If it is off (0), the foreground image is assumed to already be premultiplied.

addMattes
When enabled (1), the mattes are added together for the composite.

Screen

The *Screen* node mimics the effect of exposing two film negatives together. Technically, it inverts both layers, multiplies the two layers together, and inverts the result. It is particularly handy for reflections and luminescent elements, as it preserves the highlights.

Parameters

This node displays the following controls in the Parameters tab:

clipMode
Toggles between the foreground (0) or the background (1) image to set the output resolution.

SwitchMatte

The *SwitchMatte* node is a *Copy* command dedicated to copying a channel from the second image into the matte channel of the first image. You can choose the source channel, whether the channel is inverted, and whether the result is premultiplied.

Parameters

This node displays the following controls in the Parameters tab:

clipMode
Toggles between the foreground (0) or the background (1) image to set the output resolution.

matteChannel
Tells Shake which channel to copy from the background to the foreground alpha channel.

matteMult
Premultiplies the output image by multiplying the first input image by the alpha channel of the second input image.

- 0 = not premultiplied
- 1 = premultiplied

invertMatte
Inverts the result matte.

- 0 = do not invert the matte
- 1 = invert the matte

Under

The *Under* node is the same as the *Over* operation, except that it reverses the two input images. This is largely trivial for the script and the interface, but is extremely handy when working in command-line mode, since you can easily work on a background image.

Parameters

This node displays the following controls in the Parameters tab:

clipMode
Toggles between the foreground (0) or the background (1) image to set the output resolution.

preMultiply
When this parameter is on (1), the foreground image is multiplied by its alpha mask. If it is off (0), it is assumed that the foreground image is already premultiplied.

addMattes
When enabled (1), the mattes are added together for the composite.

Xor

The *Xor* node is somewhat like a combination of the *Inside* and *Outside* nodes. It is used to remove overlapping mask areas.

Parameters

This node displays the following controls in the Parameters tab:

clipMode
Toggles between the foreground (0) or the background (1) image to set the output resolution.

useMatte
Normally, the calculation performed by this node is based on comparing the RGB values in the foreground image to the RGB values in the background image. Turning useMatte on instead compares the alpha channels of the input images, but this effect also appears in the RGB channels.

ZCompose

The *ZCompose* node composes the input image over the background image using the Z values of both images to determine which image's pixel is placed on top. You should note that the *ZCompose* node uses an *Over* operation to compose the foreground and background input images after it has determined which should be in front, and which should be behind.

Z values are depth measurements from the camera to each object. A premultiplied image is assumed since Z values are normally produced by a render. With a *ZCompose* node, one of the image's pixels is either on top or not. This is sometimes a problem if you have subpixel objects. You might have ten dust particles in a single pixel, but if only one is in front of the corresponding object in the other image, then the pixel containing all ten particles is placed in front of that object. The problem is that the pixel does not contain the subpixel objects, just a representative color. For similar reasons, object edges can also be a problem.

For a tutorial on Z-compositing, see Tutorial 3, "Depth Compositing," in the *Shake 4 Tutorials*.

Note: The MayaZ Depth macro (see page 996) puts elements that have been rendered in Maya into a positive "normal" space; that is, using values such as 0 to 100, rather than using Maya's default -1/z formula.

Parameters

This node displays the following controls in the Parameters tab:

clipMode
Toggles between the foreground (0) or the background (1) image to set the output resolution.

invert
Turning on invert reverses the logic of the *ZCompose* node—0 becomes the most distant value, and higher values are considered progressively closer.

tolerance
The tolerance parameter determines how different, numerically, a value must be from another value to be considered equal. With the default tolerance of 0, comparing two input images with Z channel values of 40 and 50 will result in the image with 40 being promoted. With a tolerance of 10, both images are considered to have the same depth.

Other Compositing Functions

Shake contains other useful compositing nodes, located in the Other Tool tab.

Note: The *DepthKey* and *DepthSlice* node modify the alpha channel of an image based on Z-depth values. For more information, see Chapter 24, "Keying," on page 681.

AddShadow

The *AddShadow* node takes the alpha channel of an image, and then colors, blurs, fades, and moves the alpha channel. Next, the alpha is composited under the input, creating a basic shadow effect. It differs from *DropShadow* in that it already composites the shadow under the element; *DropShadow* creates an entirely new shadow element that can be manipulated separately.

Parameters

This node displays the following controls in the Parameters tab:

xOffset
Controls the shadow's horizontal offset from its originating layer.

yOffset
Controls the shadow's vertical offset from its originating layer.

fuzziness
Blurs the shadow. The slider's range is from 0 to 100, but this parameter can be set to any positive value.

Shadow Color
A color control that lets you define the color of the drop shadow.

opacity
Defines how transparent the drop shadow is. The slider's range is from 0 to 1.

DropShadow

The *DropShadow* node is similar to the *AddShadow* node, except that it eliminates the original image that the shadow is derived from, giving you an isolated shadow element that can be manipulated independently of the original source.

If necessary, you can recombine the resulting image with the original using one of the layer nodes.

Parameters

This node displays the following controls in the Parameters tab:

fuzziness
Blurs the shadow. The slider's range is from 0 to 100, but this parameter can be set to any positive value.

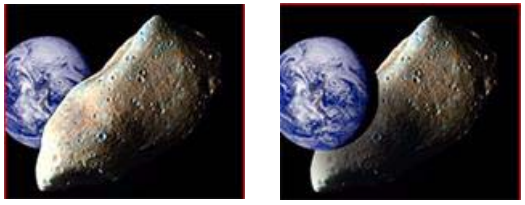
Shadow Color
A color control that lets you define the color of the drop shadow.

opacity
Defines how transparent the drop shadow is. The slider's range is from 0 to 1.

Select

The *Select* node, located in the Other Tool tab, allows you to select between multiple inputs. This node can be useful when you want to switch between different types of compositing logic over time, or when you have "doubles" that you want to insert. For example, you can switch a bluescreen character with a CG replacement using the *Select* node.

In the following simple example, you can specify whether the asteroid is in front or behind the Earth by putting two different *Over* operations into *Select* and toggling between the two operations.



Select has an infinite amount of potential inputs. Each node you connect to a *Select* node is sequentially numbered, starting at 1, from left to right.

Parameters

This node displays the following control in the Parameters tab:

branch
Lets you select the numbered input you want to use as the output image.

Layered Photoshop Files and the MultiLayer Node

17

Shake supports the use of layered Photoshop files with the MultiLayer node. This node also provides an easy way to composite three or more images within a single node, simplifying your node tree.

About the MultiLayer Node

The *MultiLayer* node duplicates the functionality of many of the atomic nodes covered in the previous chapter (with the exception of the *AddMix*, *AddText*, *Interlace,* and *KeyMix* nodes). Similarly to the *MultiPlane* and *Select* nodes, it has the additional benefit of allowing unlimited inputs into the node, and in fact acts as a miniature compositing environment inside of Shake.

The *MultiPlane* node also allows you to automatically import layered Photoshop files directly into Shake, retaining information about each layer's transparency, transfer modes, layer order, and layer visibility.

Importing Photoshop Files

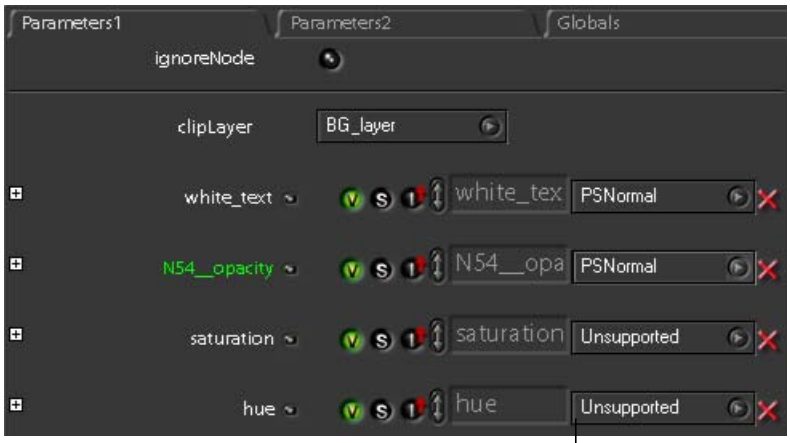
You can import a multilayer Photoshop file into Shake with a *FileIn* node or using the File menu. Shake supports most of the Photoshop transfer modes, with the exception of Color, Hue, Saturation, Dissolve, and Luminosity.

To import a layered Photoshop file using the File menu:

1 Choose File > Import Photoshop File.

The file is imported as a script—each layer in the Photoshop file is imported as a *FileIn* node that is then fed into a *MultiLayer* node. The *MultiLayer* node is named *Composite* by default.

2 Double-click the *Composite* node to display the Photoshop image (the composite) in the Viewer and load the *MultiLayer* node parameters.



Unsupported Photoshop Features and Issues

When you import a Photoshop file that contains one or more layers that are set to an unsupported transfer mode, the "Unsupported Transfer Mode (Mode Name)" message appears in the Viewer, and the mode for that layer in the Parameters tab reads "Unsupported."

Shake has no support for the following Photoshop features:

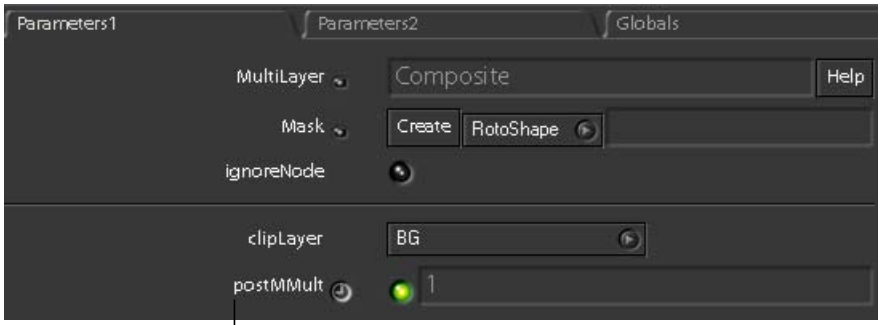
- Photoshop layer masks
- Alpha channels created in the Photoshop Channels tab
- Layer styles
- Text layers
- Fill layers

It's important to name the layers in a Photoshop file carefully. Observe the following restrictions:

- Never name a layer "color," "time," or "track."
- Never use a name that's identical to a C++ keyword.
- Never use a name that's identical to a C++ function call, for example "sin" or "rnd."

The postMMult Parameter

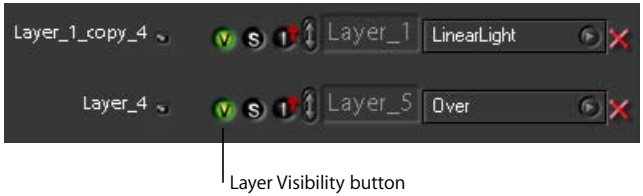
In the *MultiLayer* parameters, postMMult is enabled by default. When turned on, the postMMult parameter premultiplies the output image produced by the *MultiLayer* node (in the same manner as a composite in the Photoshop application).



For more information on the other buttons in the *MultiLayer* parameters, see "Using the MultiLayer Node" on page 478.

Photoshop Layer Visibility

When you import a Photoshop file that contains invisible layers (the eye icon is disabled in Photoshop), the visibility for that layer (the *FileIn* node) is disabled in Shake. To show the layer, toggle the layer's visibility button in the *MultiLayer* node parameters.



Photoshop Layer Opacity

To change the opacity of a layer, expand the subtree for the layer and set the opacity parameter to a value between 0 and 1. By default, the layer opacity is set to 1. A layer that is set to a Photoshop transfer mode contains an additional opacity control—the PSOpacity parameter.

This additional opacity control is necessary because Photoshop and Shake handle transparency differently. Photoshop's transparency setting works by varying the intensity of the alpha channel on the foreground image (prior to the blending operation). In Shake, the opacity setting on each layer of the *MultiLayer* node varies the intensity of all channels (RGBA). The results can differ between the two methods, depending on the selected blend mode and the way it uses color. The default PSOpacity setting is the opacity of the layer as set in Photoshop.

Note: To view the name of the Photoshop file, click the right side of a *FileIn* (Photoshop layer) node to display the *FileIn* parameters. In the Source tab, the name of the imported Photoshop file is displayed in the imageName parameter.

Supported Photoshop Transfer Modes

The layer modes listed in the following table are accessed within the *MultiLayer* node.

Note: The true math of the Photoshop transfer modes is the proprietary information of Adobe Systems Incorporated. As a result, the descriptions listed are not guaranteed to be technically accurate.

Layer Function	Description
ColorBurn	Takes the color of the second image and darkens the first image by increasing contrast. White in the second image has no effect on the first image.
ColorDodge	The opposite of ColorBurn, this lightens the image. Black in the second image does not affect the first image.
Darken	Identical to Min, taking the lower pixel value.

Layer Function	Description
Exclusion	The second image acts as a mask for inverting the first image. White areas of the second image completely invert the first image; black areas of the second image leave the first image unmodified.
HardLight	Screens or multiplies the first image, depending on the strength of the second image. Values below 50 percent in the second image multiply the first image, making it darker. Values above 50 percent in the second image act as a Screen effect. Values of pure black or white in image 2 replace image 1.
Lighten	Identical to Max. Takes the maximum value when comparing two pixels. Good for adding mattes together.
LinearBurn	Similar to ColorBurn, except it decreases brightness, not contrast, to darken the first image. Whites in the second image do not modify the first image.
LinearDodge	The opposite of LinearBurn, brightens the first image. Black areas in image 2 leave the first image unaffected.
LinearLight	A combination of LinearBurn and LinearDodge. Values above 50 percent in image 2 increase brightness of the first image. Values below 50 percent darken the first image. Values of pure black or white in image 2 replace image 1.
Overlay	To shade an image. 50 percent in the second image keeps the first image the same. Brighter pixels brighten the first image, darker pixels darken it.
PinLight	Performs Min or Max, depending on the strength of the second image. If the second image is brighter than 50 percent, a Max (Lighten) is performed. If the second image is darker than 50 percent, a Min (Darken) is performed. Values of pure black or white in image 2 replace image 1.
SoftLight	Raises or lowers brightness, depending on the strength of the second image. Values above 50 percent in the second image decrease the brightness of the first image; values below 50 percent increase the brightness.
VividLight	Raises or lowers contrast, depending on the strength of the second image. Values above 50 percent in the second image decrease the contrast of the first image; values below 50 percent increase the contrast.

Importing a Photoshop File Using the FileIn Node

If you don't want to import every layer within a Photoshop file, you can simply use a *FileIn* node, as you would with any other media in your script. When you do this, you have the option of reading any single layer from that file into Shake.

To import a Photoshop file using a *FileIn* node:

- In the Image tab, click *FileIn* and navigate to the Photoshop file in the File Browser. Select the file (or files), then click OK.

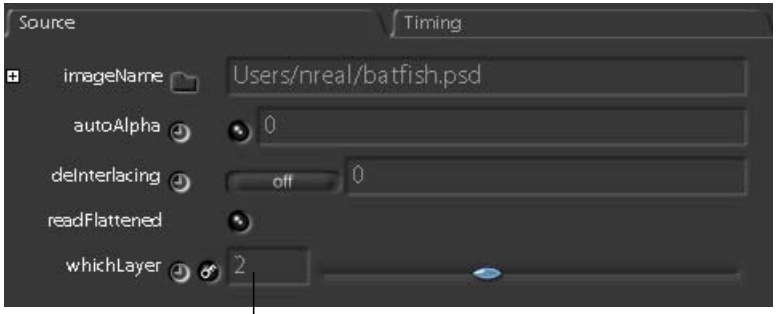
  By default, the image is imported as merged.

To select an individual layer:

1. Load the Photoshop file's *FileIn* parameters (click the right side of the node).
2. In the Source subtree, disable readMerged.
3. In the whichLayer parameter, select the layer you want to display.

   In the following example, the third layer of the Photoshop file is selected.

   When Shake imports a multilayer Photoshop file, the first layer in the file is numbered 0 in the whichLayer parameter. In the following image, the whichLayer parameter is set to 2—the third layer of the Photoshop file (since the first layer is imported as 0, the second layer imported as 1, and so on).



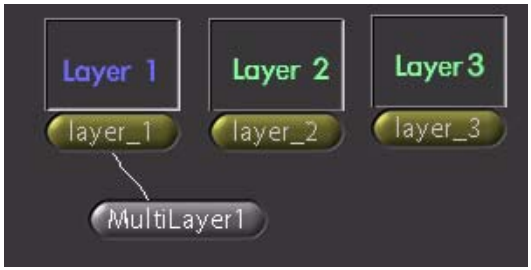
Using the MultiLayer Node

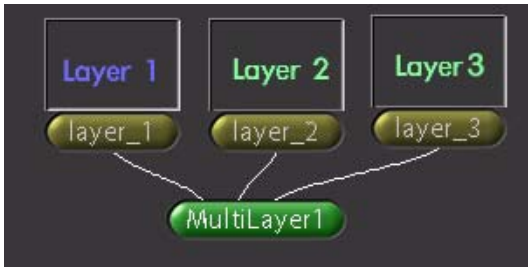
The *MultiLayer* node is a multi-function layering node that distinguishes itself from most of the other layering nodes in two respects—it accepts an unlimited number of input images, and each input image has independent settings interior to the *MultiLayer* node that let you control that layer's compositing mode, opacity, and channels. In a way, the *MultiLayer* node is its own small compositing environment inside of Shake.

When its parameters are opened, you can rearrange the layers via drag and drop directly in the Parameters tab, which allows you to work using a layer-based, rather than node-based logic. This can clean up your tree if you're compositing many different images together in a fairly straightforward way.

Connecting Inputs to a MultiLayer Node

The *MultiLayer* node accepts a variable number of inputs. Drag input noodles onto the + (plus) sign on the top of the *MultiLayer* node that appears when the pointer passes over it. The + sign always appears to the right of all other previously connected knots.



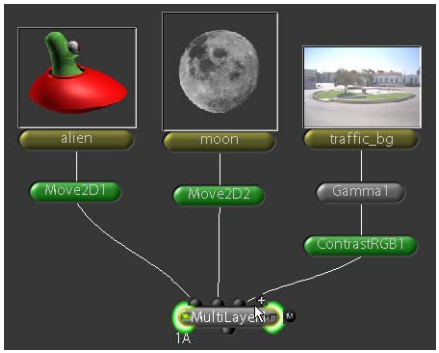


You can also attach several nodes to a single *MultiLayer* node simultaneously.

To connect several nodes to a multi-input node at once:

1 Select all of the nodes you want to attach to the multi-input node.

2 Shift-click the + sign input of the multi-input node.



All selected nodes are connected.

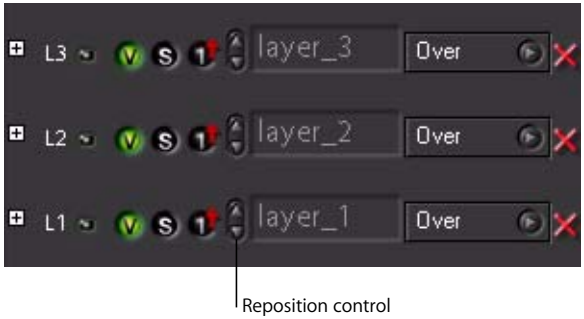
The Order in Which You Connect Nodes

Unlike the other layer nodes, the order in which you connect input images determines the initial layer order of the resulting composite. The first input represents the background layer, the second node is the next deepest, and so on, until you reach the input furthest to the right, representing the foreground.

Layer Order in the MultiLayer Node

Each image that you connect to a *MultiLayer* node is represented by its own set of controls and parameter subtree, located at the bottom of the Images tab.

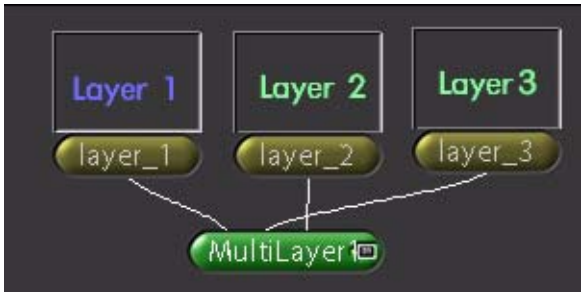
New images are inserted from the bottom up. Unlike the other layering nodes in Shake, layer ordering in the *MultiLayer* node is determined by that layer's position in the layer list—the background layer is the one at the bottom, and the foreground layer is the one at the top. Layers that appear above other layers in the layers list appear in front in the Viewer. This compositing order can be rearranged by rewiring the node in the Node View, or by dragging the Reposition control in the layers list of the Parameters tab.



To change layer order:

- Drag that layer's Reposition control up or down between other layers until a horizontal line appears, which indicates the new position of that layer.

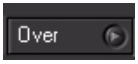
The compositing order is rearranged, and the nodes are rewired in the Node View. In the following illustration, *Layer_1* is the background, and *Layer_2* is the most prominent foreground layer.



Each layer has associated parameters and controls. In the following illustration, several controls are visible on each line.

These controls are, in order:

Control		Description
	Input	Shows the input image for the layer in the Viewer.
	Layer Visibility	Toggles the visibility for that layer. Layers with visibility turned off are not rendered.
	Solo	Turns off the visibility of all other layers. You can only solo one layer at a time. Click an enabled solo button to disable solo.
	Ignore Above	Turns off all layers above the current layer, keeping only the current layer and those behind it visible.

Control		Description
	Input Layer Name Field	Displays the name of the input image node. By blanking it out, the node is disconnected, but the layer information remains.
	Composite Mode	This pop-up menu lets you set each layer with its own composite mode, which affects how that image's color data interacts with other overlapping layers. Certain composite modes add parameters to that layer's parameter subtree. For more information on composite modes, see "Supported Photoshop Transfer Modes" on page 476.
	Disconnect Node	Clicking the Disconnect Node button disconnects that input image from the *MultiLayer* node, and removes it from the layer list without removing the input nodes from the node tree.

In the subtree of a layer, you can controls its parameters. Note that the parameter names are all prefixed by *layerName_* (*L1_* in the following image). The only tricky parameter is channels—it determines what channels get bubbled up from below. For example, to add several files together to fill the matte in a *Keylight* node, insert all the mattes first, then the *Keylight* node on top of the list, using "a" as your channel for that layer.

To rename a layer, expand the subtree and enter the new name in the layerName value field.

clipLayer
A pop-up menu that lists all the input layers currently connected to the *MultiLayer* node. Select which input layer should determine the output resolution.

postMMult
Premultiplies the node.

img
The input image for that layer. This is the area that contains the interface controls for that layer's Visibility, Solo, and so on.

Layer Parameters Subtree

Each layer in the layers list has additional parameters within a subtree that provide additional control.

layerName

The name of the layer. All associated parameters for that layer are prefixed by the *layerName*.

opacity

Opacity of the input layer. With an imported Photoshop file, there is an additional PSOpacity parameter. See "Supported Photoshop Transfer Modes" on page 476 for more information.

preMult

Indicates whether the layer is to be premultiplied.

compChannels

Sets which channels are passed up from below (behind) this layer.

Compositing With the MultiPlane Node

18

The MultiPlane node provides a simple 3D compositing environment within Shake. This environment can be used as a way to arrange and animate images within a 3D space, or as a way to integrate generated or tracked 3D camera paths into your scripts.

An Overview of the MultiPlane Node

The *MultiPlane* node provides a compositing environment for positioning 2D layers within a 3D space. A virtual camera controls the scope of the output image, similar to that found within 3D animation packages. This camera can be animated via keyframed parameters, or by importing 3D camera and tracking data from other 3D animation or 3D tracking applications. As with the *MultiLayer* node, the *MultiPlane* node accepts unlimited input images.

The *MultiPlane* node has two primary uses:

- You can composite background or foreground elements against a moving background plate using 3D camera tracking data, imported from a variety of third-party applications.
- You can arrange multiple layers within a 3D coordinate space for easy simulation of perspective, parallax, and other depth effects.

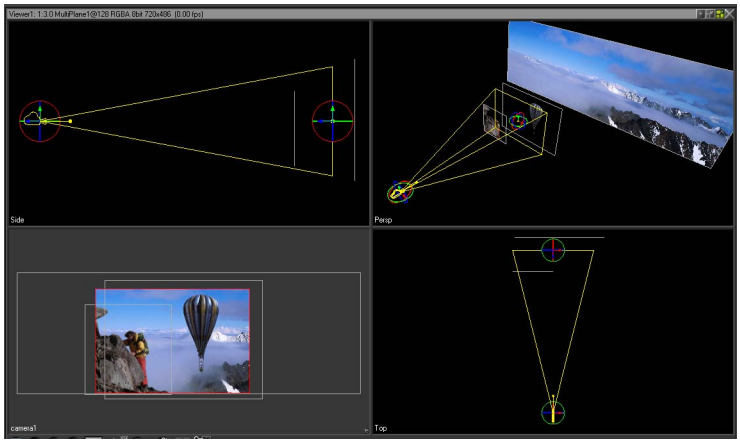
Note: There is one important limitation to positioning layers within the 3D space of the *MultiPlane* node—there is currently no support for intersecting planes. Planes that intersect appear either in front or behind, depending on the position of each layer's center point.

Additionally, the *MultiPlane* node provides transform controls for each layer connected to it. Layers can be moved, rotated, and scaled within the *MultiPlane* environment.

See Tutorial 3, "Depth Compositing," in the *Shake 4 Tutorials* for a lesson on using the *MultiPlane* node.

Viewing MultiPlane Composites

When you double-click a *MultiPlane* node to open it into the Viewer, the Viewer switches to a multi-pane interface, unique to the *MultiPlane* node.



Each pane of this interface can be toggled among single, double, triple, and quadruple-pane layouts. Each individual pane can be set to display any available camera or angle in the 3D workspace, to help you position and transform objects from any angle.

Hardware Acceleration in the MultiPlane Node

The *MultiPlane* node supports OpenGL hardware acceleration of images displayed in the Viewer. Hardware rendering provides a fast way of positioning layers and the camera in space, at the expense of rendering quality.

Whichever pane of the Viewer is set to display the currently selected renderCamera (the default is camera1, although any camera or angle can be in the renderCamera pop-up list) can be toggled between hardware and software rendering using the Render Mode button in the Viewer shelf.

Render Mode Button	Description
 Hardware Rendering Mode	This mode is the fastest for arranging your layers in 3D space, but doesn't provide an accurate representation of the final output. In hardware rendering mode, every layer is composited with an *Over* operation, regardless of that layer's selected composite type.

Render Mode Button	Description
 Hardware Mode While Adjusting	This mode sets the Viewer to use Hardware rendering while you're making adjustments using onscreen controls. Once you've finished, the Viewer goes into Software rendering mode to show the image at the final quality. To turn this setting on, click and hold the Render Mode button, then choose this option from the pop-up menu that appears.
 Software Rendering Mode	This mode displays the selected renderCamera as it appears at its actual quality. All composite types are displayed properly.

The Render Mode button only affects the image that's displayed in the Viewer when the *MultiPlane* node is selected. The image output from the *MultiPlane* node to other nodes in the tree is always at the highest quality, as are *MultiPlane* images that are rendered to disk.

MultiPlane Node Parameters

The Parameters tab of the *MultiPlane* node is divided into two subtabs, the Images and Camera tabs:

- The *Images* tab contains all the parameters for the individual layers that are being arranged in the 3D space of the *MultiPlane* node. For more information on Image tab parameters, see "Parameters in the Images Tab" on page 512.
- The *Camera* tab contains the positioning and optical simulation parameters for each of the cameras and angles used by the *MultiPlane* node. For more information on parameters in the Camera tab, see "Parameters in the Camera Tab" on page 522.

Using the Multi-Pane Viewer Display

Similarly to a 3D application, the *MultiPlane* node's multi-pane Viewer interface displays several angles of the 3D *MultiPlane* workspace. When you first open a *MultiPlane* node into the Viewer, the Viewer switches by default to a four-pane layout, which displays the Side, Perspective, Camera1, and Top angles.

Whether or not the multi-pane interface appears depends on how you open the *MultiPlane* node.

To view the final output from a *MultiPlane* node only:

- Click the left side of the node to display it in the currently selected Viewer without also loading its parameters.

The Viewer displays the final output from that node, just like any other node.

To work within a *MultiPlane* node using the multi-pane interface:

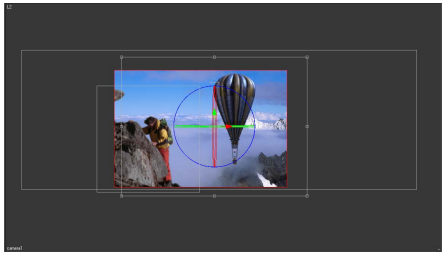
- Double-click a *MultiPlane* node to load its image into the Viewer and its parameters into the Parameters tab. Now, the Viewer switches to the *MultiPlane*'s multi-pane Viewer interface.

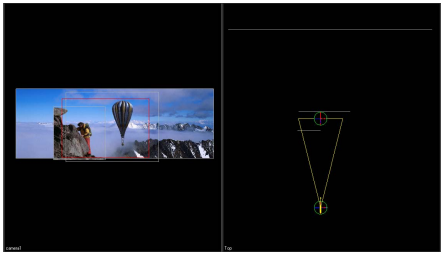
Once the multi-pane interface is displayed, you can toggle among four different layouts.

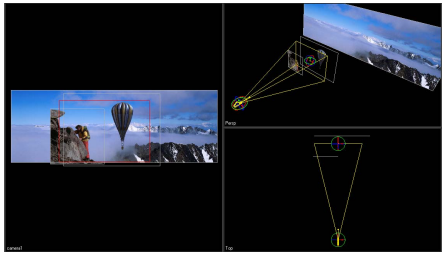
To change the *MultiPlane* Viewer layout:

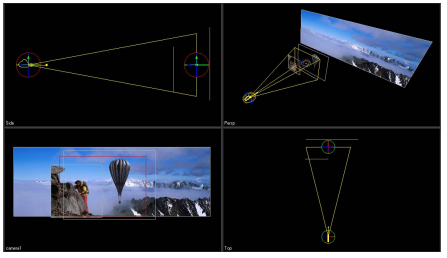
- Click the Viewer Layout button in the Viewer shelf. Keep clicking to cycle among all the available layouts.

The relative size and orientation of each pane in all four layouts is fixed, although you can zoom and pan within any pane using the standard methods.









Using Favorite Views in the Multi-Pane Viewer Display

You can use the standard Favorite Views commands within the Viewer to save and restore the framing of each individual pane. In addition, when you right-click in the Viewer shelf to access the Favorite Views commands that are available from the shortcut menu, you can save and restore the state of all visible panes at once.

Changing Angles Within a Pane

Although the multi-pane layouts are fixed, you can change the angle displayed by each pane at any time. The assigned angles appear in white at the lower-left corner of each pane.

To change the angle displayed by a single pane, do one of the following:

- Right-click within a pane, then choose a new angle from the shortcut menu.
- Position the pointer within the pane you want to switch, and press one of the numeric keypad keyboard shortcuts (0-5) to switch layouts.

The following table lists the available keyboard shortcuts (using the numeric keypad only) that are available for changing angles in a pane.

Numeric Keypad	Description
0	Cycles through every angle.
1	Displays the currently selected renderCamera.
2	Front
3	Top
4	Side
5	Perspective

Using and Navigating Isometric Display Angles

The various angles that are available are intended to help you position layers and the camera within 3D space. Internally to Shake, each angle is actually an invisible camera. The first three angles are isometric views—if you set one of these angles as the renderCamera, you'll notice that the focalLength parameter is set to 0.

You can pan and zoom within any pane using the middle mouse button, and the same keyboard modifiers that apply to other areas in Shake.

The isometric angles are:

- *Front:* Useful for transforming a layer's X and Y pan, angle, and scale parameters. You can pan and zoom to navigate within the Front view.
- *Top:* Useful for transforming a layer's X and Z pan, angle, and scale parameters. You can pan and zoom to navigate within the Top view.
- *Side:* Useful for transforming a layer's Y and Z pan, angle, and scale parameters. You can pan and zoom to navigate within the Front view.

Using and Navigating Within the Perspective Angle

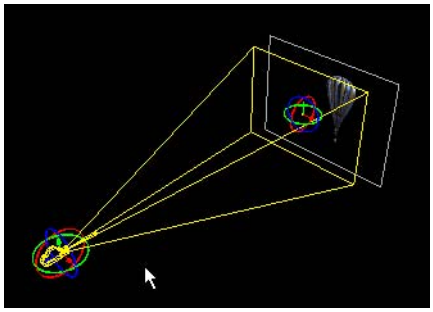
In addition, there is a single non-isometric angle, Perspective (Persp). This is the only pane where you can transform a layer's X, Y, and Z parameters all at once.

In addition to panning and zooming to navigate within this angle, you can also orbit the Perspective angle.

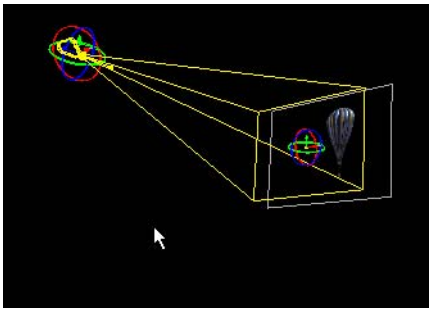
To orbit the Perspective view:

- Move the pointer within a pane displaying the Perspective angle, press X, and drag with the middle mouse button held down.

The Perspective angle rotates about the perspective view's orbit point.



Before rotating the Perspective angle



After rotating the Perspective angle

To center the Perspective view on a selected object:

- Select an object that you want to be the new position of the orbit point (the camera or a layer), and press Shift-B.

The Perspective view's orbit point is moved so that it is centered on the selected object, and the view within that pane is repositioned to a default position.

Note: You can also use the Shift-B keyboard shortcut in the camera view. However, Shift-B only changes the interestDistance parameter of the camera to best match the position of the selected object.

The renderCamera Angle

The camera or angle that's assigned to the renderCamera parameter in the Camera tab is special, since it represents the final output of the *MultiPlane* node. Each *MultiPlane* node is created with one default camera, named camera1. However, you can use the copy button to create duplicate cameras, or import additional cameras into a *MultiPlane* node by importing one or more .ma (Maya ASCII) files.

Whichever angle is assigned as the renderCamera has the following additional properties:

- It's the only angle that can be switched between viewing the software-rendered final output, and the hardware-rendered preview.
- It's the only angle with a compare control (in software-rendering mode only).
- The image border ROI (Region of Interest) appears only for the renderCamera angle.
- There are additional keyboard shortcuts available for transforming a camera. For more information, see "Manipulating the Camera" on page 517.

How the renderCamera Defines Image Output

Even though the renderCamera angle shows the region defined by the camera target's ROI, the image data appearing outside of this area is not cropped. Shake's Infinite Workspace once again preserves this information for use by downstream nodes elsewhere in your script.

Customizing the MultiPlane Default Camera

You can customize the default angles that appear for new *MultiPlane* nodes.

Use the following declaration within a .h preference file:

```
DefDefaultMPCamera( Camera(0, "v4.0", "Front", imageWidth/(2*settings-
    >yPixelUnit), imageHeight/(2*settings->yPixelUnit), 3000,   0, 0, 0,
    "XZY", 0, 0.980, 0.735, 2, "Fill", 0, 0, 0, "default", xFilter, 0, .5,
    0 ) );
```

MultiPlane Viewer Shelf Controls

When you open a *MultiPlane* node into the Viewer, a group of controls specific to the *MultiPlane* node appear in the Viewer shelf. These controls let you toggle the visibility of various onscreen controls in the Viewer.

Button		Description
	Point Cloud Display	Displays/hides the individual locator points that display the imported point cloud data. Displaying this data can make it easier to align layers with the approximate locations of features that were tracked. Hiding this data can make it easier to manipulate the individual layers. For more information on locator points, see "Viewing and Using Locator Points" on page 498.
	XYZ Angle	Displays/Hides the X, Y, and Z angle controls, which can make it easier to reposition objects without accidentally rotating them. A third option is available by clicking and holding down the mouse button to reveal a pop-up menu.

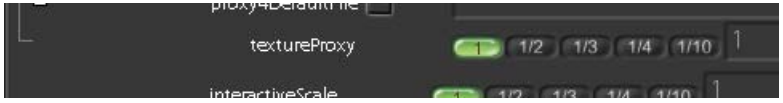
Button		Description
	Path Display	Shows/hides animation paths for image plates.
	Rendering Mode	Toggles the Camera View pane of the Viewer between hardware (HW) and software (SW) rendering. Hardware rendering is faster, but the color and quality of the image is less accurate; this mode makes it easier to position objects. Software rendering is slower, but allows you to accurately see the final image as it will be rendered.
	Viewer Layout	Toggles the Viewer among four different multi-pane layouts, single, double, triple, and quadruple. Each pane can be set to display a different view of the 3D layout.
	Keyframe All or Selected Layers	When this button is turned on, adjusting a single layer using the *MultiPlane* node produces a keyframe that only affects that layer. Animation applied to any other layer is not affected by this new keyframe.

Global Parameters That Affect MultiPlane Display

Two subparameters in the Globals tab let you adjust the quality of hardware-accelerated images displayed within the multi-pane Viewer, and the relative scale of distances represented by imported locator points.

textureProxy

Located within the useProxy subtree, textureProxy sets the proxy level at which texture-rendered images that are used by the *MultiPlane*'s hardware-rendering mode are displayed in the Viewer. This is similar to the interactiveScale setting, in that the proxy level set here is used to generate on-the-fly Viewer images.



multiPlaneLocatorScale

Adjusts the size of locator points that appear in the Viewer when you load data from a .ma file into a *MultiPlane* node. This lets you scale them up to make them easier to select, or down to get them out of the way.

Connecting Inputs to a MultiPlane Node

Like the *MultiLayer* node, the *MultiPlane* node accepts a variable number of inputs. Drag input noodles from other nodes onto the + sign on the top of the *MultiPlane* node that appears when the pointer passes over it. The + sign always appears to the right of all other previously connected knots.

Before

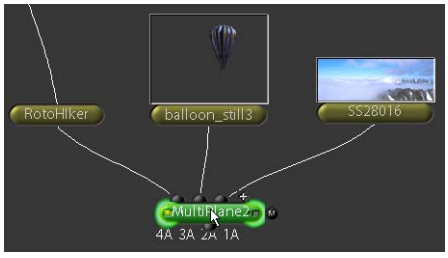
After

You can also attach several nodes to a single *MultiPlane* node simultaneously.

To connect several nodes to a *MultiPlane* node at once:

1 Select all of the nodes you want to connect to the *MultiPlane* node.

2 Shift-click the plus sign input of the *MultiPlane* node.



All selected nodes are connected to the *MultiPlane* node. By default, all new layers you connect appear centered on the Camera view; rotate to face the camera's current position.

Using Camera and Tracking Data From .ma Files

The *MultiPlane* node supports .ma (Maya ASCII) files, allowing you to import 3D camera paths from a variety of 3D animation packages, or use 3D tracking data clouds generated by third-party tracking applications. Every new *MultiPlane* node is created with one camera, named camera1. Importing a .ma file adds a camera to the renderCamera pop-up menu in the Camera tab of the *MultiPlane* node's parameters. You can add as many cameras as you like to a single *MultiPlane* node.

Important: When exporting camera path data from Maya, you must bake the camera data for it to be usable by Shake.

A single .ma file can also contain data for multiple cameras. Shake imports every camera that's found within a .ma file, adding each one to the renderCamera pop-up menu. Choosing a camera from the renderCamera pop-up menu displays that camera's parameters within the Camera tab.

Note: There is currently no support for the culling of 3D tracking data from within Shake. Any manipulation of point cloud data should be performed before that data is imported into Shake.

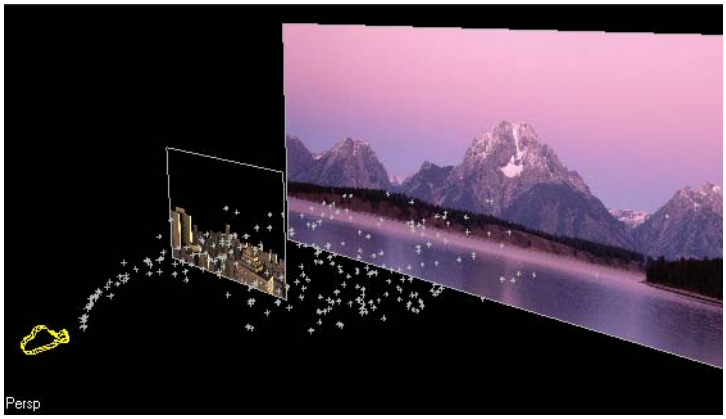
Importing .ma File Data

Data from .ma files is managed using a group of buttons at the bottom of the Camera tab within the *MultiPlane* Parameters tab.

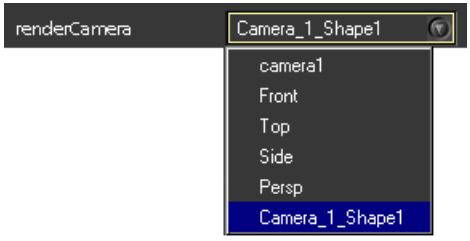
To import a .ma file into a *MultiPlane* node:

1. Load a *MultiPlane* node into the Parameters tab, then click the Camera tab that appears.
2. Click the Load button, choose a .ma file, and click OK.

The data from the .ma file appears in the Viewer as a cloud of points.



The camera or tracking data populates the parameters of the Camera tab, and a new camera appears at the bottom of the renderCamera pop-up menu.



Important: Many 3D applications export camera paths with timing that begins at frame 0. Shake scripts begin at frame 1, which can result in a one-frame offset. To correct this, edit the timing of each camera path point as described in "Editing Locator Point Data" on page 499. (This is not an issue for 3D tracking point clouds—they have no timing data associated with them.)

Once a 3D camera path or tracking data cloud has been imported into the *MultiPlane* node, you can attach the layer that produced the tracking data to the camera. Doing so forces the attached layer to follow along with the camera as it moves to conform to the tracking data. As a result, the attached layer itself doesn't appear to be moving in the camera output. You can tell that the correspondence is correct if the imported tracking points conform to the matching features within the attached layer.

Unattached layers that are positioned within the 3D workspace should now appear as if they're moving along with the features within the attached layer. For more information about attaching a layer to the camera, see "Attaching Layers to the Camera and to Locator Points" on page 506.

Importing Data Clouds From Maya

In Shake, the aspectRatio parameter of a layer within the *MultiPlane* node is determined by the width and height values of the filmBack parameter in the Camera tab. These values are obtained from the imported .ma file.

When tracking a scene in Maya, do one of the following to make sure that the resulting point cloud matches the features of the tracked media:

- Set the film back aspect ratio in Maya to match the render resolution aspect ratio.
- Turn on the "lock device aspect ratio" option in the render globals—this also sets the device aspect ratio to match the film aspect ratio in the camera settings.

Deleting and Duplicating Cameras

To delete a camera, use the Delete button at the bottom of the Camera tab.

To delete a camera and its data:

1. Choose the camera you want to delete from the renderCamera pop-up menu of the Camera tab.
2. Press the Delete button (at the bottom of the Camera tab).

You can duplicate any camera or angle in the *MultiPlane* node with the Copy button. For example, you might want to create a custom viewing angle, or make some changes to a camera without losing the original camera path.

To copy a camera, creating a duplicate:

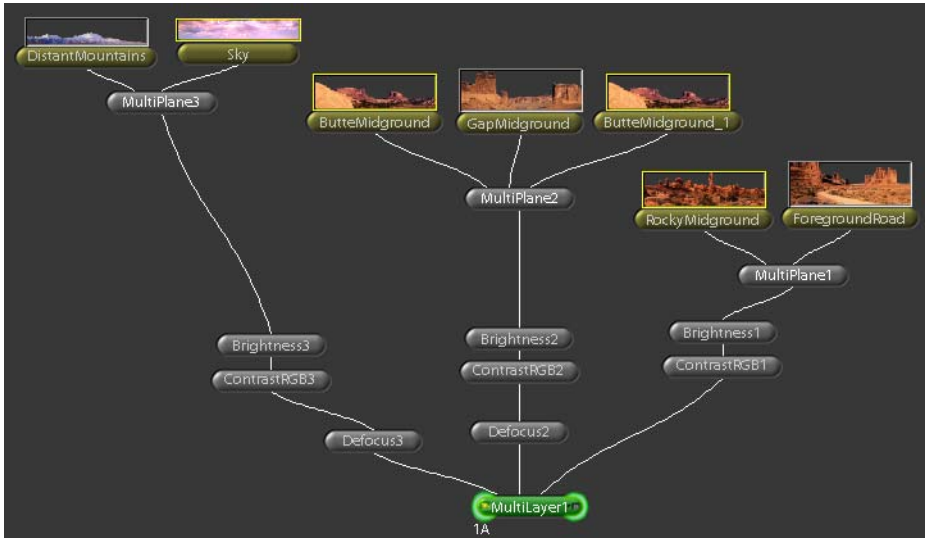
1. Choose the camera you want to copy from the renderCamera pop-up menu of the Camera tab.
2. Click the Copy button.

A duplicate camera appears in the renderCamera pop-up menu.

Linking to a Camera in Another MultiPlane Node

You can link a camera in one *MultiPlane* node to a camera in a different *MultiPlane* node, so that both cameras move together. When you do so, each camera parameter in the current *MultiPlane* node is linked via expressions to the same parameter in the second *MultiPlane* node. By default, each linked parameter is locked to prevent accidental deletion of the link.

Thus, you can set up animated *MultiPlane* composites using multiple *MultiPlane* nodes, instead of connecting all your images to a single *MultiPlane* node. For example, you might set up a composite using three different *MultiPlane* nodes—one for background layers, one for midrange layers, and one for foreground layers. This way you can apply separate color correction and *Defocus* nodes to the output of each sub-composite.

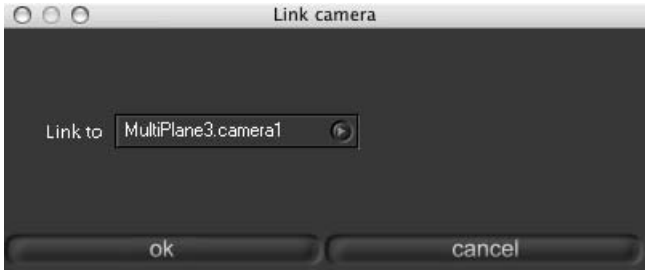


In the above example, one of the foreground image sequences has a matching 3D tracking point cloud that has been imported into the *MultiPlane1* node. For this composition to work, you need to link the cameras in all three *MultiPlane* nodes together.

To link a camera in one *MultiPlane* node to a camera in another:

1. Load a *MultiPlane* node into the Parameters tab.
2. Open the Camera tab.
3. Click the Link button, at the bottom of the Camera tab.

The "Link camera" window appears. The "Link to" pop-up menu presents a list of every camera and angle within every *MultiPlane* node in your script.



4 Choose the camera you want to link to from the "Link to" pop-up menu, then click OK.

The camera in the current *MultiPlane* node is now linked to the camera you chose. Every parameter in the Camera tab is linked, with expressions, to the matching camera parameter of the *MultiPlane* node you chose.

To unlink a camera:

- Unlock each parameter in the Camera tab, then clear each parameter's expression.

Viewing and Using Locator Points

If you've imported 3D tracking data, it's represented in the Viewer by a cloud of locator points arranged within the 3D space of the *MultiPlane* node. Each locator point corresponds to a feature that was tracked by the camera tracking software that created it.

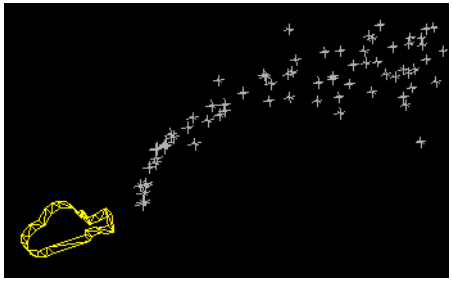
Note: Depending on how detailed the track is, the point cloud may visibly conform to the significant features within its corresponding image sequence. This can help you position elements you're placing within the scene.

To view an imported point cloud:

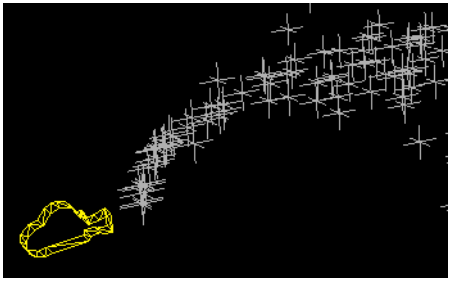
- Click the Point Cloud Display button to hide and show the point cloud.

To change the size of the locator points displayed in the Viewer:

- Adjust the multiPlaneLocatorScale parameter, located at the bottom of the guiControls subtree of the Globals tab.



Default MultiPlaneLocatorScale of 1

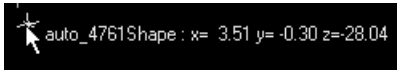


MultiPlaneLocatorScale set to 3

If the data cloud has individually labeled locator points, these labels are viewable within Shake. Some 3D tracking applications also allow you to add locator points by hand—labeling specific areas of the image that may be useful for layer positioning from within Shake.

To view information about a specific locator point:

- Position the pointer directly over an individual locator point in the Viewer to automatically reveal a tooltip with information about that point's name and location.



In addition to simply viewing locator points, you can connect layers directly to individual locator points. For more information about attaching layers to locator points, see "Attaching Layers to Locator Points" on page 510.

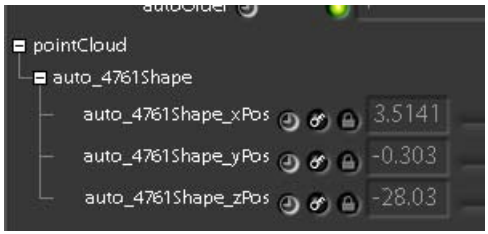
Editing Locator Point Data

You can view and edit the parameters of any locator point. You can also load points from imported camera paths into the Curve Editor to change their timing.

To reveal an individual locator point's data in the Parameters tab:

- Position the pointer over the locator point you want to edit, right-click it, then choose Expose Point from the shortcut menu.

A new subtree named pointCloud appears at the top of the Images tab of the *MultiPlane* node's parameters. Opening the subtree reveals a list of every point that's been added using the Expose command.



Opening an individual locator point's subtree in this list reveals its xPos, yPos, and zPos parameters. These parameters can be loaded into the Curve Editor to edit, animate, or slip their values in time.

Transforming Individual Layers

The *MultiPlane* node allows 3D transformations of all images that have been connected to it using either onscreen controls, or parameters within the Layers tab of the *MultiPlane* parameters. The onscreen controls are similar to those found within the *Move3D* node, except that these transformations occur within an actual 3D workspace.

Layer Transformation Values

Layers can be panned, rotated, and scaled. Transformations made using a layer's onscreen controls are relative to each layer's center point. When moving layers through space, the numeric values for all transformations are relative to the 0,0,0 center point of the *MultiPlane* node's world view.

Layer Onscreen Viewer Controls

Unlike other nodes that present onscreen controls in the Viewer, the *MultiPlane* node lets you select the layer you want to manipulate directly in the Viewer. A layer must first be selected before you can transform it.

To select a layer, do one of the following:

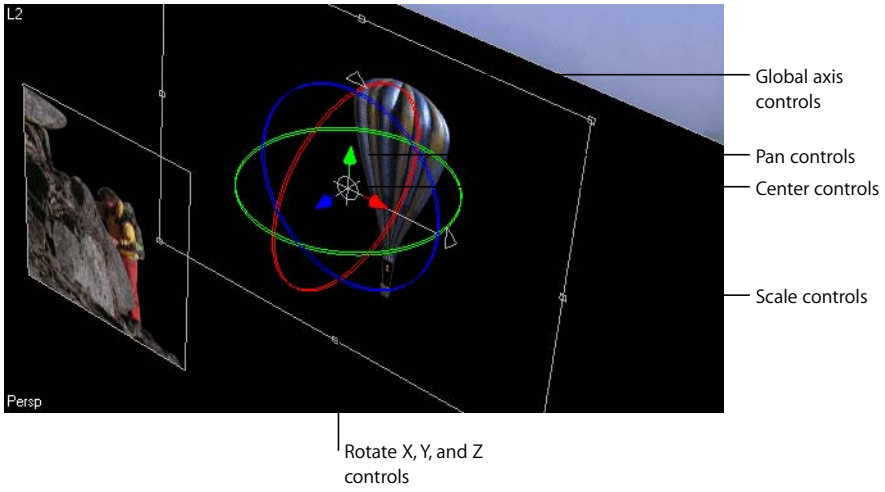
- Position the pointer within the bounding box of any layer's image in the Viewer to highlight that layer. When the pointer is over the layer you want to select, click it to expose the 3D transform controls for that layer.
- Right-click in a pane of the Viewer, then choose a layer from the Current Plan submenu of the shortcut menu.

As you move the pointer over different layers in the Viewer, the affected layer's name appears in yellow text in the upper-left corner of the screen. Once you've selected a layer, its name appears in white text in the same corner.

If you have many layers stacked up within a single *MultiPlane* node, you may want to turn one or more layers invisible to make it easier to manipulate the remaining layers. Invisible layers aren't output when the script is rendered. For more information, see "Showing and Hiding Layers" on page 506.

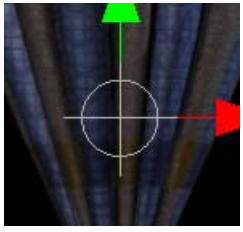
Layer Controls

When you select a layer, that layer's onscreen controls appear superimposed over it.



Center Controls

All transformations you make to a layer occur relative to the center point, indicated by crosshairs in the middle of the onscreen controls.

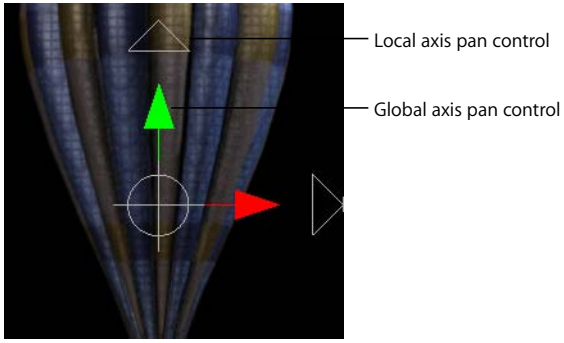


To move the center point:

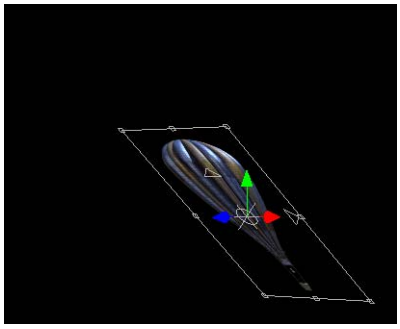
- Hold down the Control key, and drag the center point to a new location.

Pan and Center Controls

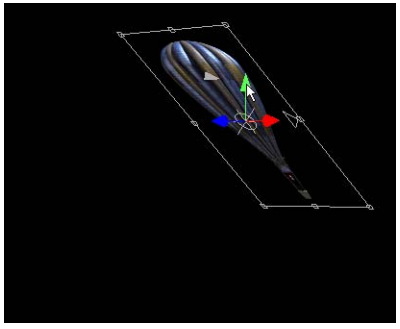
Selected layers in the Viewer have two sets of onscreen controls for panning in 3D space—global axis and local axis pan controls.



Global axis controls pan a layer relative to the overall 3D space, even if the layer has been rotated. Panning a layer up with a global axis control pans it straight up in space.



Before global axis pan

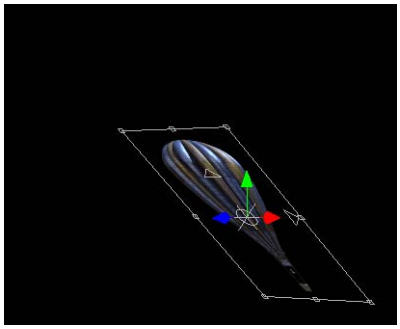


After global axis pan

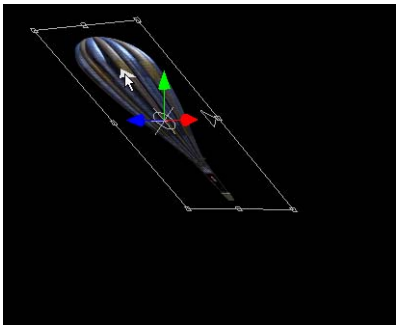
To pan around the global axis, do one of the following:

- Click a layer anywhere within its bounding box, and drag in any direction.
- Drag one of the three global axis pan controls to constrain the layer's direction.
- Select a layer, and press Q or P while dragging anywhere within the Viewer to pan a layer without positioning the pointer directly over it.

The local axis pan controls pan the layer relative to its own orientation. If a layer has been rotated using the angle controls, using the local pan controls moves it along the axis of its own rotation. Panning a layer up with a local axis control pans it diagonally in space.



Before local axis pan



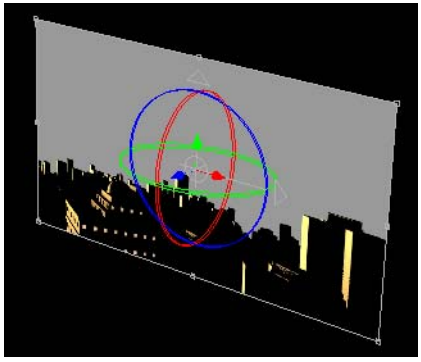
After local axis pan

To pan along the Local Axis:

- Drag one of the two local axis pan controls to move the layer in that direction.

Angle Controls

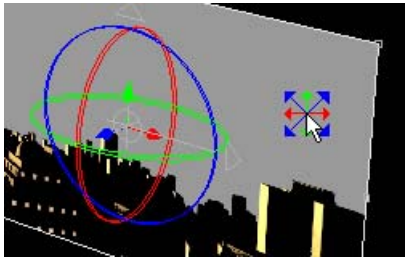
Selected layers have three angle controls that rotate the layer around the X, Y, and Z axes of that layer's center point. These angle controls work identically to those found in the *Move3D* node. The color of each angle control corresponds to the color representing each dimension of the global axis pan controls.



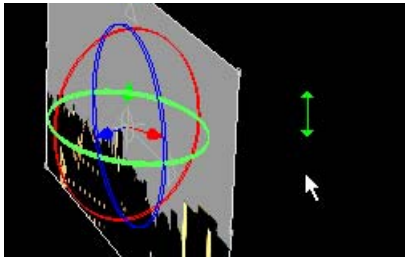
The visibility of the angle controls can be toggled by clicking the XYZ Angle button in the Viewer shelf. Hiding these controls may make it easier to use the pan controls in some situations.

To rotate a layer in the Viewer without using the angle controls:

1. Select a layer.
2. Press W or O and click in a pane of the Viewer.
3. When the dimension pointer appears, move the pointer in the direction in which you want to rotate the layer. The colors in the pointer correspond to the angle controls.



4. When you move the pointer, the axis in which you first move is indicated by a single axis arrow, and the layer rotates in that dimension.



Scale Controls

Selected layers have eight scale controls located around the outer edge of the image.

- The four corner controls rescale the layer, keeping the relative width and height of the layer constrained.
- The left and right controls let you scale just the width of the layer.
- The top and bottom controls let you scale just the height.

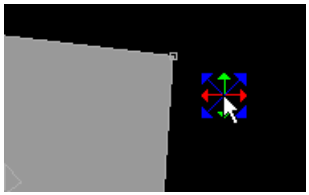
By default, all scale transformations occur about the layer's center point. Alternatively, you can hold down the Command or Control key while you drag any of the scale controls to scale a layer relative to the opposite scale control.

To scale a layer in the Viewer without using the scale handles:

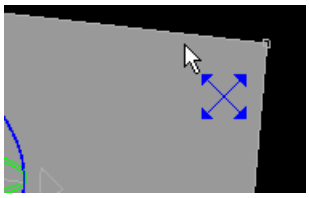
1. Select a layer.
2. Position the pointer over a pane of the Viewer and hold down the E or I key.

   The dimension pointer appears.

3 Drag the dimension pointer in the direction in which you want to scale the layer. The colored arrows correspond to the pan controls.



4 When you move the pointer, the direction in which you first move is indicated by a single axis arrow, and the layer scales up and down in that dimension.



Creating Layer Hierarchies

You can create hierarchical relationships between layers within a *MultiPlane* node using the parentTo parameter. You can lock one layer's transformations to those of another by assigning it a parent layer from this pop-up menu, which contains a list of every layer that's currently connected to the *MultiPlane* node. By default, this parameter is set to off.

Note: The *MultiPlane* node only supports one level of parenting.

To assign a layer a parent layer:

- Choose a parent from that layer's parentTo pop-up menu.

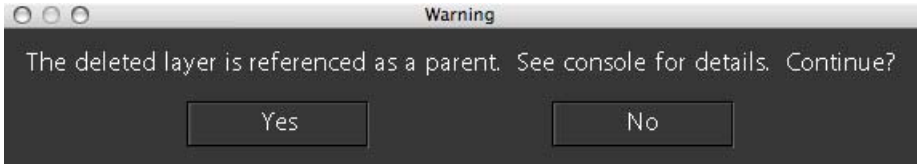
Once a layer has been assigned a parent layer, it is removed from the parentTo pop-up menu. When you select a layer that is parented to another, the parent layer appears with a red border in the Viewer.

You can make local transformations to a layer after you've assigned it a parent. This lets you create an offset between it and the parent layer. Transformations that are applied to a parent layer also affect all layers that are assigned to it.

Important: Always assign a parent layer before making any local transformations to a layer itself. Otherwise you may encounter unpredictable results.

Deleting Parent Layers

When you disconnect a layer that's being used as a parent, a warning appears:



Clicking Yes removes the parent layer, eliminating the parent-child hierarchy. The remaining layers' parentTo parameters still show the original layer, in the event you decide to reconnect it later on.

Showing and Hiding Layers

You can toggle the visibility of layers. For example, if you need to transform a layer that's hard to select because it's obscured by other layers in the composition, you can "solo" it to hide all other layers, or simply hide the individual layers that are in the way. HIdden layers are not rendered.

To show or hide layers, do one of the following:

- Right-click a layer in the Viewer, and turn it on or off in the Plane Visibility submenu.

  The Plane Visibility submenu also has commands to Hide All Planes and Show All Planes.

- Open the Images tab, then click the Layer Visibility button for that layer.

To solo a layer:

- Open the Images tab, and turn on the Solo button for that layer.

  When you solo a layer, every other layer in that *MultiPlane* node is hidden.

Animating Layers

Layer transformations within the *MultiPlane* node can be animated similarly to the parameters within any of Shake's transform nodes. For more information on keyframing parameters, see Chapter 10, "Parameter Animation and the Curve Editor," on page 291.

Attaching Layers to the Camera and to Locator Points

When you import camera path or 3D tracking data into a *MultiPlane* node, you gain the ability to attach layers to the camera, or to one of the locator points distributed within the 3D workspace.

Attaching Layers to the Camera

To use a *MultiPlane* node to matchmove one or more images into a scene using 3D tracking data, you need to do three things:

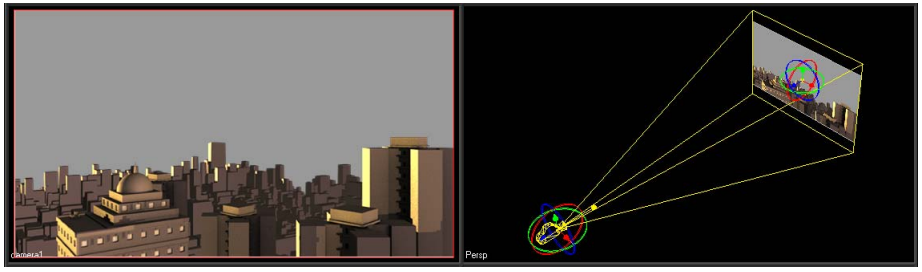
- Import data from a .ma file.
- Position the images you want to matchmove.
- Attach the originally tracked image sequence to the camera.

Attaching the layer that produced the tracking data to the camera forces the attached layer to follow along with the camera as it moves according to the tracking data. As a result, the attached layer itself doesn't appear to be moving in the camera output, while the unattached layers that are positioned within the 3D workspace appear as if they're moving along with the features in the attached layer.

To attach a layer to the camera:

- Click the Attach to Camera button of a layer in the Images tab—the lock icon closes when Attach to Camera is on.

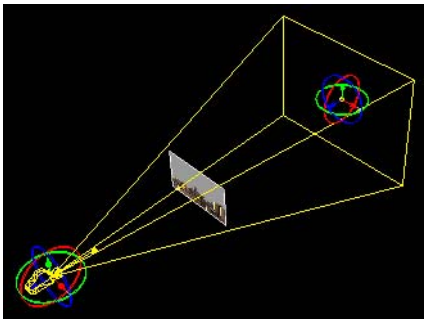
That layer's image is automatically locked to the full area of the renderCamera angle in the Viewer.



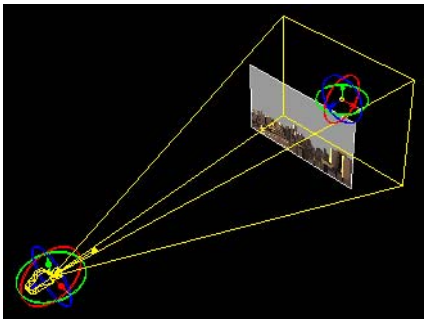
When you turn on a layer's Attach to Camera button, the faceCamera, parentTo, pan, angle, scale, center, and aspectRatio parameters all disappear from that layer's subtree in the Images tab, replaced by a single parameter—cameraDistance.

The cameraDistance parameter lets you adjust the relative spacing between layers that are attached to the camera, and the other unattached layers that are arranged within the 3D workspace. This lets you determine which layers appear in front of and behind attached layers.

Decreasing the cameraDistance value brings the layer closer to the front of the composition, while increasing the cameraDistance pushes the layer farther away. Attached layers move back and forth along the lines that are projected from the camera itself to the four corners of the frustum surrounding the camera target.



cameraDistance reduced



cameraDistance increased

Regardless of how the cameraDistance parameter is set, a layer that's attached to the camera always lines up with the locator points imported from its matching .ma file. Similarly, attached layers always appear, within the camera angle, at their original scale, regardless of their position in the 3D workspace.

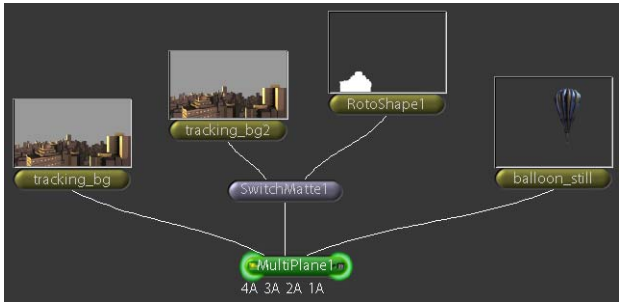
You can attach multiple layers to the camera. A typical example is when you need to isolate one or more foreground elements in a tracked shot, so that a matchmoved element can be placed behind it. In the following example, a hot-air balloon is inserted into a tracked cityscape background plate.

Example: Isolating an Element in a MultiPlane Composite

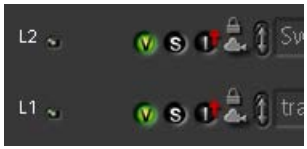
1 First, duplicate the cityscape image sequence that's used as the background plate, and create a rotoshape to isolate the front building.

2 Next, attach the original cityscape image, the isolated building, and the hot-air balloon images to a *MultiPlane* node.

They appear within the Images tab as separate layers.

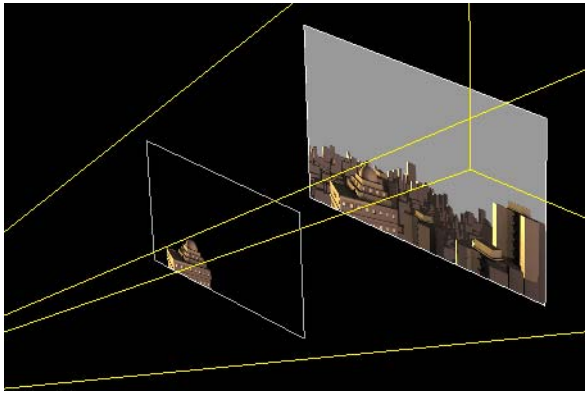


3 Turn on Attach Layer to Camera for the background and isolated building layers to lock both images to the full area of the renderCamera angle in the Viewer.



4 Open the subtree for each attached layer, and adjust the cameraDistance parameters to create the necessary spacing between each element for your composition.

In this case, the front building is moved forward so there's room between the building and the rest of the background plate for the hot-air balloon.

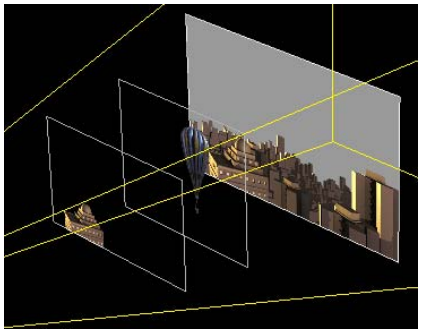


5 Connect the hot-air balloon image to the *MultiPlane* node.

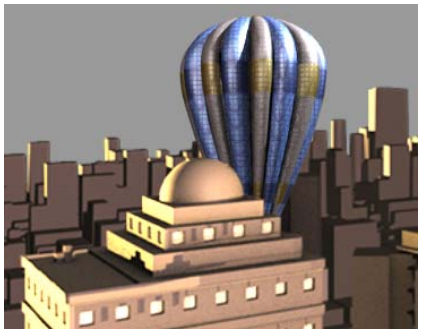
A third layer appears in the layer list.

6 Using the onscreen controls, you can now position this new layer between the isolated building and the overall cityscape plate.

As a result, the balloon appears positioned between the front building and the rest of the city in the camera view.



A balloon image inserted between. the front building and cityscape



The resulting camera angle

Because both the building and city layers are attached to the camera, they look identical to the original input image from which they're derived, regardless of each layer's position within the 3D workspace.

Attaching Layers to Locator Points

In addition to attaching layers to the camera, you can also attach a layer to any locator point in the data cloud. This lets you quickly set a layer's position in space to match that of a specific feature in a tracked background plate.

When you attach a layer to a locator point, the layer is transformed to match the position of the locator point using expressions. As a result, operations that change the position of locator points, such as changing the sceneScale parameter, also change the position of any layers that are attached to locator points. In addition, animated locator points (from a tracking application capable of tracking moving subjects in addition to backgrounds) will transform any layers that are attached to them as well.

You can attach a layer to either a single locator point, or to a group of three locator points. Attaching a layer to a single locator point only pans the layer, it is not rotated.

To attach a layer to a locator point:

1. If necessary, move the layer's center point to a position at which you want the layer to be attached to the locator point.
2. Right-click a locator point in the Viewer, then choose a layer from the Attach Plane to Point shortcut menu.

The layer is panned to the position of the locator point, and attached at the layer's center point. Expressions are added to that layer's pan parameters that maintain the relationship between the attached layer and the locator point.

To attach a layer to three locator points:

1. If necessary, move the layer's center point to a position at which you want the layer to be attached to the locator point.
2. Shift-click three locator points in the Viewer.
3. Right-click one of the selected locator points, then choose a layer from the Attach Plane to Point shortcut menu.

The layer is panned so that its center point is at the center of the triangle defined by the three selected locator points. In addition, it is rotated to match the orientation of the plane defined by the three points. The order in which you select the locator points determines the orientation of the attached layer. If you select the same three points in the reverse order, then choose Attach Plane to Point, the layer will be flipped.

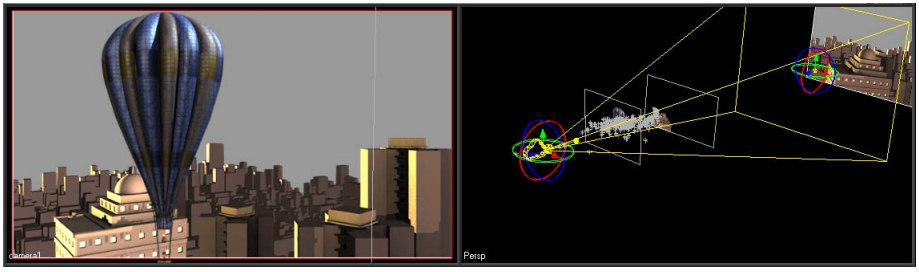
Note: You can attach a layer to three locator points to orient its rotation, then attach a layer to a single point afterwards to nudge its position in space, while maintaining the current rotation.

Adjusting sceneScale

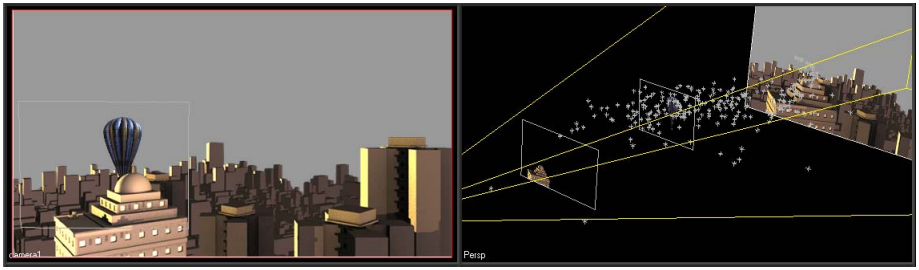
The sceneScale parameter, at the top of the Camera tab, lets you scale the relative distribution of locator points making up a camera path or tracking data cloud within the 3D workspace of the *MultiPlane* node. This lets you increase or decrease the space between different planes within your composition, to make it easier to arrange layers.

If you adjust the sceneScale parameter in the Camera tab, layers that are attached to locator points move along with the expanding or contracting point cloud:

- Lowering sceneScale contracts the distribution of locator points, bringing any layers that are locked to locator points closer to the camera. The locator points themselves appear to bunch up.



- Raising sceneScale expands the distribution of points, moving any layers that are locked to locator points away from the camera. The locator points themselves appear to stretch out.



Changing the sceneScale parameter has no effect on layers that are attached to the camera, nor does it affect the position of layers that are not attached to locator points. It does, however, affect the size of the frustum, increasing or decreasing the area that is seen by the camera. It also changes the position of the camera—if you make a big enough change to the sceneScale parameter, the camera may move past layers in the 3D workspace.

Parameters in the Images Tab

The first three parameters in the Images tab determine the overall image that is output from the *MultiPlane* node.

ClipLayer
Defines the output resolution of image data produced by the *MultiPlane* node.

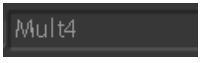
postMMult
When turned on, the postMMult parameter premultiplies the output image produced by the *MultiPlane* node.

autoOrder
When turned off, layer order is determined by the position of layers in the Parameters tab, much like with the *MultiLayer* node. When autoOrder is turned on, layer order is determined by each layer's position in 3D space.

Note: The layer order of coplanar layers (occupying the same coordinate in the 3D workspace) is determined by their position in the Parameters tab.

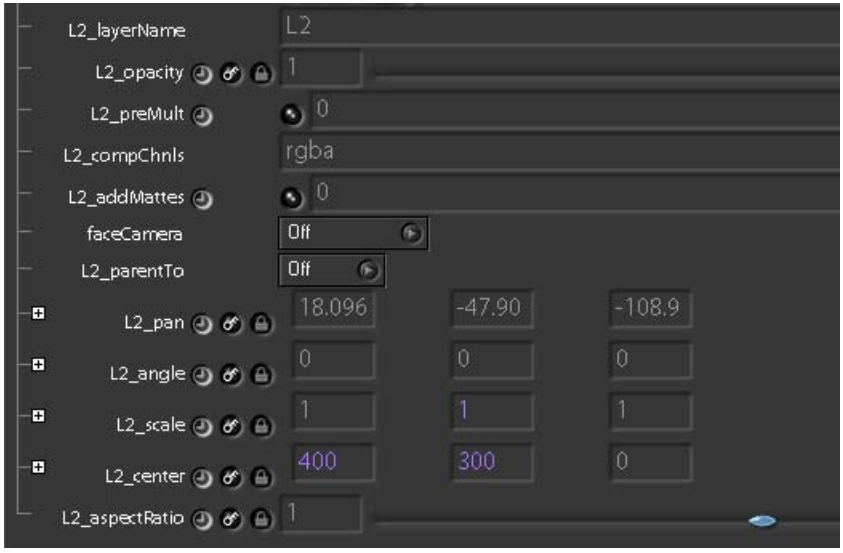
Individual Layer Controls

Each image that you connect to a *MultiPlane* node is represented by its own set of layer controls and subtree parameters in the Images tab. These controls are similar to those found within the *MultiLayer* node, and are labeled in the order in which the layers were connected to the *MultiPlane* node. For example, L1 is the name of the first connected layer, followed by L2, and so on.

Layer Button or Control	Description
 L1	The number of each layer corresponds to the input knot that image is connected to. L1 is the first knot at the left.
 Input	Clicking this button shows the input image for that layer in the Viewer.
 Layer Visibility	Toggles the visibility for that layer. Layers with visibility turned off are not rendered.
 Solo	Turns off the visibility of all other layers. You can only solo one layer at a time. Click an enabled solo control to disable solo.
 Ignore Above	Turns off all layers above the current layer, keeping only the current layer and those behind it visible.
 Attach Layer to Camera	If a particular layer corresponds to a tracked sequence that's imported from a .ma file, turn this button on to attach the image to the camera, so that it's not transformed when the camera moves to follow a track. As a result, unattached layers appear locked relative to the tracking information when composited against the attached layer.
 Reposition	Dragging this control up and down lets you rearrange the layer order within the parameter list. This only affects layers that are coplanar, unless autoOrder is turned off.
 Input Layer Name Field	Gives the name of the preceeding node that's actually connected to the *MultiPlane* node.
 Composite Mode	A pop-up menu that lets you set each layer with its own composite mode, which affects how that image's color data interacts with other overlapping layers. Certain composite modes add parameters to that layer's parameter subtree. New layers default to the Over mode. For more information on composite modes, see "Supported Photoshop Transfer Modes" on page 476.
 Disconnect Mode	Disconnects that input image from the *MultiPlane* node, and removes it from the layer list without removing the input nodes from the node tree.

Individual Layer Parameters

Opening up a layer's parameter subtree reveals a group of compositing and transform parameters affecting that particular layer.



layerName

The name of the layer. All associated parameters for that layer are prefixed by the *layerName*.

opacity

Opacity of the input layer. With an imported Photoshop file, there is an additional PSOpacity parameter. See "Supported Photoshop Transfer Modes" on page 476 for more information.

preMult

When this is on (1), the foreground image is multiplied by its alpha mask. If it is off (0), the foreground image is assumed to already be premultiplied.

compChannels

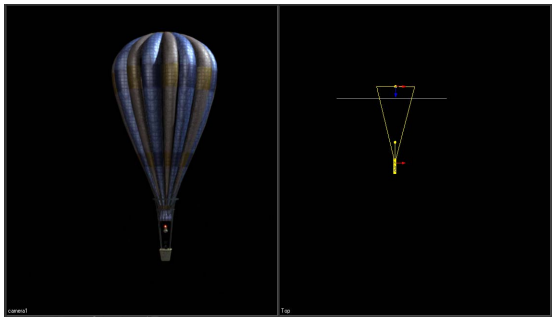
Sets which image channels are passed up from below (behind) this layer.

addMattes

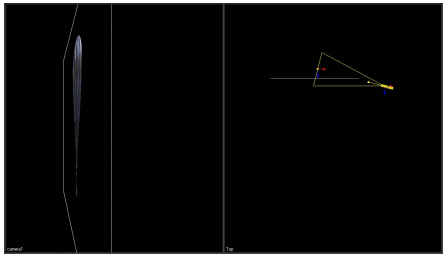
This parameter appears when a layer is set to Over. When enabled (1), the mattes are added together to create the composite.

faceCamera

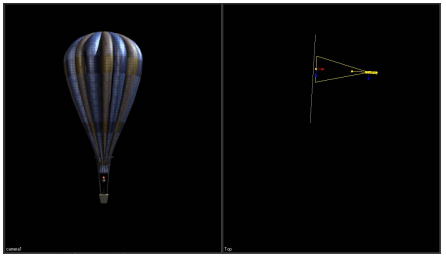
The faceCamera pop-up menu lets you choose a camera. Once set, the layer will always rotate to face the same direction as the selected camera. If the camera is animated, the layer animates to match the camera's movement so that the object remains facing the camera at every frame. This automatically animates that layer's angle parameters, even though no keyframes are applied. In the following example, the camera starts off facing a 2D balloon layer.



With faceCamera turned off, moving the camera around the layer results in the image flattening as the camera reaches the side of the layer. When faceCamera is set to Camera1, the image automatically rotates to face the direction the camera is facing.



Rotated camera with faceCamera turned off

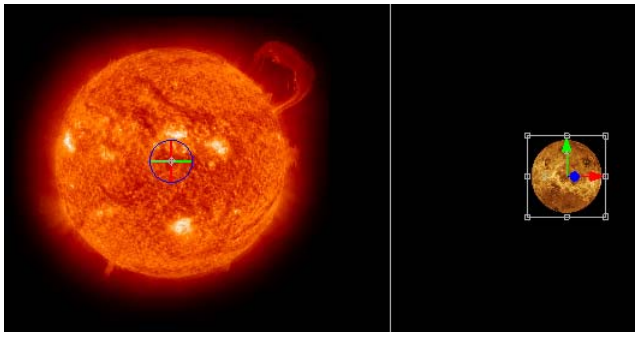


Rotated camera with faceCamera

Unlike ordinary rotation made with a layer's angle controls, the automatic rotation that's made as a result of faceCamera being turned on is relative to the center point of the bounding box that defines the actual image of the layer, and *not* by the layer's center point.

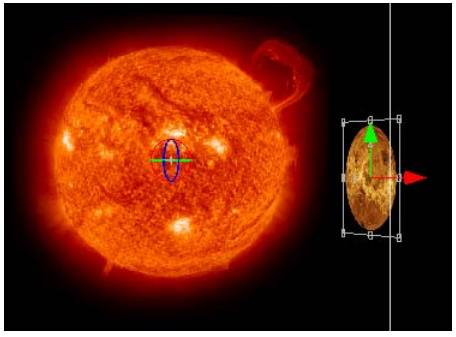
One useful application of this parameter is to offset a layer's center point, then use layer rotation to control layer position, even though faceCamera is turned on. In the following example, an image of a planet is arranged to the right of an image of the sun. The planet layer's center point is offset to the left to match the center of the sun layer.

The arrangement in the Camera pane looks like this:

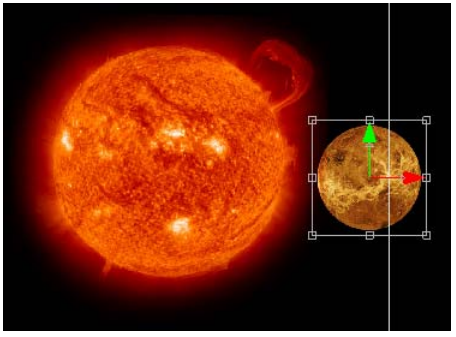


Sun and Venus images courtesy NASA/JPL-Caltech

Rotating the planet layer's yAngle parameter (corresponding to the green axis arrow) now moves the planet layer around the sun layer, so that it appears to orbit about the sun. By default, the 2D planet layer thins out as it approaches the camera. Setting faceCamera to Camera1 results in the planet layer rotating to face the camera as it moves around the sun.



Rotated planet with faceCamera turned off



Rotated planet with faceCamera set to Camera1

parentTo
This parameter lets you create layer hierarchies within a single *MultiPlane* node. You can link one layer to another by choosing a parent layer from this pop-up menu. Layers with a parent can still be transformed individually, but transformations that are applied to the parent are also applied to all layers linked to it. For more information on using the parentTo parameter, see "Creating Layer Hierarchies" on page 505.

pan (x, y, z)
Corresponds to the onscreen pan controls, allows you to pan the layer in 3D space.

angle (x, y, z)
Corresponds to the onscreen angle controls, allows you to rotate the layer in 3D space.

scale (x, y, z)
Corresponds to the onscreen scale controls, allows you to resize the layer in 3D space.

center (x, y, z)
Defines the center point of the layer, about which all pan, rotate, and scale operations occur. By default the center of newly added layers corresponds to the horizontal and vertical center of the layer's bounding box.

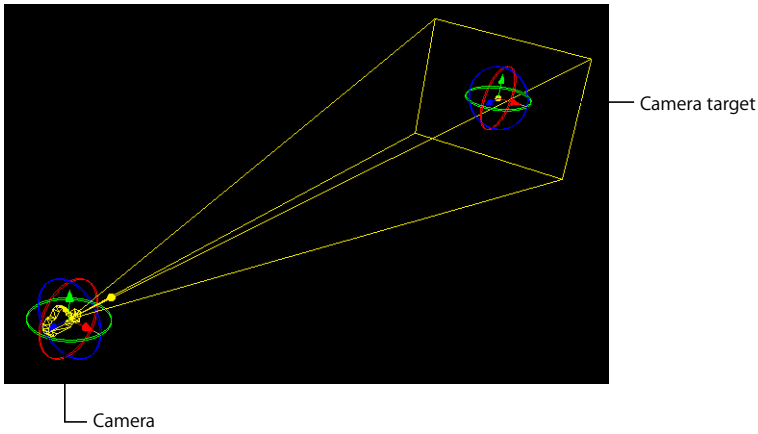
aspectRatio
This parameter inherits the current value of the defaultAspect global parameter. If you're working on a nonsquare pixel or anamorphic image, it is important that this value is set correctly to avoid unwanted distortion in the image.

Manipulating the Camera

The camera can either be positioned or animated manually, like any other layer, or by importing 3D tracking or camera data via a .ma file.

3D Transform Controls

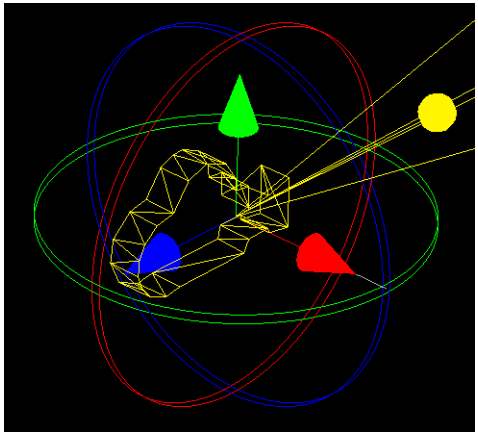
Click the camera to expose its 3D transform controls.



The camera consists of two groups of controls, the camera itself, and the camera target. Both controls are connected so that they always face one another—moving one rotates the other to match its position.

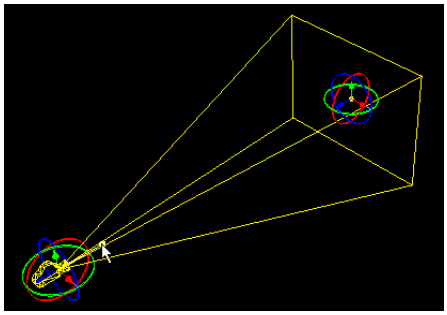
Camera Controls

Similar to layer controls, the camera has rotate x, y, and z controls, and translate (move) x, y, and z controls to constrain movement of the camera to one of these directions. Dragging the camera image itself freely moves the camera within the Viewer.

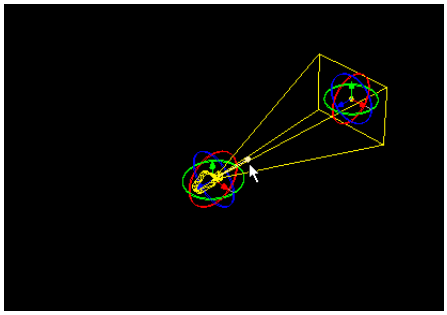


Moving the camera in relation to the camera target

An additional control at the front of the camera constrains its movement so that it moves either closer to or farther away from the camera target—which also adjusts the interestDistance parameter in the Camera tab of the *MultiPlane* parameters.

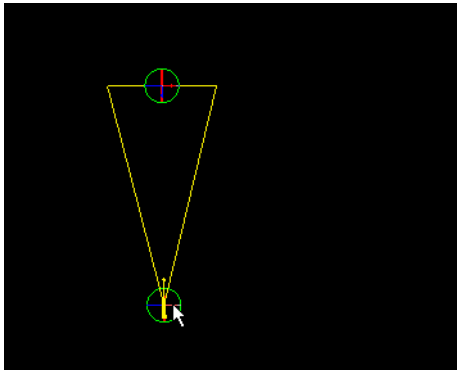


Before moving interestDistance control

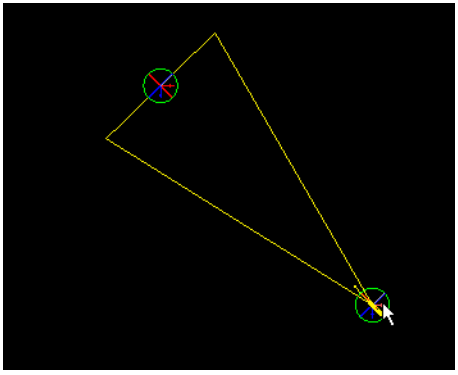


After moving interestDistance control

No matter where you move the camera in space, it rotates to face the position of the camera target. The orientation of the camera target, in turn, rotates to face the camera.



Before panning camera



After panning camera

Additionally, a special group of keyboard shortcuts lets you move the camera by dragging anywhere within the renderCamera view, without selecting the camera itself.

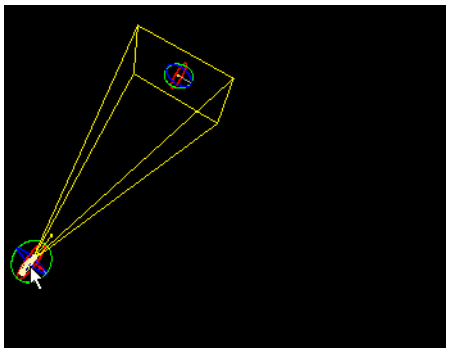
Note: These keyboard shortcuts only work within the camera view:

Keyboard	Explanation
V-drag	Rotates the camera about its Z axis.
S-drag	Rotates the camera about the X and Y axes, about the camera's own center point, changing the position of the camera target.
Z-drag	Pans the camera in and out along the Z axis.

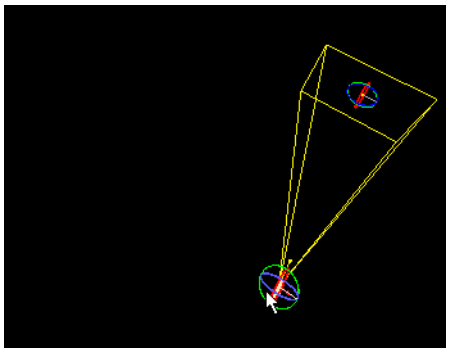
Keyboard	Explanation
D-drag	Pans the camera and camera target together along the X and Y axes.
X-drag	Pivots the camera about the camera target's orbit point.

To move the camera and the camera target together in any view:

- Press T and drag the camera or camera target controls in the Viewer.



Before T-dragging camera

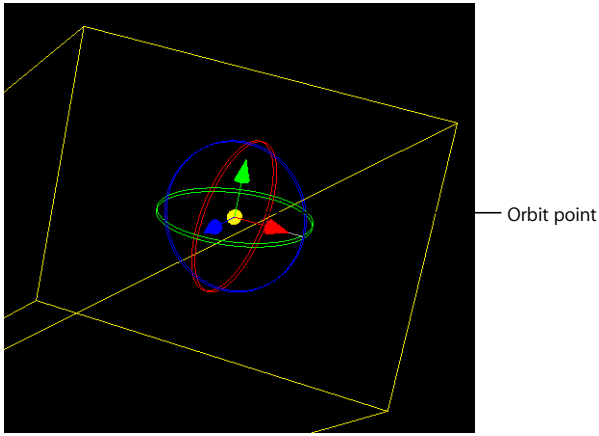


After T-dragging camera

Camera Target Controls and Frustum

The camera target represents the image that is displayed by the camera1 view. In addition to the standard angle and panning controls, the camera target displays the orbit point as a yellow sphere at the center of the target controls. This is the point about which the camera rotates when you move the camera independently of the camera target, or when you manipulate the camera target angle controls.

Note: The orbit point is also the point about which the Persp (perspective) view rotates when you rotate this view using the X key.

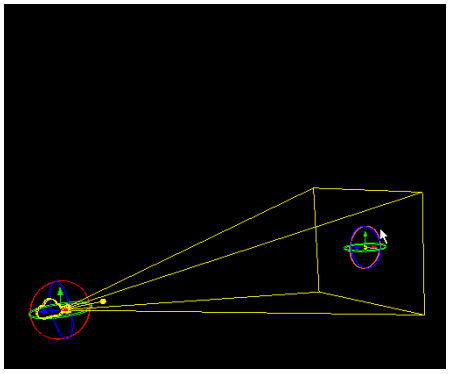


The outer bounding box represents the frustum, which defines the camera view frame that produces the output image. The size of the frustum determines, in part, the area of the 3D workspace that is output as the renderCamera image that's output by the *MultiPlane* node.

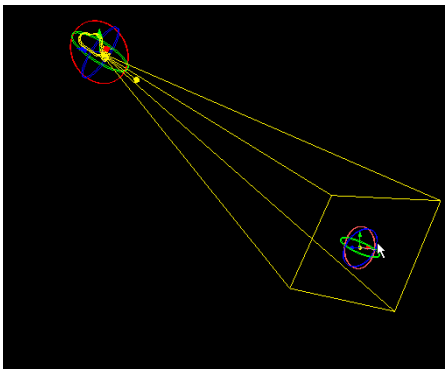
Note: Unlike frustum controls found in other 3D animation packages, Shake's *MultiPlane* frustum does not crop the image outside of the frustum boundary. Thanks to Shake's Infinite Workspace, this image data is preserved for future downstream operations in the node tree.

The controls within let you move the camera target. Using the translate (move) x, y, and z controls moves the camera target—causing the camera to rotate so that it continues to face the target.

Using the rotate x, y, and z controls rotates the camera about the camera target's orbit point.



Before rotating camera target



After rotating camera target

Moving the camera target closer to or farther away from the camera adjusts the interestDistance parameter in the Camera tab of the *MultiPlane* parameters.

Animating the Camera

The camera can either be animated manually, like any other object, or by importing 3D tracking or camera data via a .ma file.

Animating the Camera Using .ma Data

You can use the Copy, Load, Delete, and Link buttons at the bottom of the Camera tab of the *MultiPlane* parameters to import, use, and clear camera and tracking data from a .ma file.

Use the sceneScale parameter at the bottom of the guiControls subtree of the Globals tab to change the size of the locator points displayed in the Viewer.

Parameters in the Camera Tab

All of the parameters that affect the camera are located within the Camera tab of the parameters.

renderCamera
A pop-up menu that lets you choose which angle provides the output image from the *MultiPlane* node.

Lock, Unlock, Reset
Click the Lock button to lock all of the camera parameters at once, preventing them from being accidentally edited. Click the Unlock button to unlock these parameters again. Click the Reset button to restore the camera to its default position.

cameraName
A text field that lets you rename the camera or angle to more accurately describe your setup.

sceneScale
Scales the depth of the virtual space used to distribute the locator points that are displayed in the Viewer (which represent 3D tracking data clouds imported from .ma files). This parameter allows you to expand or compress the relative distance from the camera to the tracked background plate. Adjusting this parameter lets you more easily position layers in space when camera tracking data comes from a subject that's either very far away, or very close. This parameter is for reference only, and has no effect on the data itself. The default multiPlaneLocatorScale value is 50.

Camera Transform Data

The following parameters contain transform data for the camera, as well as parameters that determine how the image is resolved based on mathematically simulated camera attributes such as focal length and film gate resolution. These parameters are adjusted whether you simply reposition the camera within the 3D workspace, keyframe the camera manually, or import a camera path or 3D tracking data from a .ma file.

Note: Many of the parameters found within the Camera tab are identical to parameters exported by Maya. For more information, see the Maya documentation.

Lock, Unlock, Reset
Three buttons let you lock all camera parameters, unlock all camera parameters, or reset all camera parameters.

focalLength
Sets the focal length of the camera's virtual lens. Shake measures the focal length in millimeters. The default focalLength is 35mm.

angleOfView
A subparameter of focalLength. This value is provided for convenience, and represents the horizontal field of view of the frustum (as opposed to the vertical field of view that the *Move3D* node uses), based on the current focalLength. The value of the angleOfView has an inverse relationship to that of the focalLength parameter—raising one lowers the other, and vice versa.

translate (x, y, z)
Transform data for the camera's position. By default, the translate parameters are set to expressions that automatically set their values.

rotate
Transform data for the camera's rotation. If you keyframe rotation, the data is stored here.

rotateOrder
The order in which rotations are executed, by dimension. Defaults to XZY.

interestDistance
This parameter defines the distance of the camera target to the camera. By default, this parameter is set to an expression that automatically sets its value.

filmGate
The filmGate pop-up menu provides presets for setting the filmBack width and height parameters. There are options corresponding to most standard film formats.

filmBack (width, height)
Represents the width and height of the image that strikes the virtual film, in inches.

Note: If you're not importing camera data and you want to avoid squeezing the image improperly, make sure the aspect ratio of the filmBack width and height parameters matches that of the input image.

scale
Represents the size of the camera compared to the scene. Lowering this value reduces the visible area taken in by the virtual lens, increasing the size of the scene relative to the viewer. Increasing this value increases the visible area seen by the lens, reducing the size of the scene relative to the viewer.

Changing the scale parameter also changes the *effective* focal length of the camera. For example, if the focalLength parameter is set to 50, setting the scale to 2 changes the effective focal length to 25. Setting the scale to 0.5 changes the effective focal length to 100.

fitResolution
A pop-up menu with four options: Fill, Horizontal, Vertical, and Overscan. This parameter determines how differences between the filmBack aspect ratio and that of the input image are resolved.

filmFitOffset
If the filmBack resolution is different from that of the clipLayer, this parameter offsets the image within the filmBack resolution in inches.

filmOffset (x, y)
Offsets the image within the filmBack in relation to the area defined by the clipLayer. This parameter is measured in inches.

useDeviceAspectRatio
If this parameter is turned off, the camera uses the aspect ratio of the input image. If this parameter is turned on, the deviceAspectRatio parameter is used instead.

deviceAspectRatio
By default, this parameter uses an expression that computes the aspect ratio based on the filmBack parameter.

xFilter, yFilter
Lets you pick which method Shake uses to transform the image in each dimension. For more information, see "Applying Separate Filters to X and Y Dimensions" on page 863

motionBlur
A Motion Blur quality level of 0. 0 produces no blur, whereas 1 represents standard filtering. For more speed, use a value less than 1. This value is multiplied by the global parameter motionBlur.

- **shutterTiming**

  A subparameter of motionBlur. Zero (0) is no blur, whereas 1 represents a whole frame of blur. Note that standard camera blur is 180 degrees, or a value of .5. This value is multiplied by the global parameter shutterTiming.

- **shutterOffset**

  A subparameter of motionBlur. This is the offset from the current frame at which the blur is calculated. Default is 0; previous frames are less than 0.

.ma Load/Camera Copy, Delete, Link Buttons

Four buttons at the bottom of the Camera tab let you control how .ma camera data is used within a *MultiPlane* node.

Load
This button lets you load .ma data from a file. This creates a new camera in the currently open *MultiPlane* node.

Copy/Delete
These buttons let you duplicate or delete the currently selected renderCamera.

Link
The link button at the right lets you link the camera in the currently open *MultiPlane* node to an animated camera within another.

Delete Cloud
Deletes a point cloud, but leaves the camera angle intact. This is useful if you plan on redoing the track and you want to clear the old point cloud in preparation for importing a new one.

Using Masks

19

This chapter describes how you can use masks in Shake to create transparency and to limit the effects of other functions within your node tree.

About Masks

Masking is the process of using one image to limit another. This typically takes the form of assigning one image to be used as an alpha channel by another. Masking in Shake can also take the form of using one image to limit the effect of a particular node in the node tree.

Masking is closely related to keying. *Keying* is a process for creating (pulling) a matte, typically using color (green or blue) or brightness (whites or blacks) information from the image to mask that image. *Masking* is even simpler—it's simply assigning one image to be used as a matte to another image or operation. (For more information on keying, see Chapter 24, "Keying," on page 681.)

Masks in Shake are extremely flexible, and can be combined in any number of different ways. You can create masks that add to, or subtract from, the existing alpha channel of images anywhere within your node tree.

Using Side Input Masks to Limit Effects

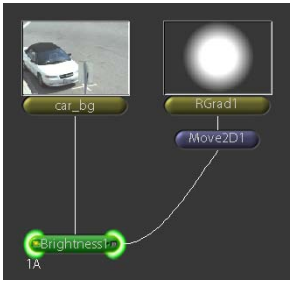
You can attach a mask to the side input of a node, thereby limiting that node's effect on the input image. In the following screenshots, a mask image (actually an *RGrad* image node modified by a *Move2D* node) is used to limit the effect of a *Brightness* node that's connected to the source image of the car.

Source image

Mask image

By connecting the mask image to the side input of the *Brightness* node, parts of the source image remain unaffected by the *Brightness* operation.

Masked Brightness node



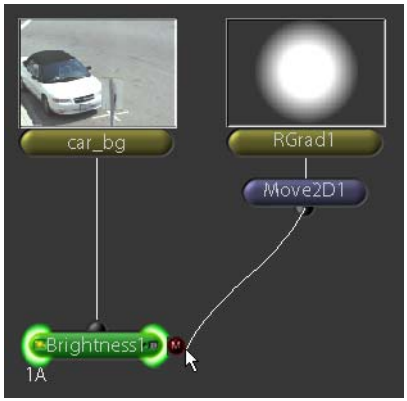
Corresponding node tree

Important: The side input is meant to be used with effects nodes only. Do not use side input nodes to mask images.

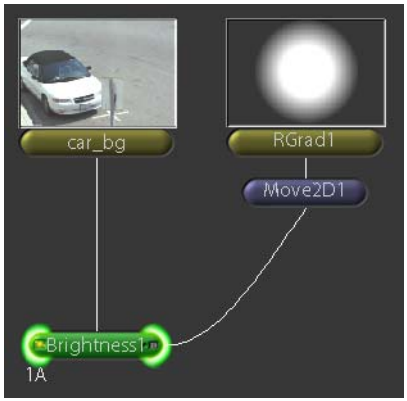
You can set up a node to use a side input mask in one of two ways. You can connect an existing mask image to the side input node of any effect node, or you can open an effect node's parameters and create an automatically connected side input mask using the Mask controls.

To attach an image in the node tree to a side input mask:

- Drag a noodle from an image's output knot, and attach it to a node's side input mask.



Drag noodle to side input..



Connected side input mask

To create a side input mask:

1. Load the parameters of the node you want to mask into the Parameters tab.
2. Do one of the following:
   - Click Create to create a new instance of the type of node listed in the pop-up menu to the right.
   - Choose a different type of node from the Create pop-up menu to the right.

A new image node is created, automatically connected to the side input mask.

Adding Custom Nodes to the Mask Shape List

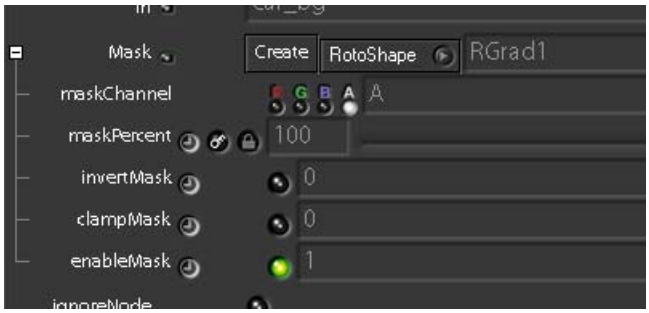
To add your own nodes to the Mask shape list, add a line similar to the following in a *ui.h* file:

```
nuiAddMaskCommand("QuickShape","QuickShape();");
nuiAddMaskCommand("QuickPaint","QuickPaint(0);");
```

For more information on *ui.h* files, see Chapter 14, "Customizing Shake," on page 355.

Parameters Within the Mask Subtree

The Mask subtree, located in the top section of any node's Parameters tab, contains the following parameters that let you customize how the input mask image is used:



maskChannel
Lets you choose which channel to use as the mask. This parameter defaults to A (alpha).

invertMask
Lets you invert the mask, reversing its effect on the image.

clampMask
Turning this parameter on clamps mask image data to a value between 0 and 1. It is important to enable this parameter when using floating point images as masks.

enableMask
Lets you turn the mask off and on without having to disconnect it.

Using Masks to Limit Color Nodes

The following example uses images from the tutorial media directory (*$HOME/nreal/Tutorial_Media/Tutorial_Misc/masks)* to show how to limit the effects of color nodes using masks.

Masking a color-correction node to create a lighting effect:

1. In the Image tab, click the *FileIn* node, and select the *car_bg.jpg*, *woman_pre.iff*, *sign_mask2.iff*, and *car_mask.iff* images in the */Tutorial_Misc/masks* directory, then click OK.
2. In the Node View, select the *car_bg* node.
3. Click the Color tab, and then click *Brightness*.

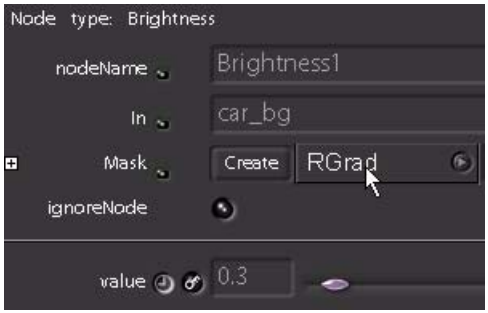
   A *Brightness* node is added to the *car_bg* image.
4. In the *Brightness* parameters, set the value to .3.

   The entire image darkens.

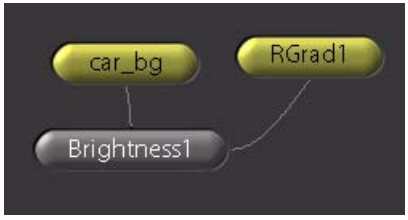
5 To create a mask that gives the appearance of a "spotlight," do one of the following:

- Create an *RGrad* node (in its own branch), and connect the *RGrad* output to the M (mask) input on the side of the *Brightness* node.
- In the *Brightness* parameters, choose RGrad from the Mask pop-up menu.

Note: To create the node type already in the Mask shape menu, click Create. For information on the rotoscoping or paint tools and their onscreen controls to draw and edit masks, see Chapter 21, "Paint," on page 579.



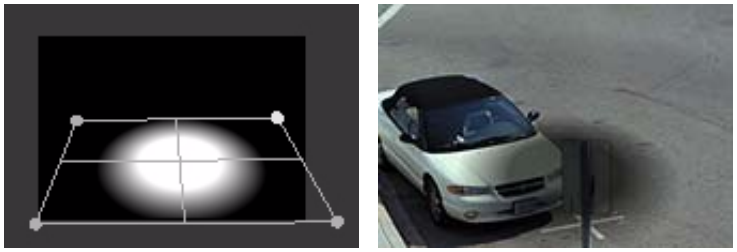
An *RGrad* is connected as the mask input for the *Brightness* node, and the masked portion of the image is darkened.



6 In the Node View, select the *RGrad* node, click the Transform tab, then click the *CornerPin* node.

7 Using the onscreen controls and the following image as a reference, adjust the *RGrad* image to put the circle in perspective.

For more information on transforming with onscreen controls, see Chapter 26, "Transformations, Motion Blur, and AutoAlign," on page 763.



8 To invert the mask, open the Mask subtree in the *Brightness* node, and enable invertMask.

The mask is inverted, and the masked portion of the image is lightened.

Don't Use Mask Inputs With Layer Nodes

Mask inputs are useful for color corrections and transforms. However, masks should not be used for layer nodes. The logic is the complete opposite of what you think it should be. Honest. As the following example shows, even with color and transform nodes, masks should be used with caution.

Masking Concatenating Nodes

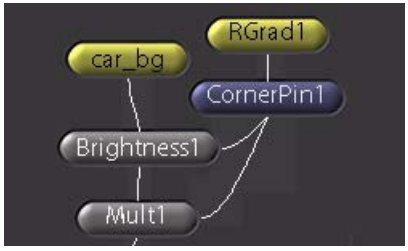
It is never a good idea to use side input masking with multiple successive concatenating nodes because doing so breaks the concatenation. The following example demonstrates the wrong way to use masks.

Breaking node concatenation with side input masks—don't try this at home:

1. Select the *Brightness* node and apply a Color–*Mult* node.
2. In the Color controls of the *Mult* node parameters, set the Color to blue.

   A blue tint is created to color correct the dark areas of the background.
3. Connect the output of the *CornerPin* node to the M input of the *Mult* node.
4. To invert the mask on the *Mult* node, expand the Mask controls and enable invertMask.

   The *Mult* (color-correction) node is masked and the mask is inverted (like the *Brightness* node), so that only the dark areas are tinted blue.



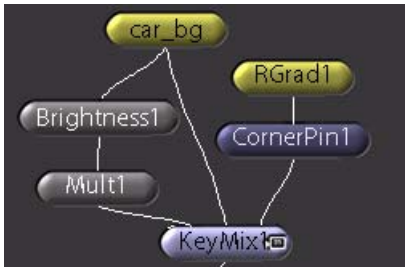
5. In the *Mult* node, adjust the Color controls to a deeper blue color.

Although the result appears fine, there are several problems with the above approach:

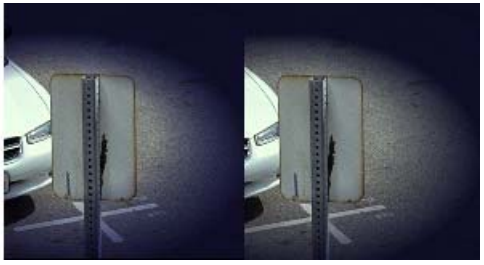
- Normally, the *Mult* and *Brightness* functions concatenate. By masking either node, you break concatenation. When concatenation is broken, processing time slows and accuracy decreases.
- Masking twice with the same node (the *RGrad* node in this example) slows processing.
- Your edges get multiple corrections, and tend to degrade. This is evident by the blue ring around the soft parts of the mask.

A better way to mask a series of concatenating nodes:

1. Disconnect the masks from the previous example.
2. Select the *Mult* node and add a Layer–*KeyMix* node.
3. Connect the *car_bg* node output to the *KeyMix* node's Foreground input (the second input).
4. Connect the *CornerPin* node to the *KeyMix* node's Key input (the third input).



This eliminates the above problems. The following images compare the two resulting renders. In the right image that used the *KeyMix* node, there is no blue ring around the soft part of the mask area.



Masking Transform Nodes

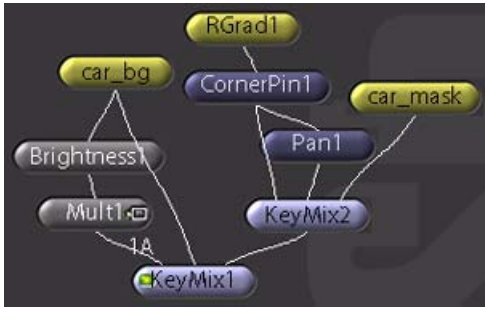
You can also use masks to isolate transforms.

To mask a pan to create depth between the street and the hood of the car:

1. Select the *CornerPin* node, and apply a Transform–*Pan* node.
2. Select the *CornerPin* node again, and Shift-click Layer–*KeyMix*.

   A *KeyMix2* node is added to the *CornerPin* node as a separate branch.
3. Connect the *Pan* node to the Foreground input of the *KeyMix2* node.
4. Connect the *car_mask* node to the Key input of the *KeyMix2* node.

5 Connect the *KeyMix2* node to the Key input of the *KeyMix1* node.

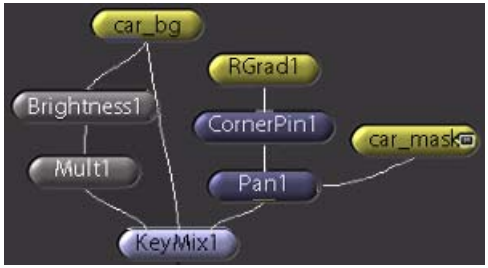


6 Click the right side of the *KeyMix1* node to show the resulting image in the Viewer.

7 Click the left side of the *Pan* node to show the onscreen controls in the Viewer, and load the parameters.

8 Using the following illustration as a guide, pan the *RGrad* up slightly to the left.

Note: If you had simply connected the *car_mask* image to the M input of the *Pan* node, rather than using the *KeyMix* method, you would have masked normally concatenating nodes and broken the concatenation between the *CornerPin* and *Pan* functions.



Using Images Without an Alpha Channel

A masked image does not need an alpha channel. Connecting an image without an alpha channel as the mask doesn't immediately have an effect, however, since by default mask inputs are expecting an alpha channel.

To fix this, switch the mask channel to R, G, or B in the Mask subtree to select a different channel to use as a mask. To use the luminance of the image, apply a *LumaKey* node to your mask image and leave the channel at A, or apply a *Monochrome* node and select R, G, or B.

Masking Layers

Another form of masking involves using an image as a holdout matte to cut holes in another image. Masking layers requires a different approach, since you should *never* mask a layer using the side input.

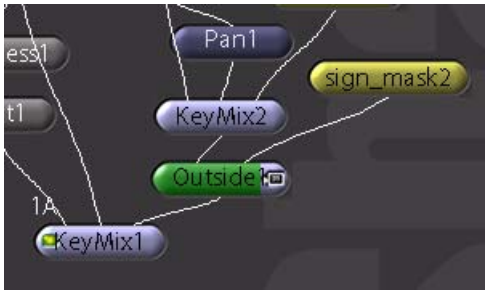
Typical nodes used for this task are *Inside* and *Outside*. The *Inside* node puts the first image in only the white areas of the second image's alpha, and the *Outside* node puts the first image in only the black areas of the second image's alpha.

The following example continues with the result node tree from the above example. Since the sign is further in the foreground of the scene, you do not want the sign to get the brightening effect. Use the *Outside* node to put the light mask outside of the sign mask, in effect punching a hole in the light mask with the sign mask.

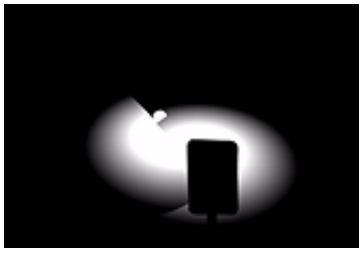
Using the *Outside* node to isolate the sign:

1. In the Node View, select *KeyMix2* and apply a Layer–*Outside*.
2. Double-click the *Outside* node to load it in the Viewer.

3 Connect the *sign_mask2* node to the Background input (the second input) of the *Outside* node.



The light mask is "outside" of the sign mask.

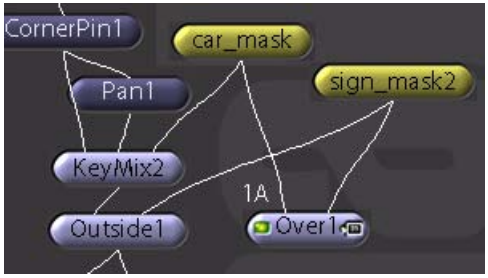


Outside1

KeyMix1

The following example demonstrates the *wrong* way to combine the sign and car masks.

- Using the following image as a guide, combine the *sign_mask* and the *car_mask* with an *Over* (or *Max*) node.



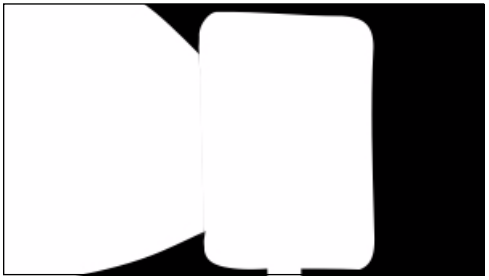
A slight problem occurs when you try this using the *Over* node. A matte line appears between the two masks.

Fortunately, in Shake, there's always a different method you can try. This problem is easily solved by substituting a different node for the *Over*.

A better way to combine the sign and car masks:

1 Select the *Over* node, and Control-click *IAdd*.

The *Over* node is replaced with the *IAdd* node, and the line disappears.



Next, you can put the woman outside of the new mask *IAdd1* using the *Outside* node. Since this technique was used in the previous example, try a different approach. Use the *Atop* node—similar to *Over*, except the foreground only appears where there is an alpha channel on the background image.

2 Add a *SwitchMatte* node, and connect the *KeyMix1* node to the Foreground input of the *SwitchMatte* node.

3 Connect the *IAdd1* node to the Background input of the *SwitchMatte* node.

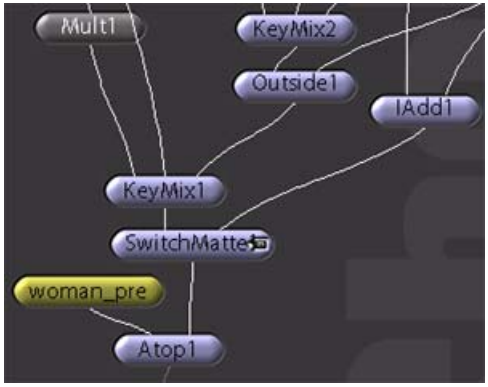
The alpha channel is copied from *IAdd1* to *KeyMix1*.

4 In the *SwitchMatte* node parameters, enable invertMatte to invert the mask (since *Atop* only composites in the white areas in the background alpha mask).

5 In the *SwitchMatte* node parameters, disable matteMult.

6 Select the *woman_pre* node and apply a Layer–*Atop* node.

7 Connect the *SwitchMatte* node to the Background input of the *Atop* node.



If the image looks wrong, make sure that matteMult is disabled, and invertMatte is enabled in the *SwitchMatte* node parameters.

Masking Filters

Filters have special masked versions of the node that not only mask an effect, but also change the amount of filtering based on the intensity of the second image. These take the same name as the normal filter node preceded by an *I*, for example, *Blur* and *IBlur*. This is much more convincing than using the mask input.

To mask a filter:

1 Create an Image–*Text* node, and type some text in the text field. (Type in the second field labeled "text" since the first field is to change the name of the node.)

2 Adjust the xFontScale and yFontScale parameters so the text fills the frame.

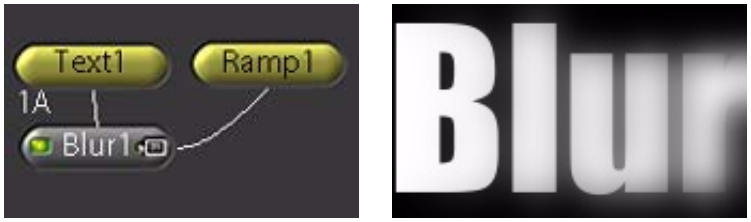
3 Create an Image–*Ramp*, and set the alpha1 parameter to 0.

4 Select the *Text* node, and add a Filter–*Blur* node.

5 Connect the *Ramp* node to the M input of the *Blur* node.

6 In the Blur parameters, set the xPixels and yPixels value to 200.

The result looks bad, rather like the following. Notice that the right side of the image merely mixes the completely blurred image with the non-blurred image.



7 Select the *Blur* node, and Control-click Filter–*IBlur*.

The *Blur* node is replaced with the *IBlur* node.

8 Disconnect the *Ramp* from the blur node's M input, and connect it to the *IBlur* node's second image input.

9 In the *IBlur* parameters, set the xPixels and yPixels value to 200.

The result is much nicer—the right side is blurred to 200 pixels, the middle is blurred to 100 pixels, and the left edge has no blur at all.

The -mask/Mask Node

This node is only used in the script, but is created invisibly whenever you insert a side-input mask. The *Mask* node masks out the previous operation (in command-line mode) or a node that you specify when in scripting mode. This is how the interface interaction of setting a mask is saved in script form. For more information, see "About Masks" on page 527.

Parameters	Type	Defaults	Description
mask	image		The image to be used as a mask on the result of the first input image.
maskChannel	string	"a"	The channel of the mask image to be used as the mask.

Parameters	Type	Defaults	Description
percent	float	100	A gain control applied to the maskChannel. • 100 percent is full brightness. • 50 percent is half brightness. • 200 percent is twice as bright, and so on.
invertKey	int	0	A switch to invert the maskChannel. • 0 = do not invert • 1 = invert
enableKey	int	1	A switch to turn the key on and off. • 0 = off • 1 = on

Synopsis

```
Mask(
  image,
  image mask,
  const char * maskChannel,
  float percent,
  int invertKey,
  int enableKey
);
```

Script

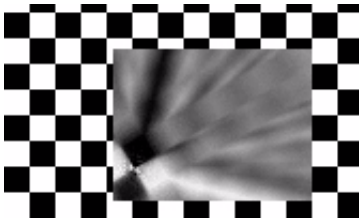
```
image = Mask(
  image,
  mask,
  “maskChannel”,
  percent,
  invertKey,
  enableKey
);
```

Command Line

shake -mask image maskChannel percent...

Masking Using the Constraint Node

The Layer–*Constraint* node also helps to limit a process. The *Constraint* node mixes two images according to a combination of modes. The modes are Area of Interest (AOI), tolerance, channel, or field. In the following example, the AOI is enabled and the area box is set. Only the area inside of the box of the second image is calculated.



Constraint

Constraint is a multifunctional node that restricts the effect of nodes to limited areas, channels, tolerances, or fields. Toggle the type switch to select the constraint type. Certain parameters then become active; others no longer have any effect. This is similar to the *KeyMix* node in that you mix two images according to a third constraint. *KeyMix* expects an image to be the constraint. *Constraint* allows you to set other types of constraints.

The *Constraint* node also speeds calculation times considerably in many cases. The speed increase always occurs when using the ROI or field mode, and for many functions when using channel mode. Channel mode decreases calculation time when the output is a result of examining channels, such as layer operations. Calculation time is not decreased, however, when it must examine pixels, such as warps and many filters. The tolerance mode may in fact increase calculation times, as it must resolve both input images to calculate the difference between the images.

Parameters

This node displays the following controls in the Parameters tab:

clipMode

Toggles between the foreground (0) or the background (1) image to set the output resolution.

type

Selects the type of constraint you use.

- *AOI - Area of Interest (1):* Draws a mixing box.
- *Threshold (2):* Only changes within a tolerance are passed on.
- *Channel (4):* Only specific channels are modified.
- *Field (8):* Only a selected field is modified.

Because of the labeling, you can do multiple types of constraining in the script by adding the numbers together. For example, 7 = AOI (1) + Threshold (2) + Channel (4); in other words, AOI, Threshold, and Channel are all active.

AOI Controls
These are active only if the *type* parameter is set to 1. (See "type," above.) They describe a cropping box for the effect. Opening this parameter reveals left, right, bottom, and top subparameters.

rTol
If the type parameter is set to 2, the red color channel tolerance. If pixels between the two images are less than the Tolerance value, they are considered common.

gTol
If the type parameter is set to 2, the green color channel tolerance. If pixels between the two images are less than the Tolerance value, they are considered common.

bTol
If the type parameter is set to 2, the blue color channel tolerance. If pixels between the two images are less than the Tolerance value, they are considered common.

aTol
If the type parameter is set to 2, the alpha channel tolerance. If pixels between the two images are less than the Tolerance value, they are considered common.

thresholdType
Active only when the type parameter is equal to 2. This sets the Tolerance to "lo" or "hi."
- *0* = "lo." Changes are made only if the difference between image1 and image2 is less than the Tolerance values you set.
- *1* = "hi." Changes are made only if the difference between image1 and image2 is greater than the Tolerance values.

tReplace
Active when the *type* parameter is set to 2. Toggles whether the entire pixel is replaced, or just the channel meeting the Tolerance criteria.

Channel and Field Controls
Opening this parameter reveals two subparameters.

channels: If the *type* parameter is set to 4 (see "type," above), the operation only applies to the specified channels.

field: If the *type* parameter is set to 8 (see "type," above), effect only applies to one field.

- *0* = even field
- *1* = odd field

invert
Inverts the selection. For example, everything beyond a color tolerance is included, rather than below, and so on.

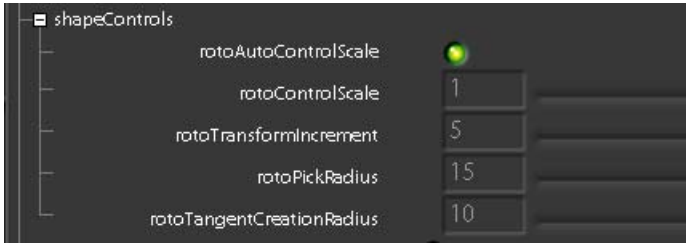
Rotoscoping

20

Shake provides rotoscoping capabilities with the RotoShape node. When combined with Shake's other image processing, layering, and tracking functions, you have a powerful rotoscoping environment.

Options to Customize Shape Drawing

Before you start working with Shake's *RotoShape* node, you should be aware that there are several parameters in the guiControls section of the Globals tab that allow you to customize shape-drawing behaviors and shape-transform controls in the Viewer. You can change these parameters to make it easier to use Shake's controls for your individual needs.



rotoAutoControlScale
An option which, when enabled, increases the size of the transform controls of shapes based on the vertical resolution of the image to which the shape is assigned. This makes it easier to manipulate a shape's transform control even when the image is scaled down by a large ratio.

rotoControlScale
A slider that allows you to change the default size of all transform controls in the Viewer when rotoAutoControlScale is turned on.

Note: You can also resize every transform control appearing in the Viewer by holding the Command key down while dragging the handles of any transform control in the Viewer.

rotoTransformIncrement
This parameter allows you to adjust the sensitivity of shape transform controls. When this parameter is set to lower values, transform handles move more slowly when dragged, allowing more detailed control. At higher values, transform handles move more quickly when dragged. A slider lets you choose from a range of 1-6. The default value is 5, which matches the transform control sensitivity of previous versions of Shake.

rotoPickRadius
This parameter provides the ability to select individual points on a shape that fall within a user-definable region around the pointer. This allows you to easily select points that are near the pointer that may be hard to select by clicking directly. A slider allows you to define how far, in pixels, the pointer may be from a point to select it.

rotoTangentCreationRadius
This parameter lets you define the distance you must drag the pointer when drawing a shape point to turn it into a Bezier curve. Using this control, you can make it easier to create curves when drawing shapes of different sizes. For example, you could increase the distance you must drag to avoid accidentally creating Bezier curves, or you can decrease the distance you must drag to make it easier to create Bezier curves when drawing short shape segments.

Using the RotoShape Node

The *RotoShape* node can create multiple, spline-based shapes that can be used as an alpha channel for an element, or to mask a layer or an effect. You can only create closed shapes with the *RotoShape* node. Shapes created using the *RotoShape* node are grayscale, and filled shapes are white against a black background. An alpha channel is automatically created, and has exactly the same data as the R, G, and B channels.

Shapes can be filled or unfilled. For a shape to have an effect on the alpha channel, it must be filled. Shapes that are filled with white create solid areas in the alpha channel. Shapes that are filled with black create areas of transparency. Unfilled shapes have no effect on the alpha channel.

Why Use the RotoShape Node Instead of the QuickShape Node?

The *RotoShape* node is a newer, faster, more flexible, and more able rotoscoping tool that replaces the *QuickShape* node.

The *RotoShape* node has the following advantages over the *QuickShape* node:

- You can create multiple shapes within the same node.
- You can have a soft-edge falloff on each shape that can be modified independently on each control point.
- You can make one shape cut a hole into another.
- It is much faster to enter keyframes.
- Once you break a tangent, the tangent remains at the angle you specify until you break the tangent again.

This chapter covers the *RotoShape* node as it's used for rotoscoping. For techniques on using the *RotoShape* node to apply masks, see Chapter 19, "Using Masks."

Note: You can copy shapes, either partially or in their entirety, between the *RotoShape*, *Warper*, and *Morpher* nodes. When copying a shape from a *RotoShape* node to a *Warper* or *Morpher* node, you can assign it as a source, target, or boundary shape. This is especially useful in cases where you've isolated a subject using a *RotoShape* node already and you can use that shape as a starting point for your warp effect. Be aware that you cannot copy shapes in *RotoShape* nodes that were created in Shake 3.5 or earlier.

Add Shapes Mode Versus Edit Shapes Mode

When the *RotoShape* node is active, the associated tools appear on the Viewer shelf.

There are two main modes you'll toggle between when using the *RotoShape* controls in the Viewer shelf.

Add Shapes Mode

You initially create shapes using the Add Shapes mode.

Edit Shapes Mode

You modify and animate shapes using the Edit Shapes mode.

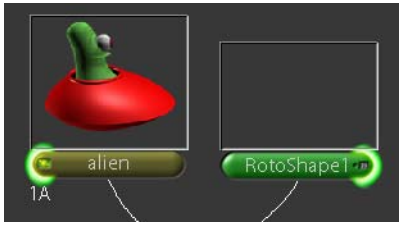
Drawing New Shapes With the RotoShape Node

Drawing new shapes works the same whether you're creating a source, target, or boundary shape. In each case, you create a new, unassigned shape first, then assign its type in a subsequent step. Unassigned shapes appear yellow, by default.

To create a new shape:

1. Add an Image–*RotoShape* node to the Node View.
2. Click the Parameter control of the *RotoShape* node to load its parameters into the Parameters tab, and its controls into the Viewer shelf.

   Note: If you're rotoscoping over the image from a particular node, click the left side of the node you want to trace in the Node View to load its image into the Viewer. Make sure the *RotoShape* node's parameters remain loaded in the Parameters tab, otherwise the shape controls will disappear from the Viewer shelf.

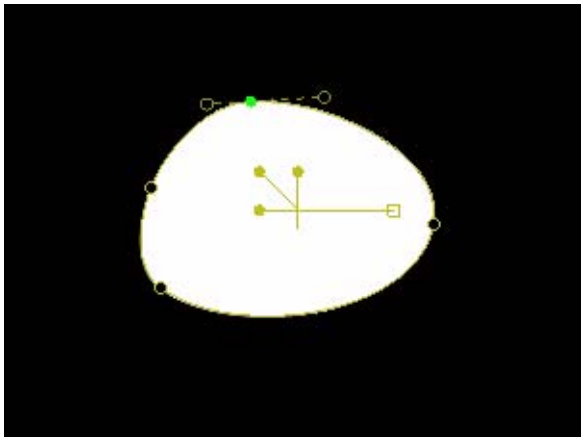


3. In the Viewer shelf, click the Add Shape button.

4. If necessary, zoom into the image in the Viewer to better trace the necessary features of the subject.
5. In the Viewer, begin drawing a shape by clicking anywhere to place a point.
6. Continue clicking in the Viewer to add more points to the shape.
   - Click once to create a sharply angled point.
   - Drag to create a Bezier curve with tangent handles you can use to edit the shape of the curve.
7. To close the shape, click the first point you created.

   The shape is filled, and the Edit Shapes mode is automatically activated.

Note: If you traced the image from another node, you'll need to load the *RotoShape* node into the Viewer to see the fill.



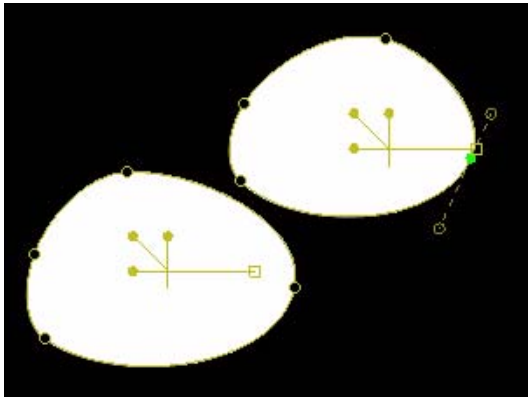
Important: You can only create filled shapes in the *RotoShape* node. To create single-point and open shapes, use the *Warper* or *Morpher* node.

A single *RotoShape* node can contain more than one shape.

To create multiple shapes in a single node:

1 To create another shape, click the Add Shapes button again.

2 Use the techniques described previously to create the additional shape.



3 When you're finished, click the first point you created to close the shape.

Each shape you create has its own transform control.

To duplicate a shape:

1. Click the Edit Shapes button to allow you to select shapes in the Viewer.
2. Move the pointer over the transform controls of the shape you want to duplicate, then right-click and choose Copy Shape from the shortcut menu.
3. Right-click in the Viewer, then choose Paste Shapes from the shortcut menu.

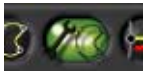
Editing Shapes

Once you've created a shape, there are several ways you can modify it by turning on the Edit Shapes button.

Note: When you're editing a *RotoShape* node containing multiple shapes that are very close to one another, it may be helpful to turn off the Enable/Disable Shape Transform Control button in the Viewer shelf. Doing so hides transform controls that may overlap the shape you're editing.

To edit a shape:

1. Click the right side of the *RotoShape* node you want to modify to load its parameters into the Parameters tab, its controls into the Viewer shelf, and its splines into the Viewer.
2. In the Viewer shelf, click the Edit Shapes button.



3. Select one or more points you want to edit by doing one of the following:
   - Click a single point to select it.
   - Shift-click additional points to add them to the selection.
   - Click in the Viewer and drag a selection box over all the points you want to select.
   - Hold the Shift key down and drag to use another selection box to add points to the selection.
   - Hold the Command or Control key down and drag to use another selection box to remove points from the selection.
   - Move the pointer over the edge, or the transform control, of a shape, and press Control-A or Command-A to select every point on that shape.

4 When the selected points are highlighted, rearrange them as necessary by doing one of the following:

- To move one or more selected points, drag them where you want them to go.
- To move one or more selected points using that shape's transform control, press the Shift key while you use the transform control.

  Note: Using the transform control without the Shift key pressed modifies the entire shape, regardless of how many points are selected. For more information on using the transform control, see page 553.

To add a point to a shape:

1 Click the Edit Shapes button.

2 Shift-click the part of the shape where you want to add a control point.

A new control point appears on the shape outline where you clicked.

To remove one or more points from a shape:

1 Select the point or points you want to remove.

2 Do one of the following:

- Click the Delete Knot button in the Viewer shelf.

- Press Delete.

Those points disappear, and the shape changes to conform to the remaining points.

To convert angled points to curves, and vice versa:

1 Select the point or points you want to convert.

2 Click the Spline/Line button to convert angled points to curves, or curves to angled points.

An optional step is to set the Show/Hide Tangents button to All or Pick to view tangents as they're created.

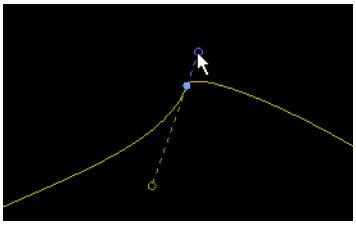
To change a curve by editing a point's tangent handles:

1 Make sure the Show/Hide Tangents button is set to All to view all tangents, or Pick to view only the tangents of points that you select.

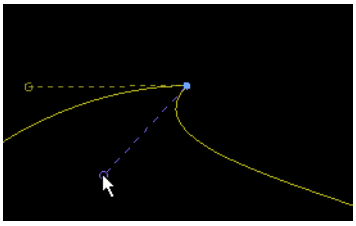
2 Make sure the Lock Tangents button is turned off.

3 Do one of the following:

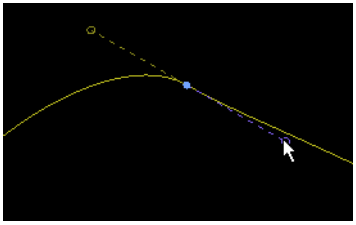
- To change the length of one of the tangent handles independently from the other, while keeping the angle of both handles locked relative to each other, drag a handle to lengthen or shorten it. You can also rotate both handles around the axis of the selected point.



- To change the angle of one of the tangent handles relative to the other, along with its length, press the Command or Control key while dragging a handle around the axis of the selected point. The selected tangent handle moves, but the opposing tangent handle remains stationary.



- To keep the angle of both tangent handles at 180 degrees relative to one another, keeping the lengths of each side of the tangent identical, press the Shift key while dragging either of the tangent handles around the axis of the selected point. If you Shift-drag tangent handles that were previously angled, they are reset.



To edit a shape using its transform control:

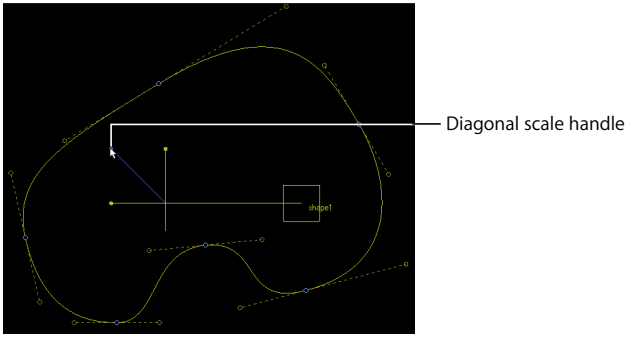
1. Make sure the Enable/Disable Shape Transform Control button is turned on.

   Each shape's transform control affects only that shape. For example, if a *RotoShape* node has three shapes in the Viewer, each of the three transform controls will only affect the shape its associated with. This is true even if you select control points on multiple shapes at once.

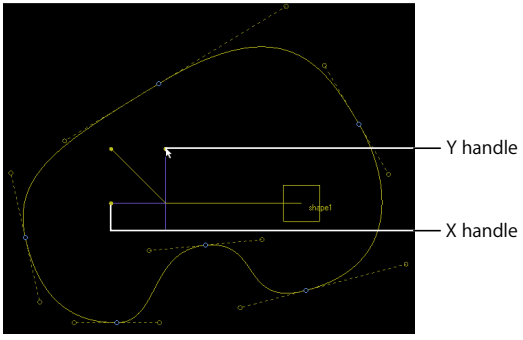
2. When you move, scale, or rotate a shape using its transform control, each transformation occurs relative to the position of the transform control. To move a shape's transform control in order to change the center point about which that shape's transformation occurs, press the Command or Control key while dragging the transform control to a new position.

3. To manipulate the shape, drag one of the transform control's handles:

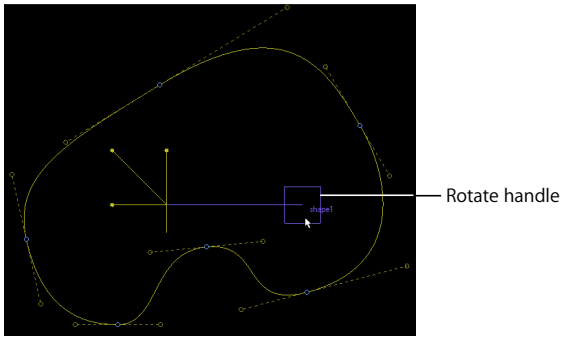
   - Drag the center of the transform control to move the entire shape in the Viewer. Both the X and Y handles will highlight to show you're adjusting the X and Y coordinates of the shape.
   - Drag the diagonal scale handle to resize the shape, maintaining its current aspect ratio.



   - Drag the X handle to resize the shape horizontally, or drag the Y handle to resize the shape vertically.



Drag the Rotate handle (to the right of the transform control) to rotate the shape about the axis of the transform control.



To edit selected control points using a shape's transform control:

1. Select one or more control points.
2. Hold down the Shift key while you manipulate one or more selected points with the transform control to modify only the selected points.

   Note: Using a shape's transform control without pressing the Shift key modifies the entire shape, regardless of how many points are selected.

To change the position of a shape's transform control:

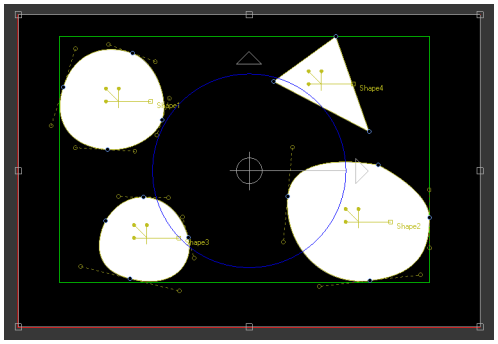
- Press the Command or Control key while you drag the center of a transform control to move it in relation to the shape it is associated with.

  This moves the center point around which shape transformations occur. For example, if you move the transform control of a shape to an area outside the shape itself, rotating the shape results in the shape moving around the new position of the transform control, instead of rotating in place.

To change the position of all shapes in a *RotoShape* node simultaneously:

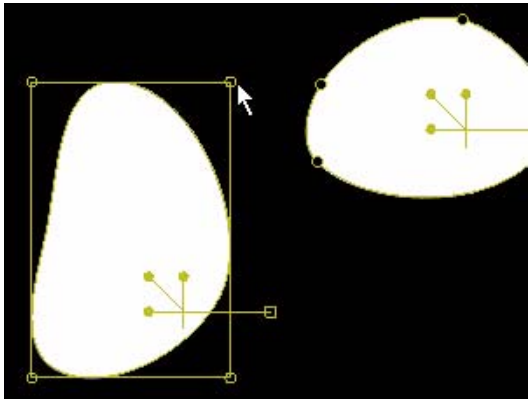
- Turn on the Enable/Disable Transform Control button.

A transform control that affects the entire node appears across the entire frame. Each shape's individual transform control remains visible.



Shape Bounding Boxes

Right-click a point, then choose Bounding Box Toggle from the shortcut menu to display a box around that shape that can be transformed to move and scale the shape. This works in addition to the shape's transform control, which appears at the center of the shape.



Changing a Shape's Color

You can change the color of individual shapes, to change their effect on the alpha channel they create. White shapes create solid areas, while black shapes create regions of transparency.

To change the color of a shape:

1. Right-click a shape's transform control, or any of its main control points.
2. Choose Black or White from the shortcut menu to change the shape to that color.

Reordering Shapes

You can reorder multiple overlapping shapes to change the effect they have on the alpha channel. For example, placing a black shape over a white shape lets you create a transparent area, while placing a white shape over a black shape creates a solid region.

To change the order of multiple shapes in the same *RotoShape* node:

1. Right-click a shape's transform control, or any of its main control points.
2. Choose one of the following options from the shortcut menu:
   - *Move to Back*: The selected shape is put behind all other shapes in that node.
   - *Move Back*: The selected shape is moved one level behind other shapes in that node.
   - *Move Forward*: The selected shape is moved one level in front of other shapes in that node.
   - *Move to Front*: The selected shape is moved in front of all other shapes in that node.

Showing and Hiding Individual Shapes

Each shape in a *RotoShape* node is labeled in the Viewer with a number based on the order in which it was created. You can use this information to show and hide individual shapes. Hidden rotoshapes aren't rendered.

To hide and show shapes, do one of the following:

- Right-click any shape in the Viewer to display a shortcut menu with commands to hide that shape, hide other shapes, or show all shapes.
- Right-click anywhere in the Viewer to display a shortcut menu that allows you to show or hide any shape in that node by its label.

Locking Tangents

When the Lock Tangents button is turned on, the tangent angles are locked when the control points are moved, rotated, or scaled.

When Unlock Tangents is selected, the tangent angles are unlocked. Select Unlock Tangents when moving, scaling, or rotating to maintain the shape.

Copying and Pasting Shapes Between Nodes

There are several nodes that use shapes besides the *RotoShape* node. These include:

- *LensWarp*
- *Morpher*
- *Warper*

Shapes can be copied and pasted between all of these nodes, so that a shape drawn in one can be used in any other. Animated shapes are copied along with all of their keyframes.

Note: You cannot copy shapes from *RotoShape* nodes that were created in Shake 3.5 or earlier.

To copy a single shape:

- Right-click the transform control, outline, or any point of the shape you want to copy, then do one of the following:
  - Choose Copy Shape from the shortcut menu.
  - Press Control-C.

To copy all visible shapes:

- Right-click anywhere in the Viewer, then choose Copy Visible Shapes from the shortcut menu.

Note: Since this command only copies visible shapes, you can turn the visibility off for any shapes you don't want to copy.

To paste one or more shapes into a compatible node, do one of the following:

- Right-click anywhere within the Viewer, then choose Paste Shapes from the shortcut menu.
- Move the pointer into the Viewer, then press Control-V.

Animating Shapes

Animating shapes created with a *RotoShape* node is a process of creating and modifying keyframes. When auto-keyframing is enabled, every change you make to a shape is represented by a single keyframe in the Time Bar, at the current position of the playhead.

To animate a rotoshape:

1. Click the right side of the *RotoShape* node to load its controls into the Viewer shelf.
2. Turn auto-keyframing on by clicking the Autokey button.

3. Move the playhead to the frame in the Time Bar where you want to change the shape.
4. Adjust the rotoshape using any of the methods described in this chapter.

   Each time you make an adjustment to a shape with auto-keyframing on, a keyframe is created at the current position of the playhead.

5 If necessary, move the playhead to another frame and continue making adjustments until you're finished.

6 When you're done, turn off auto-keyframing.

Rules for Keyframing

How keyframes are created and modified depends on two things: the current state of the Autokey button, and whether or not there's already a keyframe in the Time Bar at the current position of the playhead.

When animating shape changes, the following rules apply:

- When auto-keyframing is off and you adjust a shape that has no keyframes, you can freely adjust it at any frame, and all adjustments are applied to the entire duration of that node.
- When you adjust a shape that has at least one keyframe already applied, you must first turn auto-keyframing on before you can make further adjustments that add more keyframes.
- If auto-keyframing is off, you cannot adjust a shape at a frame that doesn't already have a keyframe. If you try to do so, the shape outline turns orange to let you know that the changes will not be permanent. However, you can still adjust a shape if the playhead is on an already existing keyframe.

  Note: If the playhead is not currently on a keyframe and you modify a *RotoShape* node while auto-keyframing is off, that change disappears when you move the playhead to another frame (the outline should be orange to indicate the temporary state of the change you've made). If you've made a change that you want to keep, turn auto-keyframing on before you move the playhead to add a keyframe to that frame.

Animating Single or Multiple Shapes

When a *RotoShape* node has multiple shapes, you can control whether or not animated changes you make to a single shape simultaneously keyframe every shape within that node. If you're careful about how you place your keyframes, this allows you to independently animate different shapes within the same *RotoShape* node without overlapping keyframes affecting the interpolation of a shape's animated transformation from one keyframe to another.

To set keyframes for only the current shape:

- Set Key Current Shape/All Shapes to Key Current Shape.

When this control is turned on, making a change to a single shape within a *RotoShape* node produces a keyframe that affects only that shape. Animation applied to any other shape is not affected by this new keyframe.

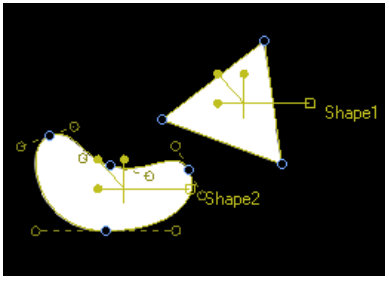
To set keyframes for all shapes:

- Toggle to Key All Shapes.

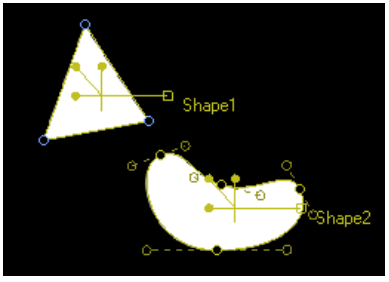
When this control is turned on, making a change to any single shape results in the state of all shapes within that *RotoShape* node being saved in the newly created keyframe.

Seeing the Correspondences Between Shapes and Keyframes

When you position the playhead in the Time Bar over a keyframe, all shapes that were animated within that keyframe appear with blue control points. Shapes that aren't keyframed remain at the current shape color, which is yellow by default.



Both shapes are keyframed.



Only the top shape is keyframed.

Cutting and Pasting RotoShape Keyframes

You can copy and paste rotoshape keyframes from one frame of the Time Bar to another. Whenever you copy a keyframe, you copy the entire state of that shape at that frame.

To copy a keyframe:

1. Move the playhead in the Time Bar to the frame where you want to copy the current state of the shape.
2. Right-click the transform control of the desired shape, then choose Copy Keyframe of Shape from the shortcut menu.

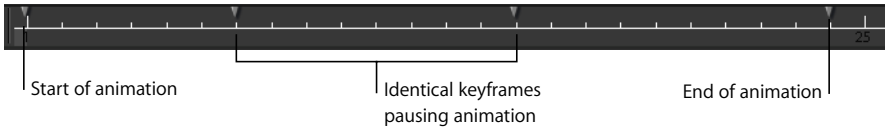
Note: You can copy the state of a shape at any frame, even if there is no keyframe there. Simply position the playhead anywhere within the Time Bar and use the Copy Keyframe command. That data can be pasted at any other frame as a keyframe.

To paste a keyframe:

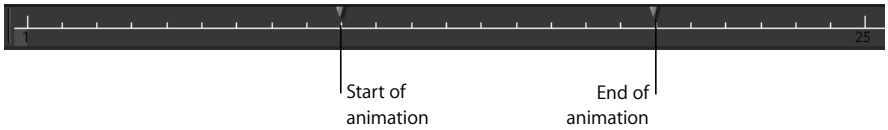
1. Move the playhead in the Time Bar to the frame where you want to paste the copied keyframe.
2. Right-click the transform control of the desired shape, then choose Paste Keyframe of Shape from the shortcut menu.

Adding Blank and Duplicate Keyframes to Pause Animation

If you want a shape to be still for a period of time before it begins to animate, insert a pair of identical keyframes at the start and end frame of the pause you want to create.



If you want to delay a shape's animation for several frames beyond the first frame, insert a keyframe with no animated changes at the frame you want the animation to begin, then modify the shape at the keyframe you want to animation to end.



To manually add a keyframe without modifying the shape:

- Click the Autokey button off and on.

A keyframe is created for the current state of the shape. If the shape is already animated, the state of the shape at the position of the playhead in the Time Bar will be stored in the new keyframe.

Shape Timing

Three parameters within the timing subtree of the *RotoShape* parameters allow you to modify when a rotoshape starts and ends. An additional retimeShapes control lets you retime all keyframes that have been applied to that *RotoShape* node, speeding up or slowing down the animation that affects the shapes within.

timeShift

Offsets the entire rotoshape, along with any keyframes that have been applied to it. This parameter corresponds to the position of that rotoshape in the Time View.

inPoint

Moves the in point of the rotoshape, allowing you to change where that rotoshape begins. This parameter corresponds to the in point of the rotoshape in the Time View.

outPoint
Moves the out point of the rotoshape, allowing you to change where that rotoshape ends. This parameter corresponds to the out point of the rotoshape in the Time View.

Retiming RotoShape Animation

The retimeShapes button, within the timing subtree of the *RotoShape* Parameters tab, lets you retime all of the keyframes that are applied to that rotoshape.

Using this command, you can compress the keyframes that are animating a rotoshape, speeding up the changes taking place from keyframe to keyframe, or expand them, slowing the animation down.

When you click retimeShapes, the Node Retime window appears. The retimeShapes command has two modes, Speed and Remap, which affect a *RotoShape* node's keyframes similarly to the Speed and Remap options found within the Timing tab of a *FileIn* node.

Speed Adjustments

The default Operation, Speed, lets you compress or expand all of the keyframes within the *RotoShape* node by a fixed multiplier.

Suppose you have the following group of keyframes:

Using the default Amount value of 2.0 and clicking apply contracts the keyframes proportionally—resulting in the following distribution:

invert
Turning on the Invert button expands the keyframes by the Amount value, instead of contracting them. This has an identical effect to setting Amount to a decimal value between 0 and 1.

Remap Adjustments

Setting Operation to Remap provides a way for you to use curve expressions to retime the current keyframe distribution. This lets you apply a retiming curve from a *FileIn* node to the keyframes of a shape that you've already animated to rotoscope that image.

Note: If you want to create a curve specifically to use with the Node Retime command, you can create a local variable within the *RotoShape* Parameters tab, load it into the Curve Editor, and create a curve expression which you can then copy and paste from the local variable into the Retime Expr field.

Attaching Trackers to Shapes and Points

You can attach preexisting *Tracker* nodes to a shape, or to any one of a shape's individual control points. Once a tracker is attached and you are happy with the result, you can bake the track into the shape's panX and panY parameters.

Attaching Trackers to Shapes

When you attach a tracker to a shape within a *RotoShape* node, the individual rotoshape control points are not changed as the shape moves along the tracked motion path. Furthermore, the offset between the position of the tracker and the original position of the shape is maintained as the shape follows the path of the tracker. You can attach separate trackers to separate shapes within the same *RotoShape* node.

To attach a track to an entire shape:

- In the Viewer, right-click the lower-center portion of the shape's transform controls and select an available track from the "Attach tracker to shape" list.

Note: You may have to click more than once for the correct menu to appear.

To remove the tracker from the shape:

- Right-click the lower-center portion of the shape's transform controls, then choose "Remove tracker reference" from the shortcut menu.

To bake the track into the shape:

- Right-click the lower-center portion of the shape's transform controls, then choose "Bake tracker into panX/panY" from the shortcut menu.

The selected track information is fed into the shape's panX and panY parameters.

Attaching Trackers to Individual Control Points

You can also attach multiple trackers to each of a shape's individual control points. You can attach as many trackers to as many separate control points as you like.

To attach an existing track to a single control point:

1. Select a shape control point in the Viewer.
2. Right-click the selected points, then choose a tracker from the "Attach tracker to selected CV's in current shape" shortcut submenu.

The selected tracker now animates the position of that control point. The offset between the position of the track and the original position of the shape control point is maintained as the point is animated along the path of the tracker.

You can also create a new *Tracker* node that is referenced by a specific control point.

To create a new *Tracker* node attached to one or more control points:

1. Select one or more points in the Viewer.
2. Right-click one of the selected points, then choose "Create tracker for selected points" from the shortcut menu.
3. Attach the new tracker to the image you want to track, and use the tracker's controls to track the desired feature.

The control point you selected in step 1 automatically inherits the tracking data.

To remove a tracker from one or more control points:

1 Select one or more shape control points in the Viewer.

2 Right-click one of the selected points, then choose "Remove tracker reference on selected points" from the shortcut menu.

Adjusting Shape Feathering Using the Point Modes

Each shape you create with a *RotoShape* node actually consists of two overlapping sets of points. The main shape points define the filled region of the shape itself, and a second set of edge points allows you to create a custom feathered edge. Since the shape edge and feathered edges are defined with two separate sets of points, you can create a feathered edge with a completely different shape.

The following table describes the four point modes that allow you to adjust the shape and feathered edges either together or independently of one another.

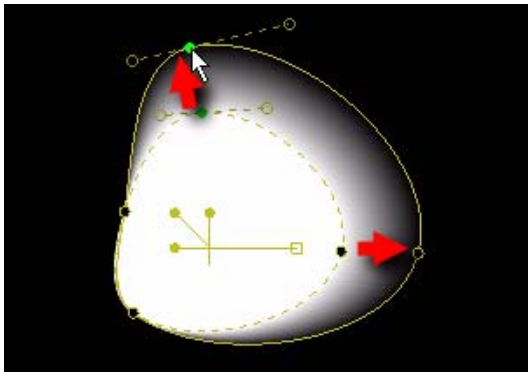
Button		Shortcut	Description
	Group Points	F1	Group Points mode lets you move the main shape point and its associated edge point together. Selecting or moving main shape points automatically moves the accompanying edge point. In this mode, edge points cannot be selected.
	Main Points	F2	Main Points mode only allows you to adjust the main shape points. Edge points cannot be modified.
	Edge Points	F3	Edge Points mode only allows you to adjust the edge points. Main shape points cannot be modified.
	Any Points	F4	Any Points mode lets you select and adjust either shape points or edge points independently of one another.

To create a soft edge around a shape:

1 Click the Edge Points button.

2 Select one or more points around the edge of the shape.

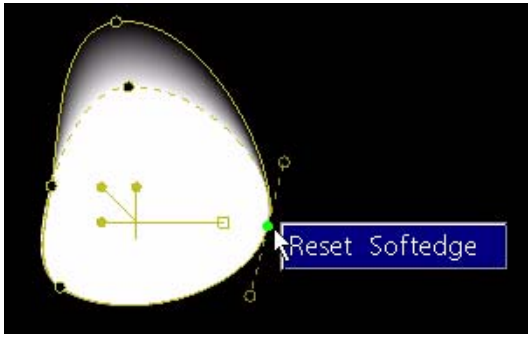
3 Drag the selected points out, away from the shape's edge. The farther you drag the edge, the softer it becomes.



To reset a soft edge segment to the default hard edge:

1 Click the Edge Points or Any Points controls.

2 Right-click the edge point you want to reset, then choose Reset Softedge from the shortcut menu.



To adjust both main and edge points at the same time:

1 Click the Group Points button.

2 Select one or more main shape points around the edge of the shape.

Note: In Group Points mode, you can neither select nor adjust edge points.

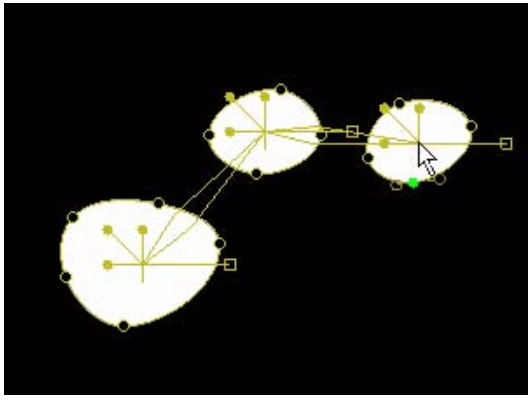
3 Adjust the selected main shape points.

The accompanying edge points are automatically adjusted to match your changes.

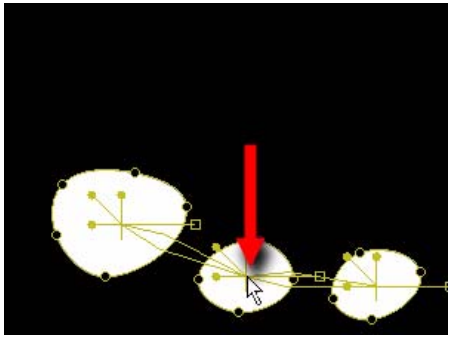
Important: Be careful with the soft edges—if you create a shape with overlapping lines, rendering artifacts may appear. To clean up minor artifacts, apply a slight blur using the *Blur* node.

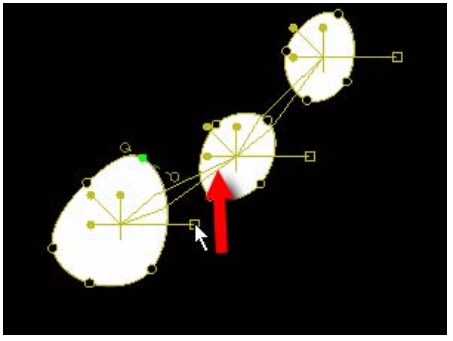
Linking Shapes Together

When you right-click the transform control, you can set up a skeleton relationship between your shapes. Right-click and choose Add Child from the shortcut menu, then click the transform control of the shape you want as a child of the current shape. To remove the link, right-click, then choose Remove Parent from the shortcut menu.



Once a link is established, modifying a shape affects its children.





Importing and Exporting Shape Data

Two controls let you import and export shape data between Shake and external applications. These controls are located in the Viewer shelf when editing *RotoShape*, *Warper*, or *Morpher* nodes.

To export shape data:

- Click the Import Shape Data button in the Viewer shelf, choose a name and destination for the export file in the File Browser, then click OK.

To import shape data:

- Click the Export Shape Data button in the Viewer shelf, choose a compatible shape data file using the File Browser, then click OK.

To support this new feature, a new shape data file format has been defined—named SSF (Shake Shape File). This format standardizes the manner in which shape data is stored by Shake for external use. Before shape data can be imported from an external application, it must first be converted into the SSF format. For more information on the SSF format, see the *Shake 4 SDK Manual*.

Right-Click Menu on Transform Control

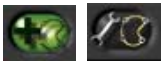
Menu Option	Description
Shape Visibility > Hide Shapes	Hides all shapes.
Shape Visibility > Hide Other Shapes	Hides all shapes except for the current one.
Shape Visibility > Show All Shapes	Turns the visibility of all shapes on.
Shape Visibility > List of all shapes	A list of every shape appears within this submenu. Choose a shape to toggle its visibility.
Bounding Box Toggle	Toggles the Bounding Box control for a shape on and off.
Arrange > Move to Back	Moves the shape behind all other shapes.
Arrange > Move Back	Moves the shape back one position in the shape order.
Arrange > Move Forward	Moves the shape forward one position in the shape order.
Arrange > Move to Front	Moves the shape in front of all other shapes.
Select All	Selects all points on the shape.
White	Renders the shape with a white interior.
Black	Renders the shape with a black interior and can therefore be made to punch holes in other shapes.

Menu Option	Description
Re-Center	Re-centers the transform tool to be the center of the shape. Control-drag to modify it without moving the shape.
Add Child	Click the transform tool of a second shape to make it a child of the current shape.
Remove Parent	Removes the current shape from the skeleton hierarchy.
Delete Shape	Deletes the current shape.
Copy Shape	Copies the current shape.
Copy Visible Shapes	Copies all visible shapes.
Paste Shapes	Pastes copied shapes.
Copy Keyframe of Shape	Copies the state of the shape at the position of the playhead.
Paste Keyframe of Shape	Pastes a copied shape keyframe.
Attach Tracker To Shape	Calls up a list of previously created trackers that may be used to transform the entire shape.

Right-Click Menu on Point

Menu Option	Description
Select All	Selects all points on the shape.
Reset Softedge	Repositions the edge knot on top of the main knot.
Remove tracker reference on selected points	Breaks the link between a tracker and the currently selected control points.
Bake tracker into selected points	Permanently bakes the transformation data from a referenced tracker into the control point.
Create tracker for selected points	Creates a new tracker that's automatically used to transform the selected control points.
Attach tracker to selected CVs in current shape	Lets you choose from all the trackers available in the current script, in order to attach a tracker to one or more selected control points.

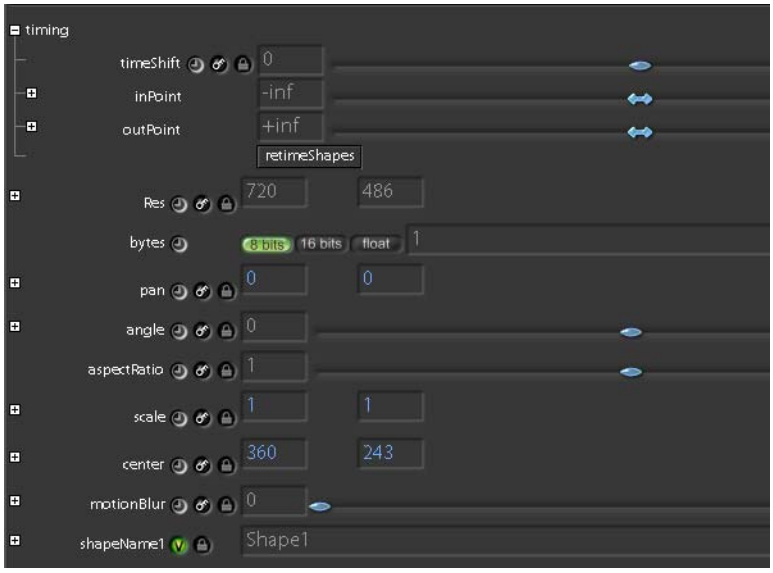
Viewer Shelf Controls

Button		Description
	Add/Edit Shapes	Click the Add Shapes button to draw a shape. Click the Edit Shapes button to edit a shape. Closing a shape automatically activates Edit Shapes mode. Rotoshapes render only when Edit Shapes mode is active.
	Import/Export Shape Data	Lets you import and export shape data between Shake and external applications.

Button		Description
	Fill/No Fill	Controls whether or not the shape is filled.
	Show/Hide Tangents	Controls the tangent visibility. In Pick mode, only the active point displays a tangent. None hides all tangents, and All displays all tangents.
	Lock/Unlock Tangents	When Lock Tangents is on, the tangent angles are locked when control points are moved, rotated, or scaled. When Unlock Tangents is on, the tangent angles are unlocked.
	Spline/Linear Mode	New points are created as splines or as linear points. Select a point and toggle this button to specify its type.
	Enable/Disable Transform Control	Enables or disables the onscreen transform control used pan the entire collection of shapes. The default setting is Hide.
	Delete Control Point	Deletes currently selected knot/control point(s).
	Points Modes	Determines what points can be selected. Use Group Points mode to select the main shape and the edge points. Use Main Points mode to select only the main shape points. Use Edge Points mode to select only edge points, and use Any Points mode to pick either main or edge points.
	Key Current/Key All Shapes	Key Current Shape/Key All Shapes: When Autokey is enabled (when animating), select Key Current Shape to keyframe only the current rotoshape. Select Key All Shapes to keyframe all rotoshapes.
	Enable/Disable Shape Transform Control	Lets you show or hide the transform control at the center of each shape. Hiding these controls will prevent you from accidentally transforming shapes while making adjustments to control points.
	Path Display	If the main onscreen transform tool is turned on, this button toggles the visibility of the animation path.

RotoShape Node Parameters

The *RotoShape* node has the following controls:



timing

Three parameters within the timing subtree of the *RotoShape* parameters allow you to modify when a rotoshape starts and ends. An additional retimeShapes control lets you retime all keyframes that have been applied to that *RotoShape* node.

timeShift

Offsets the entire rotoshape, along with any keyframes that have been applied to it. This parameter corresponds to the position of that rotoshape in the Time View.

inPoint

Moves the in point of the rotoshape, allowing you to change where that rotoshape begins. This parameter corresponds to the in point of the rotoshape in the Time View.

outPoint

Moves the out point of the rotoshape, allowing you to change where that rotoshape ends. This parameter corresponds to the out point of the rotoshape in the Time View.

retimeShapes

The retimeShapes control, within the timing subtree of a rotoshape's parameters, lets you retime all of the keyframes that are applied to that rotoshape. Using this command, you can compress the keyframes that are animating a rotoshape, speeding up the changes taking place from keyframe to keyframe, or expand them, slowing the animation down. When you click retimeShapes, the Node Retime window appears.

For more information on using the retimeShapes command, see "Retiming RotoShape Animation" on page 561.

Res
The Width and Height of the *RotoShape* node's DOD.

bytes
The bit depth of the image created by the *RotoShape* node. You can specify 8-bit, 16-bit, or float.

pan
A global pan applied to the entire image.

angle
A global rotation applied to the entire shape—points are properly interpolated according to the rotation.

aspectRatio
This parameter inherits the current value of the defaultAspect global parameter. If you're working on a nonsquare pixel or anamorphic image, it is important that this value is set correctly to avoid unwanted distortion in the image.

scale
A global scale applied to the entire image.

center
The center of transformation for the angle and x/yScale parameters.

motionBlur
Motion Blur quality level. 0 is no blur, whereas 1 represents standard filtering. For more speed, use less than 1. This value is multiplied by the Global Parameter motionBlur.

shutterTiming
A subparameter of motionBlur. Shutter length. 0 is no blur, whereas 1 represents a whole frame of blur. Note that standard camera blur is 180 degrees, or a value of .5. This value is multiplied by the global parameter, shutterTiming.

shutterOffset
A subparameter of motionBlur. This is the offset from the current frame at which the blur is calculated. Default is 0; previous frames are less than 0.

Using the QuickShape Node

The *QuickShape* node is an image generator to be used for animated garbage mattes. It is ideal for plugging into the Mask input of a node, or is used in conjunction with nodes such as *Inside*, *Outside*, or *KeyMix*.

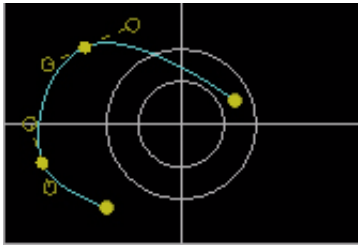
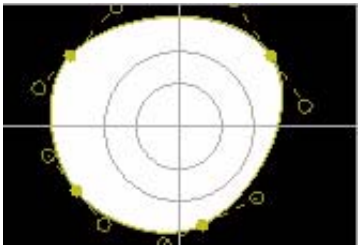
Note: The *QuickShape* node is an older node to create rotoshapes. The more flexible (and faster) *RotoShape* node is recommended. This node is maintained for compatibility purposes. The one advantage *QuickShape* has over *RotoShape* is its ability to propagate keyframe changes to other keyframes before or after the current frame.

Since these nodes create images like any node, you can modify the images with other nodes such as *Blur* or *DilateErode*.

You can enable motion blur for an animated *QuickShape*. Unfortunately, the shape does not use Shake's normal high-quality motion blur. It instead draws and renders several versions of the entire shape, so temporal aliasing occurs with extreme motion.

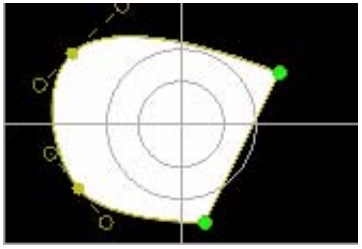
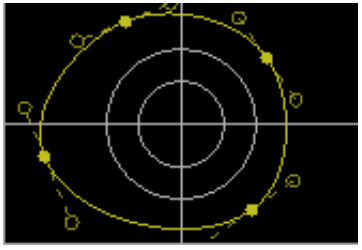
Creating QuickShapes

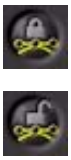
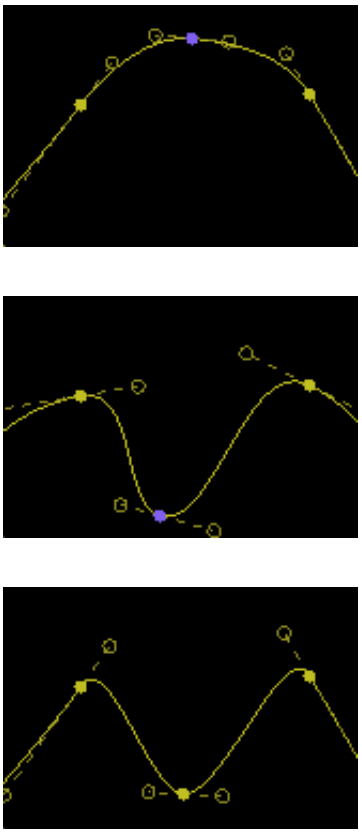
When you create a *QuickShape* node and Display Onscreen Controls is enabled for the Viewer, you can immediately add points to the shape.

Button	Usage	Example
	Start in Build mode. In Build mode, each time you click a blank spot, you append a new point between the last point and the first point. Click the points, or, if you hold down the mouse button, you can drag the new point around. You can also go back and change any key or tangent, or insert a point by clicking a segment.	
	Once you are finished with the rough shape, switch to Edit mode (click the Build/Edit Mode button). When you click a blank spot, you do not append a new point; instead, you can drag to select several points and move them as a group. This also fills in the shape.	

Modifying QuickShapes

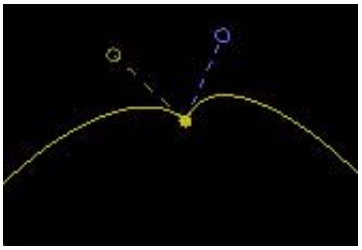
To select multiple points in Edit mode, drag to select the desired points. The selected points can then be modified as a group.

Button	Usage	Example
	The Spline/Line Mode buttons change the selected points from Linear to Smooth. Select the points and toggle the button to the setting you want. In this example, the two right points have been made Linear. When Linear is selected, no tangents are available.	
	Click the Fill button on the Viewer toolbar to turn the shape fill on and off. In Build mode, the shape is not filled. The filled shape is not just a display feature—it affects the composite.	

Button	Usage	Example
	The Lock Tangents button locks or unlocks the tangents of adjacent points when moving any point. In the first example, the tangents are unlocked. Therefore, the middle blue point is moved down. Shake tries to keep the tangents of the adjacent points smooth, and therefore moves the tangents. If Lock Tangents is on, the adjacent tangents stay locked in place. This provides accuracy for adjacent segments, but creates a more irregular shape.	
	The Show/Hide Tangents button displays or hides the tangents on the shape.	
	The Enable/Disable Transform Control button turns on and off the display of the transform tool for the *QuickShape*.	
	The Delete Control Point button deletes all selected points.	

To break a tangent:

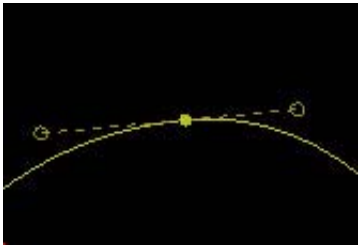
- Control-click the tangent.



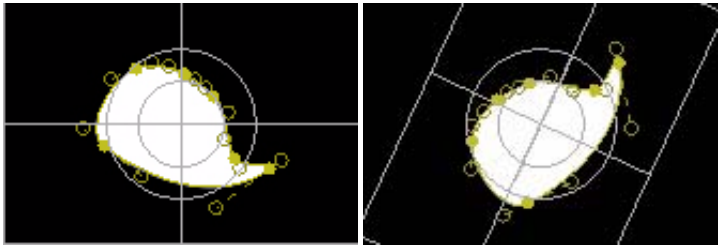
Note: No tangents are available when the points are set to Linear mode.

To reconnect the tangents:

- Shift-click the broken tangent.

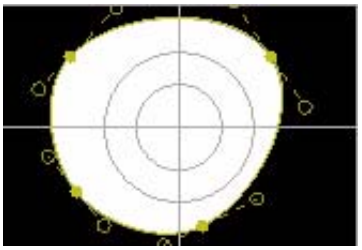
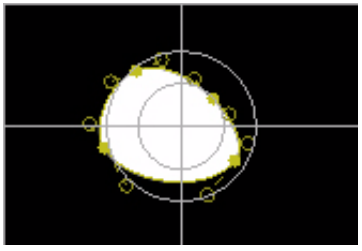


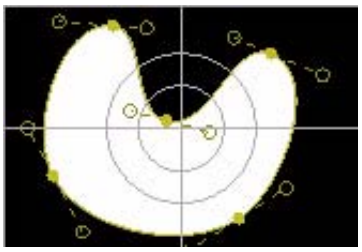
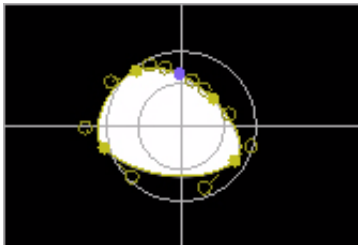
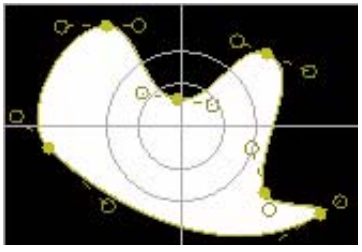
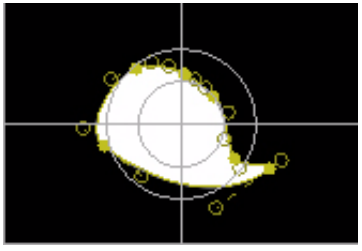
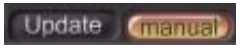
Use the transform tool to modify the entire shape. The transform tool includes pan, rotation, and scaling tools for the shape. Since this is a transformation, the points rotate properly in an angular fashion when interpolating in an animation, rather than just sliding linearly to the next position. The controls appear at the same resolution of the *QuickShape* node, so if you are dealing with 2K plates, you may want to enter a larger resolution for the *QuickShape*. Similarly, if you find the transform tool annoying, enter a resolution of 10 x 10. Neither of these techniques changes rendering speed due to the Infinite Workspace.



Animating QuickShapes

The following table discusses the *QuickShape* animation tools.

Button	Usage	Example
	To easily animate the *QuickShape*, enable Autokey and move the points. To enter a new keyframe, move to a new time, and change the shape's position (or the control points of the shape). In this example, the shape is smaller on the second keyframe. As you drag the playhead, the shape interpolates between the two keyframes.	 
	Delete a keyframe (if present) at the current frame.	

Button	Usage	Example
 	Here, a point is inserted and moved toward the center at the first keyframe. At the second keyframe's position, the shape is still round because Shake has maintained the smooth quality of the segment. If you instead turn on the Propagate buttons when you modify a point, the second keyframe's point position is also modified. For example, go back to keyframe 1, enable Propagate Forward, and insert a new point, dragging it outward. Jump to the second keyframe, and the new point is positioned in a relatively similar fashion in the second keyframe. If you have several keys, Propagate Forward or Backward can slow down your interactivity.	   
	To quickly scrub through an animation, toggle to Release or Manual Update mode (in the upper-right corner of the interface), and then move the playhead. The shapes draw in real time, but are not rasterized.	 

QuickShape Node Parameters

The following table lists the *QuickShape* parameters.

Parameters

This node displays the following controls in the Parameters tab:

width, height
The overall width and height of the frame in which the rotoshape is drawn. Defines the DOD. These parameters default to the expressions *GetDefaultWidth()* and *GetDefaultHeight()*.

bytes
Bit depth, 1, 2, or 4 bytes/channel.

xPan, yPan
A global pan applied to the entire shape.

angle
The center of transformation for the angle and x/yScale parameters.

aspectRatio
This parameter inherits the current value of the defaultAspect global parameter. If you're working on a nonsquare pixel or anamorphic image, it is important that this value is set correctly to avoid unwanted distortion in the image.

xScale, yScale
A global scale applied to the entire shape.

xCenter, yCenter
The center of transformation for the angle and x/yScale parameters.

motionBlur
Motion Blur quality level. 0 is no blur, whereas 1 represents standard filtering. For more speed, use less than 1. This value is multiplied by the Global Parameter motionBlur.

shutterTiming
A subparameter of motionBlur. Shutter length. 0 is no blur, whereas 1 represents a whole frame of blur. Note that standard camera blur is 180 degrees, or a value of .5. This value is multiplied by the Global Parameter shutterTiming.

shutterOffset
A subparameter of motionBlur. This is the offset from the current frame at which the blur is calculated. Default is 0; previous frames are less than 0.

Paint

21

Shake provides simple paint capabilities using the QuickPaint node. This chapter describes how to use the non-destructive tools found within this node to make fast fixes to your image sequences.

About the QuickPaint Node

The *QuickPaint* node is a touch-up tool to fix small element problems such as holes in mattes or scratches/dirt on your plates. It is a procedural paint tool, which allows you to change strokes long after they've been made. This helps to emphasize its key feature—it is simply another compositing tool that can easily be used in conjunction with any of Shake's other nodes. You can apply an effect and easily ignore it, remove it, or reorder it after you have applied your paint strokes.

The tools within the *QuickPaint* node respond to the pressure sensitivity found in most pen-based digitizing tablets.

Connecting Input Images to the QuickPaint Node

The *QuickPaint* node has two inputs. The first one lets you connect a background image to paint on, and also acts as the clone source. The second input is used by the Reveal tool—Reveal paint strokes expose the image that's connected to the second input (for example, a clean background plate), allowing you to replace portions of the first image with portions of the second image.

Setting the QuickPaint Node's Resolution

You can apply a *QuickPaint* node to another node, or you can create an unattached "floating" *QuickPaint* node that can be composited later with other nodes using one of the Layer functions, or used as a mask operator. *QuickPaint* nodes that are attached to other nodes assume the resolution of the node tree.

Floating *QuickPaint* nodes inherit the defaultWidth and defaultHeight of the script. To change the resolution of a floating *QuickPaint* node, create a *Color* or *Window* node, and attach the *QuickPaint* node underneath. Setting the resolution in the *Color* or *Window* node parameters will then determine the resolution of the *QuickPaint* node.

Note: In the *Color* node, the alpha channel is set to 1 by default.

It's important to make sure the resolution of the *QuickPaint* node is properly set, because you cannot paint beyond the boundaries of the frame.

Toggling Between Paint and Edit Mode

The *QuickPaint* node has two modes of operation. In Paint mode, you can create new brush strokes. In Edit mode, you can modify any previously created paint stroke.

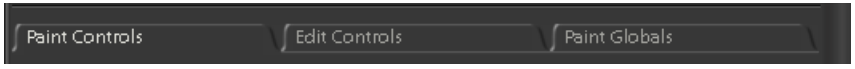
When the *QuickPaint* node is active, its associated tools appear in the Viewer shelf. Three subtabs—Paint Controls, Edit Controls, and Paint Globals—appear in the Parameters tab.

The first button on the Viewer shelf is the Paint/Edit mode toggle.

To toggle between Paint and Edit mode, do one of the following:

- Click the Paint/Edit button.

- Click either the Paint Controls or Edit Controls subtab in the Parameters window to toggle to the Paint or Edit mode.



Paint Tools and Brush Controls

Using the other controls in the Viewer shelf and Paint Controls tab, you can modify the paint characteristics of new strokes (color, size, brush type, opacity, softness).

There are five basic brush types. One modifier changes the drop-off of the five brush types. To use a brush, make sure that you are in Paint mode (click the Paint/Edit button or select a brush), then paint.

To change the size of a selected brush:

- Control-drag in the Viewer. You can also numerically set the brush size in the Paint Controls tab.

To draw a straight line:

- Hold down the Shift key while drawing in the Viewer.

The following table contains the basic brush tools.

Button		Description
	Hard/Soft	When soft is selected, paints any brush type with a soft falloff. When hard is selected, paints any brush type with a hard falloff. You can also press F11 to toggle between the soft and hard falloff. This is not a brush—it just modifies other brushes.
	Paint Brush	Applies RGBA color to the first input.
	Smudge	Smears the pixels. Smudge should always use the hard-falloff setting.
	Eraser	Erases previously applied paint strokes only. Does not affect the background image.
	Reveal Brush	Reveals the image connected to the second node input. If there is no input, the Reveal Brush acts as an *Outside* node, and punches a hole through the paint and the first input source.
	Clone Brush	Copies areas from the first image input, as well as paint strokes created in the *QuickPaint* node. To move the brush target relative to the source, Shift-drag.

Press F9 to select your last-used brush type. With this key command, you can quickly toggle between the last two brush types you selected.

Picking a Paint Color

There are several ways to pick your paint color and opacity.

The Color control in the Paint Controls tab gives you access to the standard color tools—the Color Picker, the Virtual Color Picker, or point-and-click color selection in the Viewer. With the pointer in the Viewer, you can also press F10 or P to open the Color Picker.

The Color control in the Viewer shelf indicates the current color. It works like any other color control in Shake.

When you paint, each stroke is unpremultiplied. As a result, adjusting the alpha slider in the Parameters tab does not affect what you apply to the RGB channels. However, changing opacity affects all four channels.

Other Viewer Shelf Controls

The *QuickPaint* node has the following Viewer shelf controls:

Button		Description
	Active Channels	These buttons indicate which channels are being painted on. For example, to touch up the alpha channel only, turn off the RGB channels.
	Frame Mode	When Frame mode is selected, you only paint on the current frame.
	Interp Mode	When Interp (Interpolate) mode is selected, brush strokes are animated using interpolation from one frame to the next. For example, paint a stroke on frame 1, and then go to frame 20, and paint a stroke on frame 20. When you scrub between 1 and 20, the stroke interpolates. Beyond frame 20 or before frame 1, the image is black. To insert a second interpolation stroke, click the Interp toggle until Interp is selected again, and use the strokeIndex slider in the Edit Controls tab to select the stroke you wish to modify.
	Persist Mode	When Persist (Persistent) mode is selected, the stroke persists from frame to frame. The stroke does not change unless you switch to Edit mode and animate the stroke.
	History Step	Use the History Step buttons to step backward or forward through your history. Using these buttons activates Edit mode. As you step backward, the strokeIndex parameter in the Edit Controls tab indicates the current stroke number. Although you can edit any brush at any time, this parameter sets the point at which you are evaluating your paint. You can, for example, step back several strokes, insert a new stroke, and then step forward. The later strokes are placed on top of your new stroke because the new stroke is earlier in the step history.
	Magnet Drag Mode	When Magnet Drag mode is enabled, and you are in Edit mode, you can select a group of points on a stroke. If you click near the middle of the points and drag, the points near the selected point are dragged more than points farther away. You can also press and hold Z to temporarily activate this mode if you are in Linear Drag mode.
	Linear Drag Mode	Click the Magnet Drag button to toggle to Linear Drag mode. When in Edit mode and you drag a group of points, they all move the same amount.

Button		Description
	Erase Last Stroke	By default, removes the last stroke you made. If you select another stroke with the History Step controls or strokeIndex parameter, this control deletes the currently selected stroke.
	Clear Canvas	Removes all strokes from all frames of your canvas.

Modifying Paint Strokes

In Edit mode, you can select any stroke by clicking its path. You can also adjust the strokeIndex slider back and forth to expose previous strokes numerically.

Once you've selected a stroke, you can use the parameters in the Edit Controls tab to modify its characteristics, changing the tool, softness, stroke mode, color, alpha channel, brushSize, and all other parameters long after its creation.

You can animate strokes with the Interpolation or Frame setting, or you can modify any stroke after it has been created with the Edit mode. For more information on converting paint stroke modes, see "Converting Paint Stroke Types" on page 589.

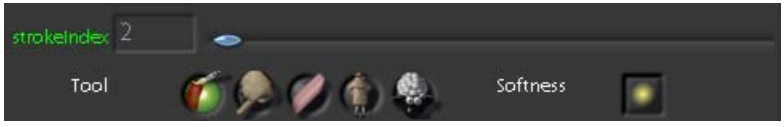
In Edit mode, you can select a stroke in one of three ways:

- Click the stroke.

  Its control points and path appear.

- Use the strokeIndex parameter in the Edit Controls tab.

  Each stroke is assigned a number, and can be accessed by using the strokeIndex slider.



- Use the History Step buttons.

This not only selects the stroke, but only renders existing strokes up to the current frame. Even though there may be later strokes in the history list, they are not drawn until you move forward using the History Step button.

To add to your selected points on a paint stroke:

- Press Shift and drag a selection box over the new points.

To remove points from the current selection:

- Control-drag to remove control points from the current selection.

To drag-select multiple control points:

1. Move the pointer over the stroke you want to edit.
2. Drag to select that stroke's control points.
3. Edit the points as necessary.

This behavior applies to *QuickPaint*, *RotoShape*, and *QuickShape* objects. For example, you can display the onscreen controls for the shapes of two different *QuickPaint* nodes by loading the parameters of one node into the Parameters1 tab, then Shift-clicking the right side of the second node to load its parameters into the Parameters2 tab. Also, if the points you want to drag-select are within a DOD bounding box, move the pointer over the shape inside the DOD, then drag to select the points.

Deleting Strokes

There are three ways you can delete paint strokes.

To delete the last stroke you created:

- Click the Erase Last Stroke button in the Viewer shelf.

To delete all paint strokes you've made on every frame:

- Click the Clear Canvas button in the Viewer shelf.

To delete one or more strokes you made earlier:

1. Change to Edit mode by doing one of the following:
   - Click the Edit button in the Viewer shelf.
   - Click the Edit Controls tab in the Parameters tab.
2. Move the playhead to the frame with the stroke you want to delete.
3. Select a stroke to delete by doing one of the following:
   - Click the History Step button in the Viewer shelf.
   - Move the stokeIndex slider in the Edit Controls tab.
4. When the stroke you want to delete is highlighted in the Viewer, click the Erase Last Stroke button in the Viewer shelf.

Editing Stroke Shape

Once you've selected one or more points, you can drag them to a new position to change the shape of the paint stroke.

To insert a new point:

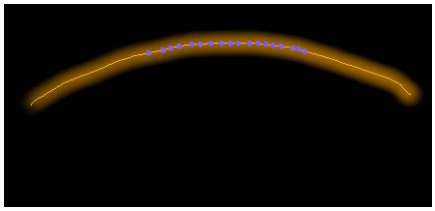
- Shift-click a segment of the stroke.

To remove a point:

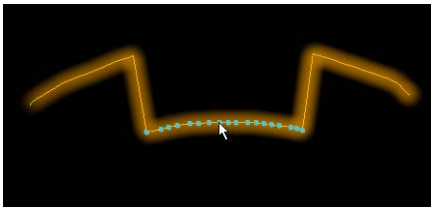
- Select the point, then press Delete.

There are two different drag modes that affect how strokes are reshaped when you move a selected group of control points.


- If Linear Drag mode is selected, all selected control points move the same amount.

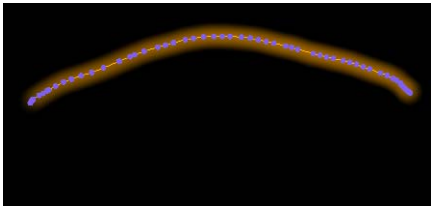


Selected points

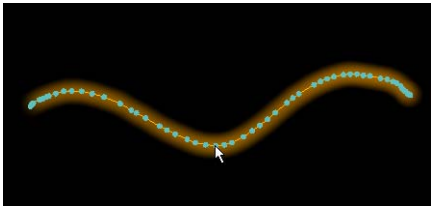


Dragging in Linear Drag mode

- If Magnet Drag mode is selected, the points nearest the pointer move the most. To temporarily activate this mode, press and hold the Z key and drag.



Selected points



Dragging in Magnet Drag mode

Animating Strokes

There are several ways you can animate paint strokes:

- You can keyframe the movement of selected points using the standard Autokey controls to set keyframes for paint strokes in Interp mode.
- You can animate the startPoint and endPoint parameters to animate the completion of a stroke along the defined stroke path.
- You can attach a tracker to a paint stroke.

Attaching a Tracker to a Paint Stroke

In Edit mode, a preexisting track can be attached to a paint stroke.

To attach a track to a paint stroke:

1. Make sure the paint stroke is a persistent stroke.

   Note: For information on converting a paint stroke from Frame to Persist mode, see "Converting Paint Stroke Types" on page 589. You can also convert the stroke after the track is attached.

2. In the Viewer, right-click the selected stroke, then choose an available track from the "Add tracker to stroke" shortcut menu.

   Note: Although only a single control point appears on the paint stroke when attached to the track, the track is applied to the entire paint stroke. You cannot apply a track to individual control points on a paint stroke.

   The selected track information is then fed into the stroke's panX and panY parameters as an offset.

   Once a tracker is attached to a paint stroke, the track information is displayed in the Viewer on the paint stroke, as well as in the Edit Controls tab, next to the Convert Stroke button. The track information is only displayed if that stroke is selected in the strokeIndex or in the Viewer.

   To remove the track, right-click the paint stroke in the Viewer, then choose "Remove tracker reference" from the shortcut menu.

   Once the paint stroke is attached to the track and you have achieved the result you want, you can "bake" the tracker information into the stroke.

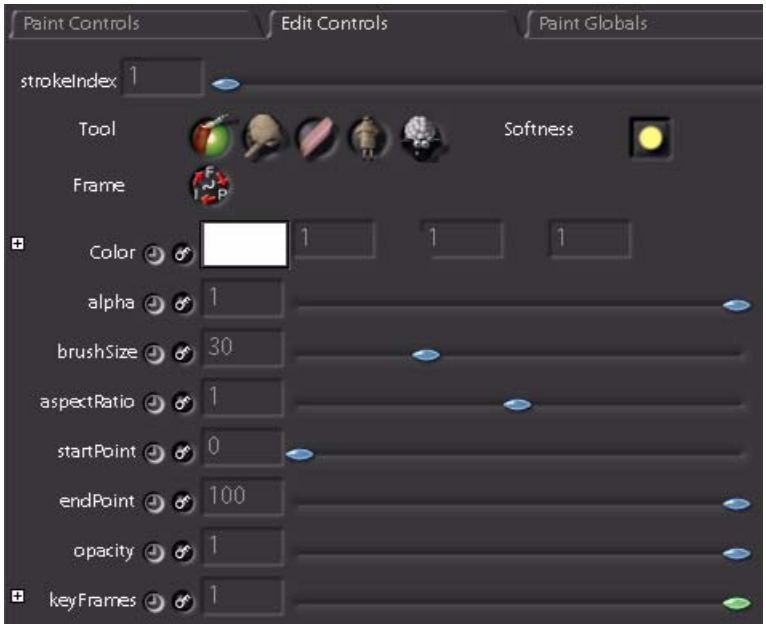
To bake the track:

1. Make surethat Edit mode is on.
2. Select the stroke (click the stroke in the Viewer or select the stroke in the strokeIndex slider in the Edit Controls tab).
3. Right-click the paint stroke, then choose "Bake tracker into paint stroke" from the shortcut menu.

   The keyframes are applied to the paint stroke, and the track information no longer appears in the Edit Controls subtab.

Modifying Paint Stroke Parameters

You can also use the Edit Controls tab in the *QuickPaint* parameters to modify your strokes.



In the Edit Controls tab, click a brush type in the Tool row to switch brush types. You can also click the Hard/Soft button to switch between a hard and soft falloff, or change the stroke type with the Convert Stroke button. Additionally, you can alter or animate the color, alpha, opacity, brushSize, or aspectRatio parameters of the current stroke.

The startPoint parameter determines the point at which the stroke begins, measured as a percentage offset from the beginning of the stroke's path. For example, a startPoint value of 50 displays a stroke that is half the length of its invisible path. The endPoint parameter works the same way, but from the other end of the stroke. You can animate a stroke onscreen, creating a handwriting effect by setting keyframes for the endPoint from 0 to 100 over several frames. All stroke types can be animated in this way. Remember that the startPoint and endPoint parameters describe a percentage of the path. Therefore, if you change control point positions relative to each other, you may introduce unwanted fluctuations in the line-drawing animation.

The keyFrames parameter (in the Paint Controls subtab) is a placeholder so that keyframe markers appear in the Time Bar—it has no other interactive function.

Interpolating Paint Strokes

In the following example, frame 1 contains three separate paint strokes, and frame 50 also contains three separate paint strokes. Interpolate the second paint stroke on frame 1 (the number "2") with the second paint stroke on frame 50 (the number "5").

To interpolate paint strokes from one shape to another:

1 In the Viewer shelf, ensure that Paint mode is enabled.

2 Ensure that Frame mode is enabled.

3 At frame 1, draw three paint strokes.

4 At frame 50, draw three more paint strokes.

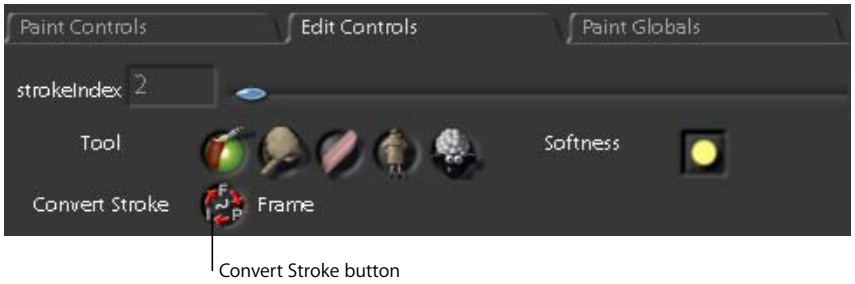
Note: In the above illustrations, each number is a single paint stroke.

5 In the Viewer shelf, click the Paint mode button to toggle to Edit mode.

Note: You can also click the Edit Controls tab in the *QuickPaint* Parameters tab.

6 In the Edit Controls tab, select stroke 2 from the strokeIndex.

7 Click the Convert Stroke button.



The Convert Stroke window opens.

8 In the Convert Stroke window, enable Interp if it is not already enabled.

9 Enter "2, 5" in the Stroke Range field.

This instructs Shake to combine paint strokes 2 and 5 into one interpolated stroke.

Note: Because there is more than one paint stroke on a frame, the comma syntax must be used for interpolation. If frame 1 contained only one paint stroke, and frame 50 contained only one paint stroke, and you wanted to interpolate the two strokes, you could enter "1-2" or "1, 2" in the Stroke Range of the Convert Stroke window to interpolate between paint stroke 1 and paint stroke 2 in the node.

As another example, if you wanted to interpolate between a stroke on frame 1 (stroke 1), a stroke on frame 5 (stroke 2), a stroke on frame 10 (stroke 3), and a stroke on frame 15 (stroke 4), enter "1, 2, 3, 4" to interpolate between all strokes.

10 Click OK.

Scrub between frames 1 and 50, and notice that the 2 (the second paint stroke in the node) and the 5 (the fifth paint stroke in the node) interpolate.

Because Frame mode was enabled when you drew the paint strokes, the strokes that are not interpolated (the numbers 1, 3, 4, and 6) exist only at the frame in which they were drawn (frames 1 and 50).

Converting Paint Stroke Types

Normally, to paint a stroke that exists in all frames, you select Persist mode in the Viewer shelf before you draw your strokes. In case you forget this step and draw your strokes in the default Frame mode, you can use the Convert Stroke feature to convert a paint stroke, or multiple paint strokes, to Persist mode.

To convert paint strokes from Frame to Persist mode:

1. Once the paint stroke is drawn (in Frame mode), click the Edit mode button in the Viewer shelf.
2. In the Edit Controls tab, select the stroke in the strokeIndex.

   Note: You can change the selected stroke later in the Convert Stroke window.
3. Click the Convert Stroke button.
4. In the Convert Stroke window, enable Persist.

   To convert multiple strokes, do one of the following:

   - To convert all paint strokes from Frame to Persist, enter the whole stroke range in the "Convert stroke(s)" value field. For example, if you had a total of 12 strokes, enter "1-12" to convert all 12 strokes to Persist mode.
   - To convert selected strokes, enter the desired range in the "Convert stroke(s)" value field. For example, enter "3-8, 11" to convert paint strokes 3 through 8 and stroke 11 (of the 12 total strokes).

     Note: Enter a frame range using standard Shake syntax ("1-100," "20-50x3," and so on.).
   - To convert a single stroke, enter the stroke number in the "Convert stroke(s)" field.
5. Click OK.

   The strokes are converted from Frame to Persist mode.
   You can also convert a persistent paint stroke to appear only within a specific frame range.

To convert paint strokes from Persist to Frame mode:

1. Once the paint strokes are drawn (in Persist mode), click the Edit mode button in the Viewer shelf.
2. In the Edit Controls tab, use the strokeIndex slider to select the desired stroke.
3. Click the Convert Stroke button.

   In the Convert Stroke window, the message "Convert Stroke from 'Persist' to 'Frame'" and the Frame Range field appear.
4. Enter the frame range for the paint stroke.

   For example, to draw the stroke on frames 1 and 3, and from frames 10 to 20, enter "1, 3, 10-20."
5. Click OK.

   The converted stroke appears on frames 1, 3 and 10 through 20.

QuickPaint Hot Keys

The following table lists the *QuickPaint* node hot keys.

Key	Function
F9	Use last brush.
F10 or P	Pick color.
F11	Toggle between hard/soft brush.
Z	Magnet drag in Edit mode.

Note: In Mac OS X, Exposé is mapped to F9-F12 by default. To use these keys in Shake, disable the Exposé keyboard shortcuts in System Preferences.

QuickPaint Parameters

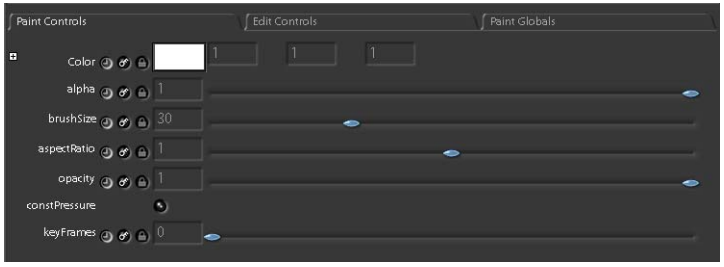
The following section lists the *QuickPaint* node parameters.

Note: The *QuickPaint* node should not be used inside of macros.

The controls in the *QuickPaint* node are divided among three nested tabs within the Parameters tab:

Paint Controls

The parameters in this tab control how new strokes are drawn.



Color
The color of the current paint stroke.

alpha
A slider that lets you change the alpha channel of the current paint stroke. This does not modify its color, as the strokes are not premultiplied.

brushSize
The size of the brush. You can also use Control-drag in the Viewer to set the brush size.

aspectRatio
Aspect ratio of the circular strokes.

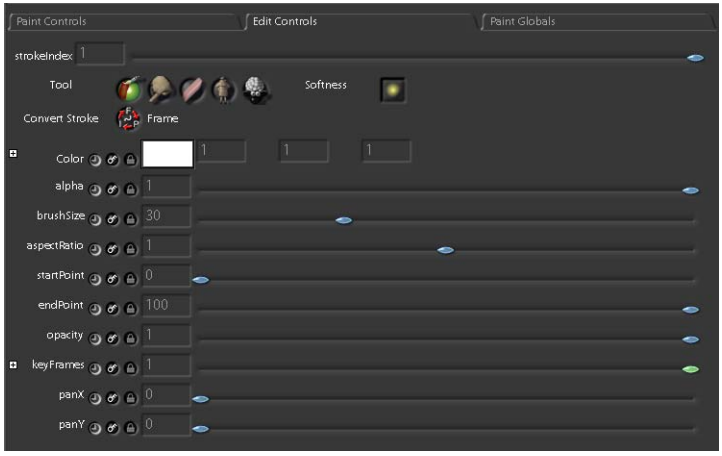
opacity
A fade value applied to the R, G, B, and A channels.

constPressure
When this parameter is turned on, the digital graphics tablet's stylus pressure is ignored.

keyFrames
This parameter has no purpose except as a placeholder to contain keyframe markers in the Time Bar. It is not modifiable by the user.

Edit Controls

The parameters in this tab let you modify the qualities of existing brushstrokes. Clicking the Edit Controls tab puts the Viewer into Edit mode, which lets you pick a brushstroke with the strokeIndex slider, then modify its properties using the parameters below.



strokeIndex
A slider that lets you pick an individual brushstroke to modify. Each brushstroke is numbered in the order in which it was created. As you change the strokeIndex number, the corresponding brushstroke appears with a superimposed path to indicate that it is selected.

Tool
This parameter lets you change the effect that a brushstroke has on the image after the fact. For example, you can change a paint stroke into a smudge, eraser, reveal, or clone stroke at any time.

Convert Stroke
Determines what happens to a stroke after the frame in which it's drawn. Strokes can be set to last for only a single frame, or to persist for the duration of the *QuickPaint* node, or to change their shape, interpolating from one frame to the next.

The types of strokes available are:

- *Persist:* Paint strokes are static, and remain onscreen from the frame in which they were drawn until the last frame of the *QuickPaint* node.
- *Frame:* Paint strokes appear only in the frame in which they were drawn.
- *Interp:* Paint strokes remain onscreen from the frame in which they were drawn until the last frame of the *QuickPaint* node. Their shape can be keyframed over time, to create interpolated animation effects.

Color
A color control that lets you change the color of an existing stroke.

alpha
A slider that lets you change the alpha channel of the currently selected stroke. This control does not modify the color of strokes, because strokes are not premultiplied. This parameter affects the R, G, B, and A color channels of the image equally.

brushSize
The size of the brush. You can also Control-drag in the Viewer to set the brush size.

aspectRatio
Aspect ratio of the circular strokes.

startPoint
The point at which the stroke begins, based on a percentage offset. This parameter can be keyframed to create write-on/off effects.

endPoint
The point at which the stroke stops, based on a percentage offset. This parameter can also be keyframed to create write-on/off effects.

opacity
A slider that lets you change the transparency of the currently selected brushstroke. Unlike the alpha slider, this parameter does change the color of the stroke.

keyFrames
This parameter contains keyframes applied to the *QuickPaint* node, but is not modifiable by the user.

panX, panY
These parameters let you move strokes, using an offset from each stroke's original position.

Paint Globals

The parameters in this tab control how stroke information is captured when using a digital graphics tablet or mouse.

snapshotInterval
Sets how many strokes are applied before the image is cached. For low-resolution images, you can probably set the value lower, but if you set it too low when working with film plates, you'll spend all your time caching 2K plates, which is bad.

maxPressure
Sets the maximum amount of pressure you can apply.

pressureCurve
You can control the pressure response of the stylus by loading this parameter into the Curve Editor. You can also, of course, change the graphics tablet's settings outside of the software.

compressSave
When this control is on, the node is saved in binary format, which is faster and smaller. When this control is turned off, Shake saves the node in an editable ASCII format (described below).

moveExpression
This expression controls the drop-off curve for the Magnet drag mode when you move a group of points.

Exercise Caution With compressSave

If you have a considerable amount of work, you should ensure that compressSave is enabled. If it is not turned on, you run the risk of creating a file too large for your computer to read in.

StrokeData Synopsis

When compressSave is enabled, data is written in a compressed format and is therefore illegible to you. However, it is faster and more compact when compressed. Each stroke has the following data in quotation marks when saved in ASCII format:

```
"FORMAT TOOL MASK
    NUMDATA;TIME,TYPE;X;Y;P;X;Y;P;...X;Y;P;TIME,TYPE;X;Y;P;....X;Y;P;
    ",
```

followed by inPoint, outPoint, and so on, up to yOffset.

StrokeData	Type	Function
TOOL	int	• Paint = 1 • Smudge = 2 • Eraser = 3 • Reveal = 4 • Clone = 5
MASK	float	The active channels on which the paint is applied.
FORMAT		Reserved for future use.
NUMDATA	int	The number of data pieces per point, placed for future compatibility reasons. This is currently 3—the X,Y position and the pressure of each point.
TYPE	float	The time the data corresponds to.
TYPE	int	The mode the data corresponds to. • Persist = 0 • Frame = 3 • Interp = 4 • inPoint (Reserved for future use) = 1 • outPoint (Reserved for future use) = 2
X;Y;P	float	X, Y position, pressure.

Shake-Generated Images

22

This chapter covers the use of the Shake-generated image nodes found within the Image Tool tab.

Generating Images With Shake

This chapter covers various nodes that generate images directly within Shake. These nodes can be used for a variety of purposes—as backgrounds, as masks, or as inputs to alter the affect of filter or warp nodes.

Checker

The *Checker* node generates a checkerboard within the boundaries of the image. It is handy to test warps, or to split a screen in half. To make a specific number of tiles per row or column (for example, 4 x 4), divide the width and the height by the number—height/4 yields 4 rows.

Parameters

This node displays the following controls in the Parameters tab:

width, height

The width and height value fields in the Res parameter set the size of the generated image.

bytes
The bit depth of the generated image. There are three settings: 8 bits, 16 bits, or float (1, 2, or 4 bytes per channel).

xSize, ySize
The width and height of each checker in the pattern.

Color

The *Color* node generates a solid field of color within the width and height of the image.

Parameters

This node displays the following controls in the Parameters tab:

width, height
The width and height value fields in the Res parameter set the size of the generated image.

bytes
The bit depth of the generated image. There are three settings: 8 bits, 16 bits, or float (1, 2, or 4 bytes per channel).

Color
A color control that lets you set the color of the generated image.

alpha
A slider that lets you adjust the transparency of the generated image by modifying the alpha channel.

depth
A slider that lets you adjust the Z depth of the image.

ColorWheel

The *ColorWheel* node generates a primitive color wheel. It can also be used as a tool to determine what HSV/HLS commands, such as *AdjustHSV* and *ChromaKey,* are doing.

Parameters

This node displays the following controls in the Parameters tab:

width, height
The width and height value fields in the Res parameter set the size of the generated image.

bytes
The bit depth of the generated image. There are three settings: 8 bits, 16 bits, or float (1, 2, or 4 bytes per channel).

satCenter
Saturation of center area.

satEdge
Saturation of edge area.

valCenter
Value of center area.

valEdge
Value of edge area.

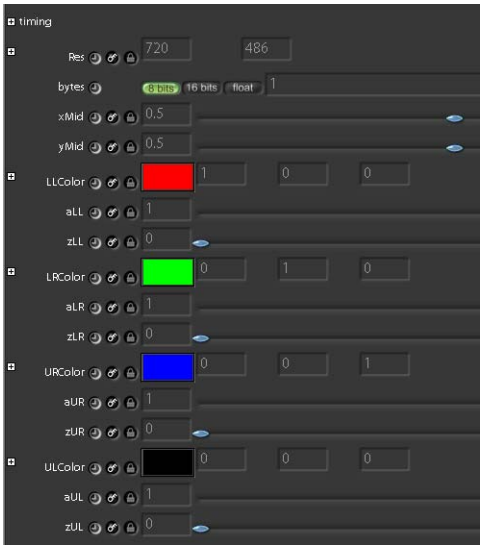
Note: By adjusting the satCenter, satEdge, valCenter, and valEdge parameters, you can change the distribution of color and brightness across the color wheel being generated.

Grad

The *Grad* node generates a gradient between four corners of different colors. The count order of the corners is: Corner 1 in the lower-left corner, corner 2 in the lower-right corner, and so on. For a simple gradient ramp, use the *Ramp* node. For a radial gradient, use the *RGrad* node.

Parameters

This node displays the following controls in the Parameters tab:



width, height

The width and height value fields in the Res parameter set the size of the generated image.

bytes

The bit depth of the generated image. There are three settings: 8 bits, 16 bits, or float (1, 2, or 4 bytes per channel).

xMid, yMid

The midpoint of the gradient, where all four colors meet.

LLColor, aLL, zLL

The color, alpha channel value, and Z channel value at the lower-left corner. The color defaults to 100 percent red.

LRColor, aLR, zLR

The color, alpha channel value, and Z channel value at the lower-right corner. The color defaults to 100 percent green.

URColor, aUR, zUR
The color, alpha channel value, and Z channel value at the upper-right corner. The color defaults to 100-percent blue.

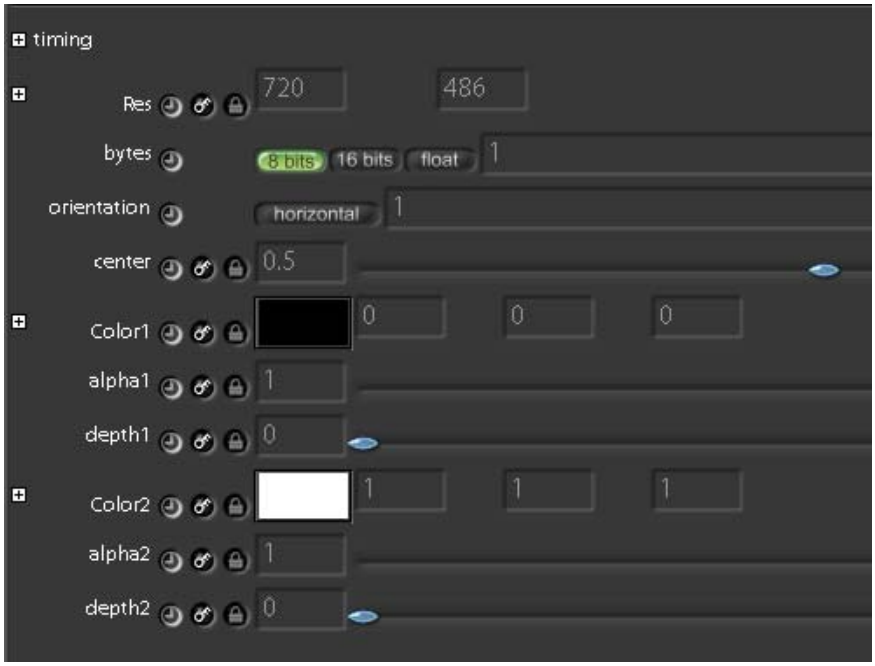
ULColor, aUL, zUL
The color, alpha channel value, and Z channel value at the upper-left corner. The color defaults to black.

Ramp

The *Ramp* node generates a linear gradient between two edges. You can set the direction of the ramp to horizontal or vertical. The *Ramp* is, among other things, a useful tool for analyzing color-correction nodes. Attach a horizontal *Ramp* node to any of the color-correction nodes, then attach the color-correction node to a *PlotScanline* node. For a radial gradient, use the *RGrad* node. For a four-corner ramp, use the *Grad* node.

Parameters

This node displays the following controls in the Parameters tab:



width, height
The width and height value fields in the Res parameter set the size of the generated image.

bytes
The bit depth of the generated image. There are three settings: 8 bits, 16 bits, or float (1, 2, or 4 bytes per channel).

orientation
Toggles between generating a horizontal or vertical gradient.

center
The point in the frame at which there's a 50-percent blend of each color.

Color1, alpha1, depth1
The color, alpha value, and Z channel value at the left (horizontal) or bottom (vertical) of the frame.

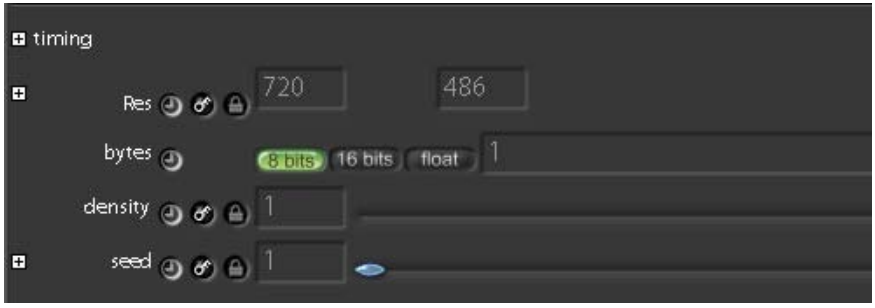
Color2, alpha2, depth2
The color, alpha value, and Z channel value at the right (horizontal) or top (vertical) of the frame.

Rand

The *Rand* node generates a random field of noise. The field does not resample if you change the resolution or density—you can animate the density without pixels randomly changing. The seed is set to time by default so that it changes every frame, but you can of course lock this parameter to one value.

Parameters

This node displays the following controls in the Parameters tab:



width, height
The width and height value fields in the Res parameter set the size of the generated image.

bytes
The bit depth of the generated image. There are three settings: 8 bits, 16 bits, or float (1, 2, or 4 bytes per channel).

density
The density of the pixels, from 0 to 1. A lower density results in fewer random pixels.

seed
Changes the random pattern of noise that's created. When Shake generates a random pattern of values, you need to make sure for purposes of compositing that you can recreate the same random pattern a second time. In other words, you want to be able to create different random patterns, evaluating each one until you find the one that works best, but then you don't want that particular random pattern to change again.

Shake uses the seed value as the basis for generating a random value. Using the same seed value results in the same random value being generated, so that your image doesn't change every time you re-render. Use a single value for a static result, or use the keyword "time" to create a pattern of random values that changes over time.

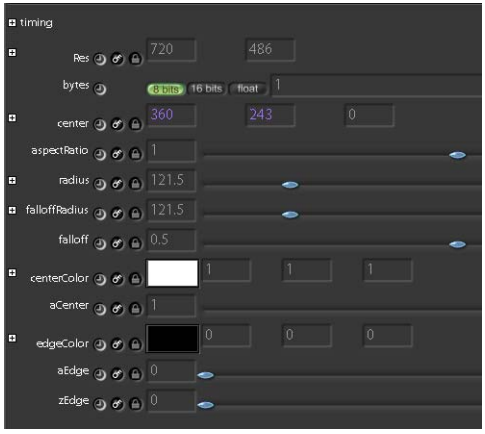
For more information on using random numbers in expressions, see "Reference Tables for Functions, Variables, and Expressions" on page 941.

RGrad

The *RGrad* node generates a radial gradient. You can also control the falloff to make circles. For a simple color ramp, use the *Ramp* node. For a four-corner gradient, use the *Grad* node.

Parameters

This node displays the following controls in the Parameters tab:



width, height
The width and height value fields in the Res parameter set the size of the generated image.

bytes
The bit depth of the generated image. There are three settings: 8 bits, 16 bits, or float (1, 2, or 4 bytes per channel).

xCenter, yCenter
The pixel defining the center of the gradient.

aspectRatio
This parameter inherits the current value of the defaultAspect global parameter. If you're working on a nonsquare pixel or anamorphic image, it is important that this value is set correctly to avoid unwanted distortion in the image.

radius
The non-blurred radius of the center.

falloffRadius
The blurred edge radius (the total width of the circle is radius+falloffRadius).

falloff
The midpoint, in terms of percentage, of the falloff. 0 or 1 equals a hard-edge circle; .5 is a smooth ramp.

centerColor, aCenter, zCenter
The color, transparency, and Z depth at the center of the gradient.

edgeColor, aEdge, zEdge
The color, transparency, and Z depth at the edge of the gradient.

Text

The *Text* node calls on software that is basically identical to GL Render for character generation. You can type in your text, or use variables and links to insert characters, making it an ideal tool for generating slates.

You can use any TrueType (.ttf) and Type 1 (PostScript, .pfa for ASCII and .pfb for binary) font. If an Adobe Font Metrics (.afm) file is present for the font (for example, you have MyFont.pfa and MyFont.afm), it is supported and provides kerning for the font. Shake first looks for fonts in its distribution directory, under fonts. You can also place them in a direct path by setting the environment variable *NR_FONT_PATH*. Finally, Shake also detects fonts placed in the standard directories for your OS:

Macintosh OS X: All "Installed" directories.

Linux: /usr/lib/DPS/AFM

The *Text* node uses the Shake implementation of the GL Render. It allows you to not only manipulate the characters in 3D space (including X, Y, and Z position, rotation, and scaling), but also in a camera field of view. Because of this, it is better to animate text within the *Text* (or *AddText*) node to ensure crisp, clean edges.

To select a font in the interface:

1 In the *Text* node parameters, open the font pop-up menu.

To preview a font in the Viewer, right-click the font name in the list.

2 To choose a font from the list, click the font name.

Note: For long font lists, drag the scroll bar to see more fonts.

You can also use the following shortcuts at any time without any special formatting:

Text Shortcut	Writes
{parameter}	Prints either the local or global parameter value, for example: *Script={scriptName}* writes *Script=My_Script.shk*
{nodeName.parameter}	Prints the parameter's value from a selected node, for example: *Red Val = {Mult1.red}* could write *Red Val = .6*
\n	New line. Example: *Hello\nWorld* returns Hello World
%f	Unpadded frame number
%F	Four-digit padded frame number
%t	Short non-dropframe timecode: no 00: (if not needed)
%T	Long non-dropframe timecode: hr:mn:sc:fr
%tD	Short dropframe timecode: no 00: (if not needed)
%TD	Long dropframe timecode: hr:mn:sc:fr
%H	Host name
%U	Username
%c,%C	Locale's center
%d	Locale's day 01-31
%D	Locale's abbreviated day name: Wed
%E	Locale's full day name: Wednesday
%m	Locale's month: 01-12
%M	Locale's abbreviated month name: Nov

Text Shortcut	Writes
%N	Locale's full month name: November
%x,%X	Full date representation: mm/dd/yy
%y	Year without century: 00-99
%Y	Year as ccyy

Examples 1:

Text String	Writes
My name is Peter	`My name is Peter`
My name is %U	If login = Dufus: `My name is Dufus`
My name is %U.\nToday is %M. %d	`My name is Dufus.` `Today is Nov. 12`
Mult red = {Mult1.red}	Assuming the node *Mult1* exists, and the red value is .46: `Mult red = .46`

To get special characters, such as umlauts, copyright symbols, and so on, use octal and hexadecimal codes preceded by a \ (backslash). These codes can be found in UNIX with the main page for "printf" in its special characters section. The following example was provided by Thomas Kumlehn, at Double Negative. Copy and paste it into the *Text* node:

```
Auml=\xC4, Ouml=\xD6, Uuml=\xDC, \n auml=\xE4, ouml=\xF6, uuml=\xFC
     \n Szett=\xDF \ntm=\x99, Dot=\x95, (R)=\xAE, (C)=\xA9
```

A good reference website for characters can be found at www.asciitable.com. (Thanks to Christer Dahl for this tip.)

To use expressions, preface the text with "a :" (but without the enclosing quotation marks).

All printed commands must be enclosed in quotation marks. For example, if you want to print "Hell" from frames 1 to 10 and "o World" from frames 11 onward, enter:

```
:time<11?"Hell":"o World"
```

Finally, you can also use full C formatting for your strings. This is initialized with "a :"at the start of the text string as well:

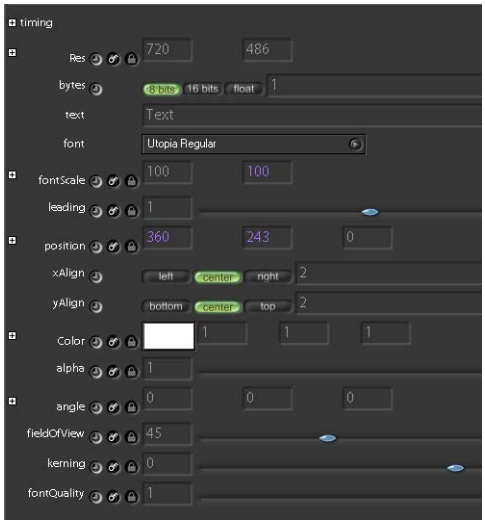
```
: stringf(
  "Red = %4.2f  at Frame %03f",
  Grad1.red1, time
)
```

To append strings from another parameter, use something like:

```
in Text1.text:    Hello
in Text2.text:    : Text1.text + " World"
```

Parameters

This node displays the following controls in the Parameters tab:



width, height

The width and height value fields in the Res parameter set the size of the frame containing the generated image.

bytes

The bit depth of the generated image. There are three settings: 8 bits, 16 bits, or float (1, 2, or 4 bytes per channel).

text

A text field where you enter the text you want to generate in the Viewer.

font

A pop-up menu that lets you choose a font.

xFontScale, yFontScale

Two sliders that let you change the horizontal and vertical size of the generated text. By default, yFontScale is linked to xFontScale.

leading

The amount of spacing between each line if there are multiple lines of text.

xPos, yPos, zPos

The horizontal, vertical, and Z depth position of the text in the frame. The text is positioned relative to the center point that's defined by the xAlign and yAlign parameter settings.

xAlign
Three buttons that let you define how the generated text should be aligned, horizontally. The options are:

- *left:* Aligns the text from the left edge.
- *center:* Aligns the text from the center.
- *right:* Aligns the text from the right edge.

yAlign
Three buttons that let you define how the generated text should be aligned, vertically. The options are:

- *bottom:* Aligns the text from the bottom edge.
- *center:* Aligns the text from the middle.
- *top:* Aligns the text from the top edge.

Color
A color control lets you set the color of the text.

alpha
A slider lets you adjust the transparency of the generated text.

xAngle
A slider lets you create a 3D effect by spinning the text vertically, relative to the position of the yAlign parameter.

yAngle
A slider lets you create a 3D effect by spinning the text horizontally, relative to the position of the xAlign parameter.

fieldOfView
The aperture angle in degrees of the virtual camera used to render the 3D positioning of the *xAngle* and *yAngle* parameters.

kerning
The spacing between each letter. Larger values space the letters farther apart, while smaller values bring the characters closer together. You can also use negative values to make the characters overlap.

fontQuality
The polygonalization factor of the font splines. This is conservatively set to a high value. For flat artwork, you can probably get away with a value of 0. When you have extreme perspective, you should keep it set to a high value.

Tile

The *Tile* node is located in the Other tab. *Tile* does not generate an image, but makes small tiles of an image within that image. The more tiles created, the slower the processing (for example, more than 40 tiles).

Parameters

This node displays the following controls in the Parameters tab:

nXTile

The number of times the image is duplicated and shrunk horizontally.

nYTile

The number of times the image is duplicated and shrunk vertically.

Color Correction

23

Shake's color-correction and pixel-analyzer functions provide many ways of analyzing and manipulating the color values of your images.

Bit Depth, Color Space, and Color Correction

By default, Shake works with a color range of 0 to 1 in RGB linear space. Shake allows you to work at different bit depths, so 0 is considered black and 1 is considered white.

Ordinarily, values above 1 and below 0 are *clamped* (constrained to a value between 1 and 0) as they're passed from node to node down the tree of image-processing functions. This can have a profound effect on the resulting image processing in your script, with the following caveats:

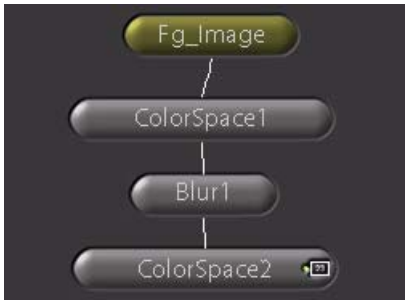
- Nodes that concatenate are not subject to this clamping.
- Clamping does not occur when you work in float bit depth, because 32-bit (float) computations preserve values above 1 and below 0.

Working in Different Color Spaces

When working with logarithmic Cineon plates, apply a *LogLin* node to avoid unpredictable results. The *LogLin* node allows you to jump from logarithmic to linear space, or linear to logarithmic space. For more information, see "The Logarithmic Cineon File" on page 437.

To apply an effect such as a *Blur* node in a different color space (for example, to blur the color difference channels in a YUV image, but not the luminance), apply a *ColorSpace* node to the image to convert it to the different color space. Then, add the effect node—in this example, the *Blur* node. To return to your original color space, add another *ColorSpace* node.

For a practical discussion on using this technique, see Chapter 24, "Keying," on page 681.



Note: To view the images in this chapter in color, refer to the onscreen PDF version of this documentation.

Concatenation of Color-Correction Nodes

A powerful aspect of Shake's color handling is its ability to concatenate many color corrections. This means that if you have ten concatenating color functions in a row, Shake internally compiles the functions into a single lookup table, then executes that table. (Internally to Shake, one node is processed, rather than ten.)

When you concatenate color-correction nodes, you avoid clamping values over 1 and under 0 *within the concatenating node group*. By not clamping, you preserve much more image data. As a result, concatenation preserves quality and speeds processing time.

Note: Color-correction nodes that process image data in float don't concatenate because there is no advantage to doing so. Because the image data is calculated in 32-bit float space, clipping isn't an issue. Also, there is no computational advantage to concatenating nodes that are subject to float calculations.

Which Nodes Concatenate With One Another?

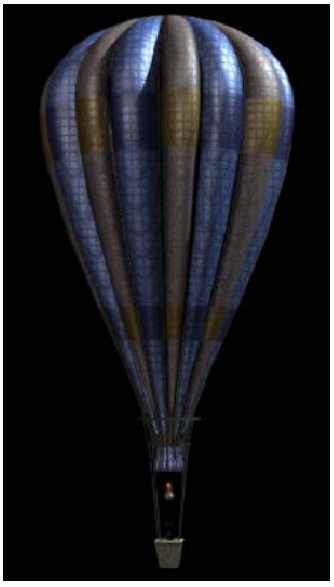
Nodes that concatenate with one another are labeled with the letter "c" (for concatenation) in the upper-left corner of their buttons in the Tool tabs. (The "c" doesn't appear on nodes in the Node View.)

Note: Nodes only concatenate with other nodes that have the same color "c."

Example 1: Proper Color-Correction Concatenation

The following example illustrates the correct method of concatenating color-correction nodes. First, a *Brightness* node (set to a value of 4) is applied to an image. Next, a second *Brightness* node (set to a value of .25) is added to the tree. The two nodes concatenate (.25 * 4 = 1) to multiply the image by 1, resulting in no change to the image. This is the ideal result.

If you turn on enhanced Node View (with the pointer in the Node View, press Control-E), you'll see that the nodes concatenate—indicated by a green line linking the affected nodes.



Example 2: Incorrect Color-Correction Concatenation

The next example illustrates the pitfalls of incorrectly combining color correction with other nodes. It uses the same node setup as in the previous example, but with an added *Blur* node, which breaks the concatenation. The result is that two separate color- correction adjustments are made. Because 8-bit and 16-bit image processing paths truncate values above 1 and below 0, the above values have unexpectedly negative results.

Prior to the *Blur* node, all of the values are boosted to 1 when multiplied by the first *Brightness* node's adjustment of 4. After the *Blur* node, the values are then dropped to a maximum value of .25 when the second *Brightness* value of .25 is applied.

As you can see, the result is altogether different than the result in Example 1. This can be avoided by always making sure that the color-correction nodes in your tree are properly concatenating.

The following nodes concatenate with one another:

- *Add*
- *Brightness*
- *Clamp*
- *Compress*
- *ContrastRGB* (but not *ContrastLum*)
- *DelogC*
- *Expand*
- *Fade*
- *Gamma*
- *Invert*
- *LogC*
- *Lookup*
- *Mult*
- *Set*
- *Solarize*

Note: *AdjustHSV* and *LookupHSV* only concatenate with each other.

Masked Nodes Break Concatenation

If you mask a node, concatenation is broken. To avoid broken concatenation, use a node tree structure with *KeyMix*.

Making Concatenation Visible

When you turn on enhanced Node View, you can see the links between concatenated nodes. Make sure that showConcatenationLinks is set to enhanced in the enhancedNodeView subtree of the Globals tab, then turn on the enhanced Node View. Nodes that are currently concatenating appear in the Node View linked with a green line. For more information, see "Using the Enhanced Node View" on page 221.

Avoiding Value Clamping Using a Bit Depth of Float

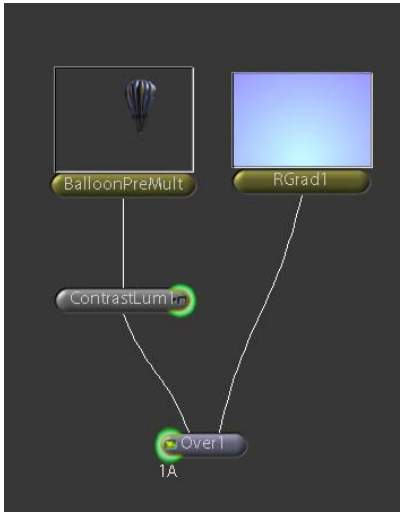
One way to avoid the consequences of broken concatenation is to boost your bit depth to float with a *Bytes* node, prior to performing any color correction. This sets up the image processing path within the node tree to preserve values above 1 and below 0, instead of clamping them.

Note: Be aware that the float bit depth is more processor-intensive, and can result in longer render times. For more information, see "Bit Depth" on page 408.

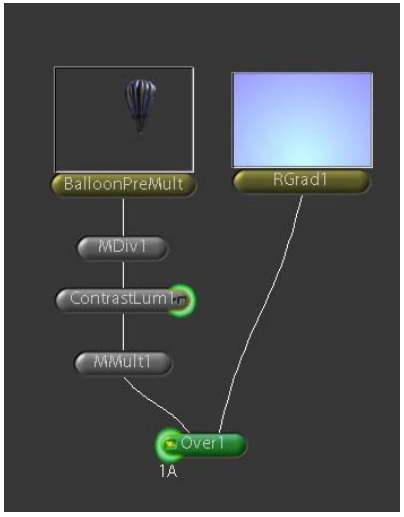
Premultiplied Elements and CG Element Correction

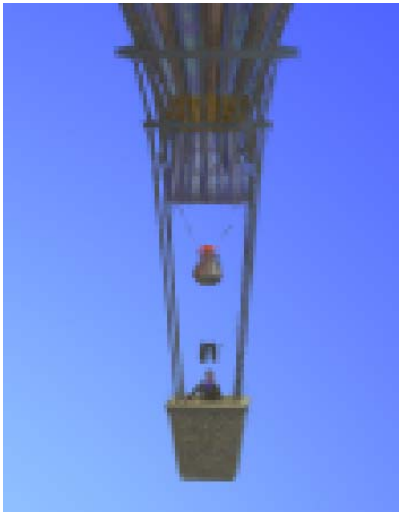
You may sometimes spot problems in the edges of computer-generated images when applying color-correction nodes. A cardinal rule of image processing in Shake is to *always color correct unpremultiplied images*.

In the following example, a computer-generated graphic is composited with a background image. The addition of a *ContrastLum* node (with a value of .6) results in premultiplication issues—manifested as a fringing around the edges of the image.

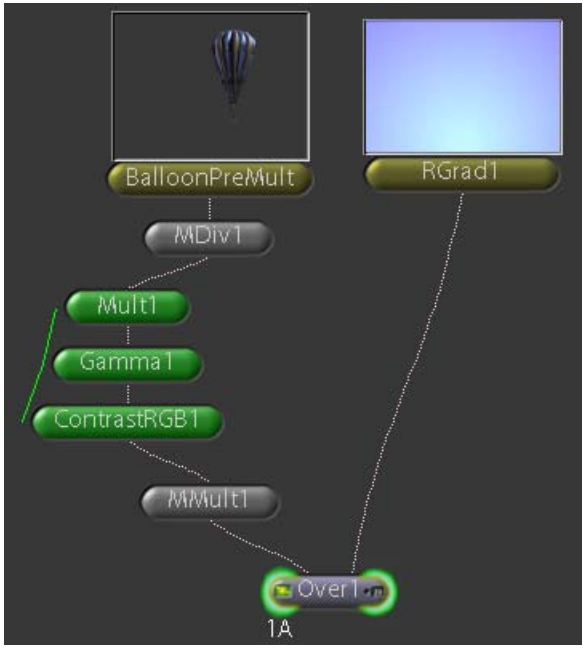


To eliminate this problem, unpremultiply the graphic prior to color correction by inserting an *MDiv* node prior to any color-correction nodes in the tree. Later, at the end of the chain of color-correction nodes you apply in the tree, make sure you once again premultiply the graphic by adding an *MMult* node (this can also be done by turning on the preMultiply parameter in the *Over* node).





The following screenshot shows a correctly set up node tree. *Mult, Gamma,* and *ContrastRGB* nodes are inserted between a pair of *MDiv* and *MMult* nodes, prior to compositing the two images with a layering node (the *Over* node).



Note: In the above example, all three color-correction nodes concatenate properly, shown by the green line that is visible with enhanced Node View is on.

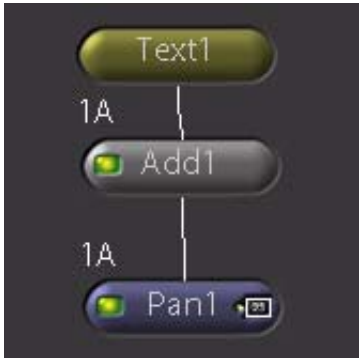
For a more detailed description of premultiplication and its importance in compositing, see "About Premultiplication and Compositing" on page 421.

Color Correction and the Infinite Workspace

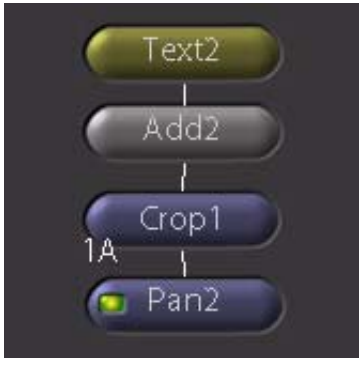
The Shake engine applies effects to a potentially infinite canvas, so occasionally you may encounter an unexpected result when you color correct and pan a small element across a larger element. This occurs when an *Invert*, *Set*, or *Add* node is applied to an image, since the previously black pixels outside of the frame are changed to a different color.

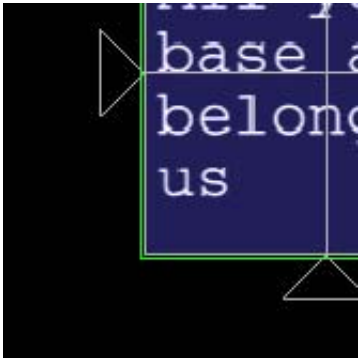
In the following example, an artifact of Internet pop culture is recreated using a *Text* node. The default black background color is raised to blue with an *Add* node.

When the image is panned, blue continues to appear in the area that was previously outside of the frame.

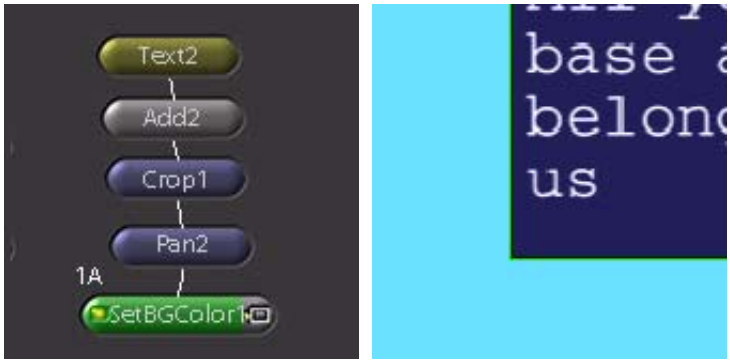


To create a black outline, insert a *Crop* node before the *Pan* in the node tree:





To color correct the area outside of the Domain of Definition (DOD, represented by the green bounding box), use the *SetBGColor* node.



For more information on the Infinite Workspace, see "Taking Advantage of the Infinite Workspace" on page 405. For more information on the DOD, see "The Domain of Definition (DOD)" on page 82.

Using the Color Picker

The Color Picker tab is a centralized interface that lets you assign colors to node parameters using the ColorWheel, luminance gradient, swatches from a color palette, or numerically, using a variety of color models. You can also store your own frequently used color swatches for future use in the Palette.

Color sampling swatches

Choose which color to pick when clicking in the Viewer.

ColorWheel

Click the ColorWheel and luminance bar to choose a color.

Palette

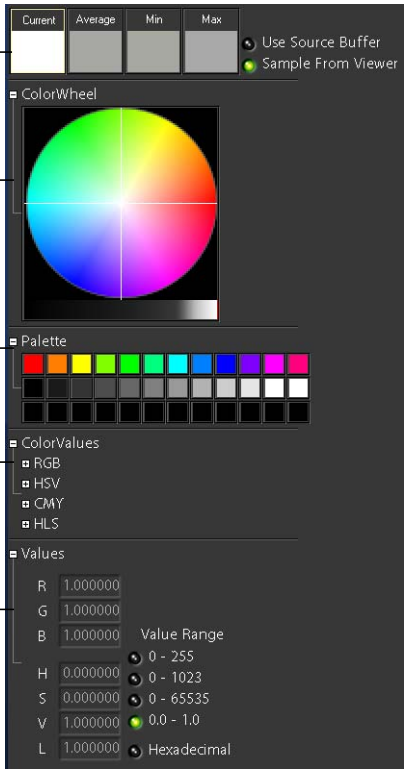
Store frequently used colors here.

ColorValues

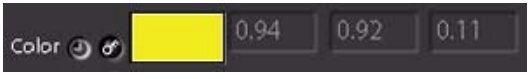
Adjust individual color channels using a variety of color models.

Values

Adjust individual color channels using different numeric representations.



Many of the nodes described in this chapter use the Color control that appears within the Parameters tab.



This control corresponds to the Color Picker (clicking the Color control swatch opens the Color Picker window). Both the Color control in the Parameters tab and the Color Picker can be used interchangeably.

Using Controls in the Color Picker

You can adjust the controls in the Color Picker in the following ways:

To choose a color from the ColorWheel:

- Drag the pointer in the wheel to select a point.

  Crosshairs make it easy to spot the precise color that's chosen, and the four sampling controls above the ColorWheel reflect the selection.

To change the overall value of the wheel and the selected color:

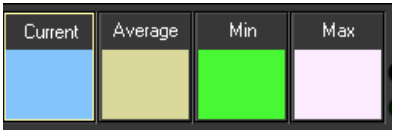
- Drag in the luminance bar underneath the ColorWheel.

  The overall brightness of the ColorWheel changes.

To sample color from the Viewer:

- Drag over the image in the Viewer.

  The ColorWheel and color sampling swatches above the ColorWheel automatically update as you drag in the Viewer.



The color swatches sample pixels from the image in four different ways:

- *Current:* Samples the exact color value of the last pixel you clicked or dragged over.
- *Average:* Samples the average color value of all the pixels you drag over.
- *Min:* Samples the minimum color value of all the pixels you drag over.
- *Max:* Samples the maximum color value of all the pixels you drag over.

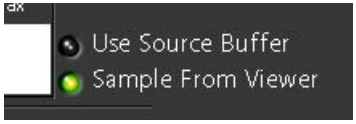
To load a sampled color into a Color control in the Parameters tab:

1. In the Parameters tab, click the Color control for the color parameter you want to adjust.

   A yellow outline appears around the edge of the Color control, and the Color Picker opens.

2. Select a new color in any of the ways described above (using the ColorWheel or dragging over the image in the Viewer, for example).

   Note: If you're sampling a color from an image that has no color-correction nodes applied, turn on Sample From Viewer in the Color Picker.



   If you're sampling a color from an image that has already been color-modified (for example, an image modifed by a *Mult* node), then the real-time update of the color correction interferes with onscreen selection of color, causing an unwanted feedback loop. To avoid this, turn on Use Source Buffer instead, and color values are taken from the original image node, not the currently selected node.

   This is especially useful when doing scrubs for keying nodes, since it allows you to pick color values from the *prekeyed* image.

   Notice that the color sampling swatches above the ColorWheel update as you drag.

3. Click one of the color sampling swatches to load its color into the selected Color control (in the Parameters tab).

   The color sampling swatches reset themselves each time you click in the Viewer. If you've accidentally dragged a region and the Min and Max values are unsatisfactorily set, simply click in the Viewer again to choose new values in all swatches.

   You can also load colors into the Parameters tab using the Palette, located under the ColorWheel. The Palette can also be used to store colors that you use repeatedly in your composition.

To select a color from the Palette:

- Click a color swatch.

  You can also drag and drop between the Palette swatches and other Color Picker swatches.

To assign a color to a Palette swatch:

- Drag a color sampling swatch (above the ColorWheel) and drop it into the Palette swatches area.

The new color appears in the Palette.

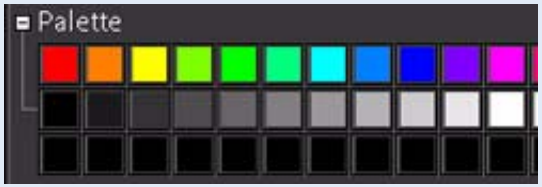
To save your own custom color assignments:

- Choose File > Save Interface Settings.

You can also choose or adjust colors numerically in the Color Picker by manipulating the values of each individual channel.

Defining Custom Default Palette Colors

If you want to change the default Palette colors that Shake starts with, add the following declaration to a .h file in the *ui* directory:




```
nuiSetColor(1,1,0,0);
nuiSetColor(2,1,0.5,0);
nuiSetColor(3,1,1,0);
etc...
```

The syntax is:

```
nuiSetColor(swatchNumber, redValue, greenValue, blueValue);
```

The first value is the swatch position in the Palette (swatchNumber) from left to right starting with the top line, and the next three values are the red, green, and blue values designating the color you want in RGB space.

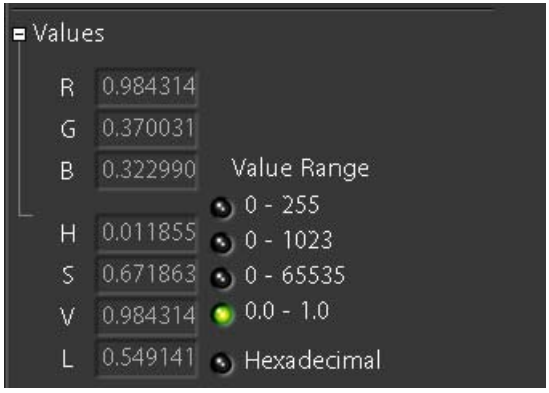
To read different color channels for a node, do one of the following:

- Open the ColorValues subtree, then open one of the color space subtrees.

  Use the sliders to adjust colors based on individual channels from the RGB, HSV, CMYK, and HLS color space models.

- Open the Values subtree.

Use the color channel value fields to enter numeric values or expressions. The numeric ranges representing each color channel can be changed using the ValueRange button, making manipulation of color by expressions easier.



The channel slider buttons can also be individually controlled by the same hot keys used for the Virtual Color Picker. These channel sliders let you adjust colors using individual channels from each of three different color space representations: RGB, HSV, and CMYK.

Note: Offset (hot key O) is not included in the channel slider keys.

To animate color values:

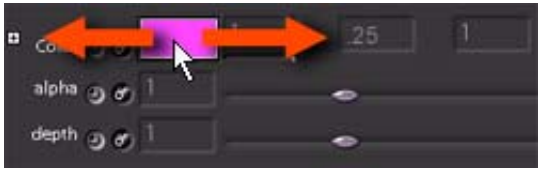
- Use the associated Autokey button in the Parameters tab.

Using a Color Control Within the Parameters Tab

The Color controls found in the Parameters tab or Tweaker window have many powerful shortcuts you can use to directly manipulate individual color channels. In many instances, using the Color Picker may be unnecessary.

To use the Virtual Color Picker to make specific adjustments:

- Press a channel key on the keyboard, then drag within the Color control to adjust that channel.



The following chart lists all the keyboard shortcuts for color adjustments within a color control.

Keyboard	Channel	Description
R	Red	Adjusts red channel independently.
G	Green	Adjusts green channel independently.
B	Blue	Adjusts blue channel independently.
O	Offset	Boosts or lowers all color channels relative to one another.
H	Hue	Adjusts all channels by rotating around the ColorWheel.
S	Saturation	Adjusts color saturation, according to the HSV model.
V	Value	Adjusts color "brightness," according to the HSV model.
T	Temperature	Adjusts overall color between reds and blues.
C	Cyan	Adjusts cyan, according to the CYMK colorspace model.
M	Magenta	Adjusts magenta, according to CMYK.
Y	Yellow	Adjusts yellow according to CMYK.
L	Luminance	Adjusts black level, otherwise referred to as luminance.

To quickly select a color from a color control, use the Virtual Color Picker.

To access the Virtual Color Picker from any Color control:

- Right-click a color control, then drag to select a color from the Virtual Color Picker.

The Virtual Color Picker is available in the Parameters tab of all nodes that contain Color controls, or in the Color Picker.

Customizing the Palette and Color Picker Interface

These commands are placed in your *ui.h* file. For more information on customizing Shake, see Chapter 14, "Customizing Shake," on page 355.

Code	Description

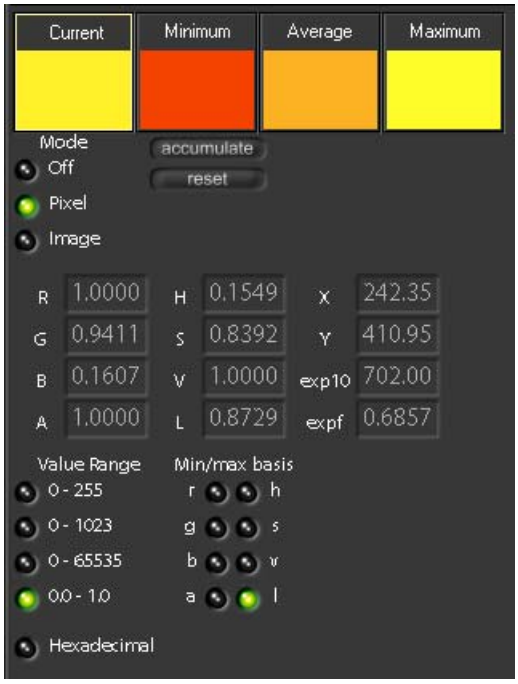
`nuiSetColor(1,1,0,0);`
`nuiSetColor(2,1,0.5,0);`
`nuiSetColor(3,1,1,0);`
	Assigns a color to a Palette swatch; the first number is the assigned box. Values are in a range of 0 to 1.

`nuiPushControlGroup("Color");`
`nuiGroupControl("MyFunction.red");`
`nuiGroupControl("MyFunction.green");`
`nuiGroupControl("MyFunction.blue");`
`nuiPopControlGroup();`
`nuiPushControlWidget(`
`"Color",`
`nuiConnectColorTriplet(`
`kRGBToggle,`
`kCurrentColor,`
`1`
`)`
`);`
`);`
	Assigns a Color Picker to your custom macros. This code creates a subtree named "Color" that contains the three parameters–red, green, and blue—although these can be any three parameters. The last function (*nuiConnectColorPControl*) selects what color space the values are returned in, what type of value, and if you want to use the source buffer or not.

Using the Pixel Analyzer

The Pixel Analyzer tab is an analysis tool to find and compare different color values in an image. You can examine minimum, average, current, or maximum pixel values on a selection (that you make), or across an entire image.

Note: The Pixel Analyzer tab should not be confused with the *PixelAnalyzer* node, found in the Other tab. For more information, see "The PixelAnalyzer Node" on page 631.



Using the Pixel Analyzer is very similar to sampling color values with the Color Picker. When you drag across an image in the Viewer with the pointer, the values update in the Pixel Analyzer. You can usually use the default Pixel Analyzer settings.

Click the color swatch that you want to examine—Current, Minimum, Average, or Maximum color value. You can toggle between the different color swatches repeatedly without having to drag again in the Viewer. The values appear in the value fields below the color swatches.

Since the Pixel Analyzer keeps the examined pixels in memory, if you switch images in the same Viewer, the Pixel Analyzer updates its values based on the new image. Because of this, you can compare images or perform color corrections, as the color correction constantly updates the Analyzer.

Using the Pixel Analyzer Tab to Set Levels

The following example shows you how the Pixel Analyzer can be used to perform a similar operation to that of the Auto Levels command in Adobe Photoshop. This method works by using the Pixel Analyzer to automatically find the lowest and highest values in each channel of an image. You can then assign these values to an *Expand* node in order to push the lowest values to 0 (black), and the highest values to 1 (white). In this example, you can see the effect clearly in the landscape under the clouds.

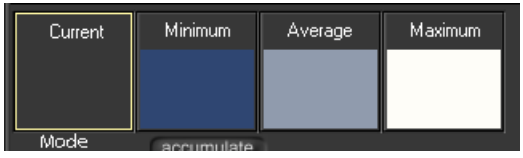
To set levels using the Pixel Analyzer:

1. Read in an image using a *FileIn* node, then attach a Color–*Expand* node to it.

Cloud images © 2004 M. Gisborne

2. Click the left side of the *FileIn* node to load the image into the Viewer, then click the right side of the *Expand* node to load its parameters into the Parameters tab.
3. Open the Pixel Analyzer tab.
4. Switch to Image mode.

The Current, Minimum, Average, and Maximum values of the image in the Viewer appear in the color swatches at the top of the window.



5 Drag the Minimum color to the Low Color control of the *Expand* node in the Parameters tab.

6 Drag the Maximum color to the High Color control of the *Expand* node in the Parameters tab.

The image is now adjusted. Load the *Expand* node into the Viewer to see the result.

Before

After

Pixel Analyzer Controls

The Pixel Analyzer has the following controls:

Mode

- *Off*: Turns off the Pixel Analyzer.
- *Pixel*: Analyzes the pixels based upon the area scrubbed with the pointer.
- *Image*: Analyzes the entire image—no scrubbing is necessary.

Accumulate

When you click the Accumulate button, all scrubbed pixels (not just the current pixel) are considered for Average, Min, and Max calculations—until the Reset button is clicked. So, Average calculates the average of every scrub since the last Reset, and Min and Max replace their values if a new minimum or maximum value is scrubbed. If the Analyzer is on Image mode, this button has no effect.

Reset

Resets the scrubbed buffer to black.

Value Range
Shake numerically describes color as a range of 0 to 1 (0, 0, 0 is black; 1, 1, 1 is white). However, you can set a different numeric range—for example, 0, 0, 0 as black, and 255, 255, 255 as white.

Hexadecimal
This button toggles the numeric display to hexadecimal values.

Min/Max Basis
The Min/Max Basis buttons set the channel for calculation of the Minimum and Maximum swatches. Normally, this parameter is set to L (luminance). To determine the minimum values in only the red channel, toggle the Min/Max Basis to R (red). For example, one pure red pixel and one pure green pixel are equivalent pixels based on luminance. However, based on red, the green pixel has a minimum value of 0, and therefore the Minimum swatch returns a different value.

Custom Entries
You can insert your own functions to return data using the following code. You provide a label and the function in a *ui.h* file. The two default plugs are called *exp10* and *expf*:

```
gui.pixelAnalyzer.customLabel1 = “exp10”;
gui.pixelAnalyzer.customFunc1 = “(int)(1024*log(l/0.18)/log(10))”;
gui.pixelAnalyzer.customLabel2 = “expf”;
gui.pixelAnalyzer.customFunc2 = “log(l/0.18)/log(10)”;
```

The PixelAnalyzer Node

The *PixelAnalyzer* node, located in the Other tab, allows you to examine one or more areas of an image over a range of frames. The data is then stored as the average, minimum, and maximum values of each area on a frame-by-frame basis. This data can then be used by other nodes to perform tasks such as matching colors, or reducing flickering in a plate. This is done by feeding image data from a *PixelAnalyzer* node into expressions within one of Shake’s color-correction nodes.

Using the PixelAnalyzer Node

The workflow used for analyzing color with the *PixelAnalyzer* node is similar to that of the Tracker. However, the analyzer does not do any motion tracking itself. It merely grabs color values at the position of the defined analysis areas.

Note: The analysis areas can be animated, and can also be moved via data from a Tracker node. Use expressions to assign track data to the areaX and areaY parameters of the analysis area you want to match to the movement of a tracker.

When using the *PixelAnalyzer* node, it’s important to make sure that it’s loaded into the Viewer prior to performing the analysis. Otherwise, the analysis cannot be performed.

To analyze an area:

1. Attach the *PixelAnalyzer* node to an image. Double-click the *PixelAnalyzer* node to load its image into the Viewer and its parameters into the Parameters tab.
2. Position the analysis area box in the Viewer to examine the necessary area of the image, and adjust its size as necessary.

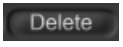
   Note: To animate the box, use the Viewer Autokey button, or use an expression to assign tracker data to the areaX and areaY parameters of the analysis area. Otherwise, the box remains stationary.
3. To create several analysis areas, use the Add button in the lower portion of the parameters.

4. Verify the frame range in the analysisRange parameter.
5. Press the Analyze Forward (or Analyze Backward) button in the Viewer.

The analysis begins. Each analysis area (controlled by the visibility toggle by the areaName) grabs color values within its area and calculates the average, minimum, and maximum values for that area.

To delete an analysis area, do one of the following:

- In the Viewer, select the box (click the box), and then click Delete in the *PixelAnalyzer* parameters.



- In the *PixelAnalyzer* parameters, click the areaName value field (highlighted green) and click Delete.

To save the data of an analysis area:

1. Click the Save button in the lower portion of the *PixelAnalyzer* parameters.

   The "Save pixel analysis data to file" window appears.
2. Select or create the directory to store the saved the data, name the file, and click OK.

Using the PixelAnalyzer to Correct Uneven Exposure

The following examples show how you can use the *PixelAnalyzer* node to obtain image data that is used to correct the exposure of a shot that dynamically brightens or darkens. The goal is to even the exposure of the image out so it doesn't change. These examples illustrate the power of expressions in Shake to automate complex operations.

Setting Up the PixelAnalyzer Node

Attach a *PixelAnalyzer* node to the problem image. It will eventually be used as a source of color values by expressions placed within the parameters of color-correction nodes. The color-correction nodes are not attached to the *PixelAnalyzer* node; instead, they're branched off of the source image. Three different examples show three different color-correction nodes in use—different situations may require different approaches, depending on the image.

Note: To accurately analyze changes in brightness, the *PixelAnalyzer* node's analysis area should be positioned over the brightest area of the image.

Method 1: Using an Add Node

Attach an *Add* node to the image, then enter the following expression into its red, green, and blue channels:

```
(PixelAnalyzer1.area1AverageRed@@1)-PixelAnalyzer1.area1AverageRed
(PixelAnalyzer1.area1AverageGreen@@1)-PixelAnalyzer1.area1AverageGreen
(PixelAnalyzer1.area1AverageBlue@@1)-PixelAnalyzer1.area1AverageBlue
```

The first part of the expression takes the first frame of the image as the base value (specified by @@1). The average channel values from all other frames are compared to frame 1. For every frame, the current channel value is subtracted from that of frame 1. For example, if at frame 1 the average red value is .5, and at frame 10 the average red value is .6, the above expression subtracts .1 from frame 10 to arrive at .5 again.

Note: To avoid problems when analyzing images with a lot of noise or grain, use the *PixelAnalyzer* node's Average value parameters.

In one possible scenario, examining the resulting image with the PlotScanLine viewer script might reveal that the midtones are OK, but that the darks are creeping down. This might indicate that the change in brightness is not occurring because of addition, but perhaps is a result of multiplication.

Method 2: Using a Brightness Node

If the *Add* node didn't provide satisfactory results, a *Brightness* node might have a better effect. Attach a *Brightness* node, then enter the following expression into the value parameter:

```
(PixelAnalyzer1.area1AverageRed@@1)/PixelAnalyzer1.area1AverageRed
```

The expression uses the same basic approach as in Method 1, except that the color value from frame 1 is divided by the current frame's color value. Since *Brightness* is a multiplier, this makes an adjustment based on the difference. As a result, if at frame 1 the average red value is .5, and at frame 10 the average red value is .6, the above expression multiplies frame 10 by .83333 (.5/.6) to arrive at .5 again.

Method 3: Using a Mult Node to Correct All Three Channels

Method 2 assumes uniform variation across all three channels, which is probably wishful thinking. On the other hand, it's fast and easy. A more accurate approach might be to feed similar expressions into the RGB channels of a *Mult* node.

The following expressions are entered into the red, green, and blue parameters of a *Mult* node:

```
(PixelAnalyzer1.area1AverageRed@@1)/PixelAnalyzer1.area1AverageRed
(PixelAnalyzer1.area1AverageGreen@@1)/PixelAnalyzer1.area1AverageGreen
(PixelAnalyzer1.area1AverageBlue@@1)/PixelAnalyzer1.area1AverageBlue
```

This adjusts each channel according to the *PixelAnalyzer* node's analysis of each channel of frame 1.

PixelAnalyzer Viewer Shelf Controls

The *PixelAnalyzer* node has the following Viewer shelf controls:

Button		Description
	Analyze Forward/ Backward	Analyzes to beginning or end of a clip. Once you've defined the analysis area, click one of these controls to analyze the image.
	Offset From Search Region	Click this button to offset the analysis area from its initial position.
	Path Display	Toggle analyze area display.

Parameters

The *PIxelAnalyzer* node has the following parameters:

analysisRange
The frame range over which the analysis is performed.

limitProcessing
When turned on, this button limits image updating to the area inside the analysis boxes.

areaName
The name of each analysis area you create. The associated parameter names for each analysis area update whenever the name is changed.

areaAverage
The average value of every pixel within the analysis area over the span of the analysisRange.

areaMinimum
The minimum value found within the analysis area over the span of the analysisRange.

areaMaximum
The maximum value found within the analysis area over the span of the analysisRange.

areaWindowParamameters subtree
The areaWindowParameters subtree contains parameters that define the size and location of the region that encompasses each analysis area, including the following:

- *areaX, Y:* The center of the analysis area. These coordinates define the location of the analysis area, and are the parameters to animate if you want to move the analysis area.
- *areaWidth, areaHeight:* The width and height of the analysis area.
- *areaVisible:* A slider toggle that makes the analysis area boundary box visible. This parameter is updated when one of the Analyze controls is clicked. This parameter corresponds to the Visibility button found next to the area name parameter.

Add, Delete, Save
These three buttons let you add, delete, and save new analysis areas.

Color-Correction Nodes

In general, Shake has three classes of color-correction nodes, which are located in the Color Tool tab.

Atomic-Level Correctors

Each atomic-level corrector node (*Add*, *Brightness*, *Clamp*, *Lookup*, and so on) applies a single basic mathematical function to your color, such as add, multiply, gamma, and basic lookup curve functions. These functions usually concatenate for speed and accuracy, and are fast ways to manipulate your data for a wide variety of purposes. In the Color Tool tab, each node's icon consists of a graph that represents its function. In the following illustration, notice the difference in the graph curves for the *Clamp*, *Compress*, and *Expand* nodes.

Utility Correctors

The utility correctors are nodes that prepare an image for other types of operations. Typically, these include *ColorSpace*, *ColorX*, *Lookup*, *MDiv*, *MMult*, *Reorder*, *Set*, *SetAlpha*, *SetBGColor*, and *VideoSafe*. For more sophisticated functionality, there is a certain degree of programmability in the *ColorX* and the *Lookup* nodes, which allow you to create expressions that affect the image.

Consolidated Correctors

The consolidated correctors (*ColorCorrect*, *ColorMatch*, *ColorReplace*, *HueCurves*) are your primary tools for tasks such as matching skin tones, shadow levels, spill suppression, and so on. Functions performed by the atomic-level nodes are also performed by these, but are combined with many other tools which provide more control over the result.

The following table includes general guidelines to help you understand why there are so many nodes in the Color Tool tab. For example, you can perform an add operation either with an *Add* node, or with a *ColorCorrect* node. In either case, the result is the same, but for simple operations the *Add* node may be more straightforward to use, with the additional benefit that it concatenates with many other color-correction operators.

The following is a basic list of the color-correction nodes and how they are useful:

Node	Description
Add	Raises or lowers all colors evenly. This is the only legal color correction in log space.
AdjustHSV	Shifts the entire Hue, Saturation, or Value range.
Brightness	Multiplies the R, G, and B channels by a single value.
Clamp	Good for clamping off values that go above or below a desired level.
ColorCorrect	A multi-functional color-correction node that lets you make many different adjustments to an image. It includes a premultiplied input flag.
ColorMatch	Matches the lows, midtones, and highlights in one image to those in another.
ColorReplace	Replaces one color in an image with another. It also makes a better chroma keyer than the *ChromaKey* node.
ColorSpace	Converts an image into a different color space, such as RGB, HSV, CMY, and so on.
ColorX	Lets you apply pixel-based mathematical expressions to an image.
Compress	Squeezes the range of color in an image up and down, and is particularly useful for creating fog effects with a *DepthKey* mask.
ContrastLum	Adjusts contrast. Use *Lum* to preserve your color levels.
ContrastRGB	Adjusts the contrast of individual channels, including the alpha channel.
Expand	Another levels adjuster that raises or lowers your low and high points.
Fade	Adjusts the opacity of an image. This node works the same as using *Mult* with identical values in all four channels.
Gamma	Gamma adjustments affect midtone color values while leaving the white and black points of the image unaltered.

Node	Description
HueCurves	A node that isolates and adjusts an image based on its hue. Ideal for spill suppression.
Invert	Turns black to white and vice versa. Works best on normalized images between 0 and 1.
LogLin	Performs logarithmic to linear, and linear to logarithmic color space conversion.
Lookup/HLS/HSV	Applies lookup expressions or curve manipulation to your image. It is faster than *ColorX* for non-pixel based lookups.
LookupFile	Pulls a lookup table from a file.
MDiv	Used to unpremultiply an image by its alpha channel.
MMult	Used to premultiply an image by its alpha channel.
Monochrome	Gives you weighted control over desaturating an image to make it black and white.
Mult	Multiplies the values in each color channel of an image. The R, G, and B color channels can be adjusted individually.
Reorder	Swaps channels within an image. See also Layer–*Copy* and Layer–*SwitchMatte* to copy channels from other images.
Saturation	Controls the saturation levels. Saturation can either be boosted or decreased.
Set	Sets a channel to a constant level, replacing whatever values were previously in that channel. Channels can be adjusted together or separately.
SetAlpha	Sets the alpha level to a constant value, replacing whatever values were previously in that channel. This node also crops the Infinite Workspace.
SetBGColor	Sets the color outside of the DOD.
Solarize	A misguided *Invert* function. Good for Beatles album covers.
Threshold	Clips the color values in an image. Color channels can be clipped individually.
VideoSafe	Clips luminance or saturation values that are above the broadcast-legal range for video.

Atomic-Level Functions

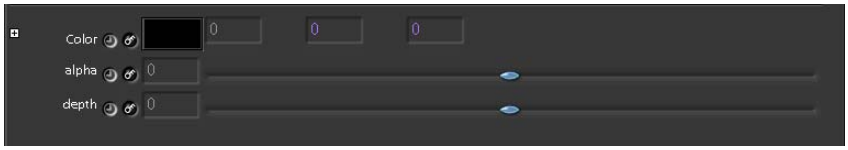
The term *atomic-level* is used because each of these nodes applies a single mathematical operation to the affected image. Because of their simplicity, they are easy to work with. They are also ideal for use on the command line.

Add

The *Add* node adds color to the R, G, B, A, or Z channel. Specifically, this node adds color to black areas, including those beyond the image frame, in case you move the image later. Any correction that occurs outside of the DOD can be corrected with the *SetBGColor* node. Shake's color is described in a range of 0 to 1, so adding -1, -1, -1 makes your image completely black. If you add the fifth value, depth, you effectively add a Z channel with your add value to the image.

Parameters

This node displays the following controls in the Parameters tab:



Color
A color control that lets you pick a color to add.

alpha
Adds to or subtracts from the alpha channel.

depth
Adds to or subtracts from the Z channel.

Brightness

The *Brightness* node is simply a multiplier on the RGB channels. It differs from *Fade* in that *Fade* also affects the alpha channel. For individual channel control, use *Mult*.

You can also use this node to convert your image to a single-channel alpha image by using a brightness of 0.

Parameters

This node displays the following control in the Parameters tab:

value
A slider you use to specify the value to multiply the image by. This control simultaneously multiplies the RGBA channels.

Clamp

The *Clamp* node clamps off values above and below a certain range. For example, if your redHi value is .7, any value above that is set to .7. You can isolate the red, green, blue, and alpha values.

Parameters

This node displays the following controls in the Parameters tab:

Low Color

The low value that all values are set to if they are less than this number. 0 is no change.

aLo

A low value control for the alpha channel.

High Color

The high value that all values are set to if they are more than this number. A value of 1 equals no change.

aHi

A high value control for the alpha channel.

Compress

The *Compress* node squeezes the image to fit within the Lo and Hi range you set. Unlike *Clamp*, the entire image is modified because it is fit between the two points.

Parameters

This node displays the following controls in the Parameters tab:

Low Color

The new lowest value in the image. A value of 0 equals no change.

aLo

A low value control for the alpha channel.

High Color
The new highest value in the image. A value of 1 equals no change.

aHi
A high value control for the alpha channel.

ContrastLum

The *ContrastLum* node applies a contrast change on the image, with a smooth falloff on both the low and high ends. You can also move the center of the contrast curve up and down (for example, move it down to make your overall image brighter). Note that this contrast is based on the luminance of the pixel, so it balances red, green, and blue, and you therefore do not shift your hue. So, an image with strong primary colors looks different than an image with the same values in a *ContrastRGB* node, which evaluates each channel separately.

Also note that a roll-off is built into the contrast node, giving you a smooth falloff at the ends of the curve. This is controlled by the softClip parameter.

Parameters

This node displays the following controls in the Parameters tab:

value
The contrast value. A higher value means it pushes your RGB values toward 0 and 1. A value of 0 equals no change. A lower value is low contrast.

center
This is the center of the contrast curve. A lower value is a brighter image.

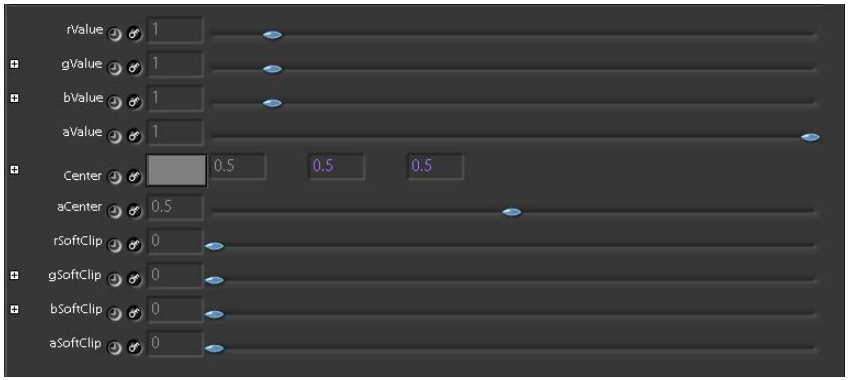
softClip
This controls the drop-off of the curve. When increased to 1, you have maximum smoothness of the curve.

ContrastRGB

The *ContrastRGB* node applies a contrast curve on each channel individually, so you can tune the channels separately. This differs from the *ContrastLum* node in that it only changes the pixel value according to its own channel. For a good example of how the functions differ, plug a *ColorWheel* node into both a *ContrastLum* and *ContrastRGB* node. Notice that the *ContrastLum* node is weighed away from the green values. This also means the *ContrastRGB* node runs the risk of shifting the hue as you adjust your channels.

Parameters

This node displays the following controls in the Parameters tab:



Value

The contrast value. A higher value means it pushes the RGB values toward 0 and 1. A low value is low contrast. A value of 0 represents no change.

Center

The center of the contrast curve. A lower value makes that channel brighter. A higher value makes the image darker. Generally, these values are between 0 and 1.

SoftClip

The roll-off value to give a smooth interpolation. A value of 0 equals no roll-off.

Expand

The *Expand* node squeezes the data between two points on the X axis of a graph of an image, increasing the amount of pure black and white (on a per-channel basis) in the image. Compress squeezes them on the Y axis of the image graph.

Parameters

This node displays the following controls in the Parameters tab:

Low Color

Pixels less than or equal to Lo value go to 0. At 8 or 16 bits per channel, pixels less than this value are clamped at 0.

aLo
A low color control for the alpha channel.

High Color
Pixels greater than or equal to Hi value go to 1. At 8 or 16 bits per channel, pixels greater than this value are clamped at 1.

aHi
A high color control for the alpha channel.

Fade

The *Fade* node multiplies the RGBA channels. It differs from *Brightness* in that *Fade* also affects the alpha channel. For individual channel control, use *Mult*.

A neat trick is to fade to 0. This effectively deactivates all nodes above the *Fade* node in the tree.

Note: Premultiplication is not a concern with the *Fade* node, since *Fade* treats the RGBA channels evenly.

Parameters

This node displays the following control in the Parameters tab:

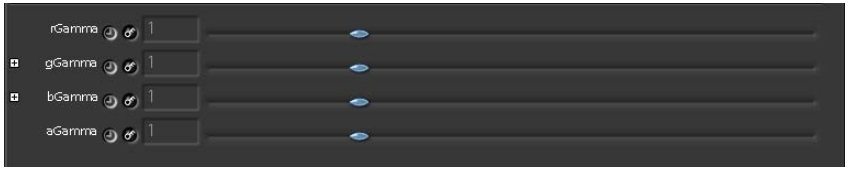
Value
The brightness factor. Greater than 1 increases brightness; less than 1 darkens it. A value of 0 is complete black.

Gamma

The *Gamma* node applies a gamma to your image. A value of 1 equals no change. Shake's ability to use expressions can be particularly useful here; for example, to invert the gamma of 1.7, type "1/1.7" as your gamma value. Typing ".588" isn't nearly as slick.

Parameters

This node displays the following controls in the Parameters tab:



rGamma
The red gamma value.

gGamma
The green gamma value.

bGamma
The blue gamma value.

aGamma
The alpha gamma value.

Invert

The *Invert* node inverts the color curve, so white becomes black and black becomes white. A predominantly yellow image becomes predominantly blue if the red, green, and blue (RGB) channels are selected in the channels field.

Invert also works on the Z channel, but assumes the Z is normalized, for example, between 0 and 1. If this is not the case, you have an unpredictable result. If you need to invert a non-normalized Z channel, use *ColorX* with a formula similar to the following in the Z channel:

MaxZRange-z

Parameters

This node displays the following control in the Parameters tab:

channels
The channels you want to invert. You can use r, g, b, a, and/or z. To use multiple channels, list them out. For example, *rgz* inverts the red, green, and Z channels.

Monochrome

The *Monochrome* node turns any image black and white. Unlike the *Saturate* node, you can adjust the relative brightness that each channel contributes to the final BW image. The default values are weighed according to the human eye's different sensitivities to red, green, and blue, but you can override these weights to simulate other effects, such as how various black and white film stocks expose an image.

This node reduces a three-channel image (RGB) to a one-channel image (BW), and a four-channel image (RGBA) to a two-channel image (BWA). If the three color channels are identical, it is more efficient to use a *Reorder* with a value of "rrr," since only the red channel is read in. *Monochrome* reads all three channels in, and therefore has more I/O activity.

Parameters

This node displays the following control in the Parameters tab:

Weight

The default R, G, and B values are set according to the human eye's sensitivity to color, but you can balance the colors differently to push a certain channel. The default values are:

- R = .3
- G = .59
- B = .11

Mult

The *Mult* node multiplies the R,G, B, A, or Z channels. To uniformly increase the red, green, blue, and alpha channels, use the *Fade* node. To affect red, green, blue, but leave alpha at the same amount, use *Brightness*.

Parameters

This node displays the following controls in the Parameters tab:

Color

The value by which the incoming R, G, and B channel pixels are multiplied.

alpha

The value by which the incoming alpha channel is multiplied.

depth

Multiplying the Z value does not necessarily add a Z channel like the *Add* node.

Saturation

The *Saturation* node changes the saturation value of an image. Unlike the Monochrome node, *Saturation* increases or decreases the saturation of the image weighing all color channels equally.

Note: You can also use *Monochrome* to desaturate an image, giving different weights to each of the color channels.

Parameters

This node displays the following control in the Parameters tab:

value

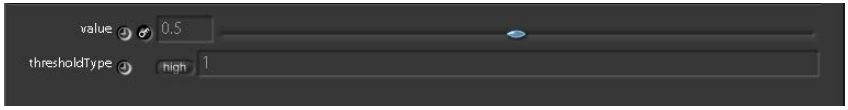
A slider that defines the saturation multiplier.

Solarize

The *Solarize* node is a partial inverse that reverses the high or low end, depending on the value of the hi/lo flag. With values above ("hi," or 1) or below ("lo," or 0), the thresholds are reversed. The resulting images are similar to a photographic effect popular in the 1960s, where the print is exposed to light during development and results in an image with blended positive and negative value ranges. It gives a metallic effect to color images.

Parameters

This node displays the following controls in the Parameters tab:



value

The point at which the image is inverted.

thresholdType

Determines whether numbers must be higher or lower than the Value parameter to be affected by the Solarize operation.

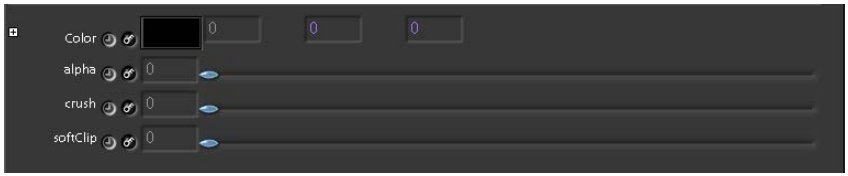
- 0 means "lo," or below the threshold value that the color is inverted.
- 1 means "hi," or above the threshold.

Threshold

The *Threshold* node cuts channels off at a certain value, and turns everything below that cut-off value to 0. Each channel can have its own separate cut value. By using the crush parameter, you can boost everything above the value 1.

Parameters

This node displays the following controls in the Parameters tab:



Color
Anything below this value goes to black.

alpha
All areas in the alpha channel below this value go to black.

cCrush
If this is set to 1, everything above the cut-off values goes to 1.

softClip
Provides a roll-off value.

Utility Correctors

These tools are more applicable for preparing data for other operations.

ColorSpace

The *ColorSpace* node converts an image from one color space to another color space. After the image is converted, you can use color-correction functions that operate in the new color space logic for interesting effects. For example, if you place a *ColorSpace* node to convert from rgb to hls (hue, luminance, saturation), and then apply a Color–*Add* node, the *Add* node's red channel shifts your hue, instead of the red channel.

The optional r, g, bWeight arguments only affect rgb to hls, or hls to rgb, conversions.

For command-line use, and compatibility with Shake version 2.1 scripts or earlier, use the dedicated space conversion functions *CMYToRGB*, *HLSToRGB*, *HSVToRGB*, *RGBToCMY*, *RGBToHLS*, *RGBToHSV*, *RGBToYIQ*, *RGBToYUV*, *YIQToRGB*, and *YUVToRGB*. These functions do not have arguments, with one exception: The HLS functions have optional r, g, and bWeight parameters.

Parameters

This node displays the following controls in the Parameters tab:

inSpace

Selects the incoming color space.

outSpace

Selects the output color space. For example, if you use one *ColorSpace*, you probably use rgb as your inSpace, and then something like hsv to convert it to hue/saturation/value space. After applying your operations, you usually apply a second *ColorSpace* node, with hsv as your inSpace and rgb as your outSpace.

luminanceBias

The weighing of the three channels for the luminance calculation in conversions involving HSL. Luminance differs from value in that luminance calculates brightness based on the human eye's perception that green is brighter than an equal value in the blue channel.

- rWeight
- gWeight
- bWeight

ColorX

The *ColorX* node modifies each pixel in the image according to an expression that you supply. *ColorX* is normally slower than a dedicated, optimized node such as *Mult* or *Add*. Use dedicated operators whenever possible. You can also set up complex rules inside the node (see below).

For more information on using expressions, see Chapter 31, "Expressions and Scripting," on page 935.

Expressions can use the following variables:

- The variables r, g, b, a, and z refer to the value of the original channels (red, green, blue, alpha, and Z).
- The variables x and y are the coordinates of the pixel.
- The variables width and height are the width and height of the image.
- The variable time is the current frame number (time).

Many operators can be represented by an arithmetic expression, such as reordering, color correction, and gradient generation, or even circle drawing. Note that no spaces are allowed in the expressions, unless you can use quotes for more explicit grouping.

Task	Example: Each value field can be filled independently in the interface—these are all command-line examples.
Reordering	*shake bg.iff -colorx b r a g*
Red correction	*shake bg.iff -colorx "r*1.2"*
Red and blue correction	*shake bg.iff -colorx "r*1.2" g "b*1.5"*
Same expression for red, green, and blue	*shake bg.iff -colorx "(r+g+b)/3" -reorder rrr*
X gradient in matte	*shake -colorx r g b "x/width"*
Y gradient in matte	*shake -colorx r g b "y/height"*
Blue spill removal	*shake primatte/woman.iff -colorx r g "b>g?g:b"*
Random noise	*shake -colorx rnd(x*y) rnd(2*x*y) rnd(3*x*y)*
Turbulent noise (1 channel)	*shake -colorx turbulence2d(x,y,20,20)*
Clip alpha if Z is less than 20	*shake uboat.iff -colorx r g b "z<20"*
Clip alpha if Z is more than 50	*shake uboat.iff -colorx r g b "z>50"*
A smooth alpha gradient from Z units 1 to 70	*shake uboat.iff -colorx r g b "(z-1)/70"*

LogLin

The *LogLin* node is typically used to handle logarithmic film plates, such as Cineon files from a film scanner, or when writing files out for film recording. It converts the images from the logarithmic color space to the linear color space for accurate compositing. You can then use the node at the end of your node tree to convert the images back into logarithmic space for film scanning. You can also use this around nodes that only work in 8 or 16 bits, functions such as *Primatte*, *Keylight*, or other plug-ins from third parties. *Ultimatte* is the exception, as it maintains float values. For a full description of this process, see "The Logarithmic Cineon File" on page 437.

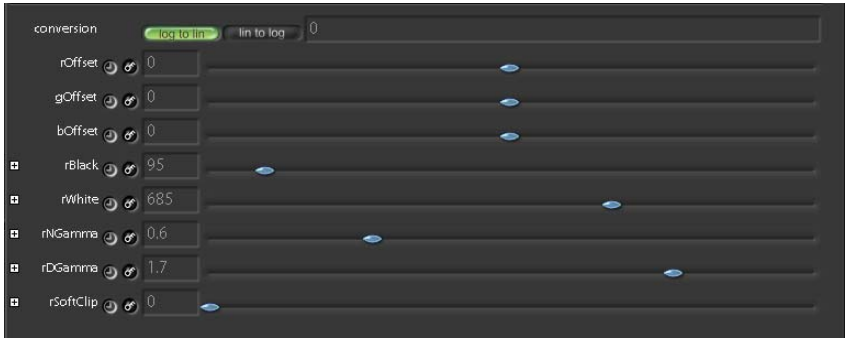
The *LogLin* color parameters are linked together by default, so gBlack and bBlack reflect the rBlack value. You can of course adjust these to further color correct your scanned plates.

Note: This node only does color correction—it does not change your bit depth or your file type. When Shake imports the Cineon files, typically a 10-bit file, it automatically promotes the files to 16 bits. This process has nothing to do with the color correction.

The default values are supplied by Kodak—if you apply a *LogLin* in Shake, you should get the same visual result as if you plugged in the same numbers into any other software package's logarithmic converter. The range of the offset and black and white points is 0 to 1023, the range of a 10-bit file (8 bit is 0 to 255, 9 bit is 0 to 511). Every 90-point adjustment of these values is equivalent to a full f-stop of exposure.

Parameters

This node displays the following controls in the Parameters tab:



Conversion

This parameter describes whether you convert from log to linear space, or linear to log space. 0 is log to lin, 1 is lin to log.

rOffset

This control offsets the red color channel.

gOffset

This control offsets the green color channel.

bOffset

This control offsets the blue color channel.

rBlack

This sets the black point. The default value is 95.

rWhite

This sets the white cutoff point. The default value is 685.

rNGamma
Generally, this number is not touched. The .6 is an average of the response curves, and may differ from stock to stock and even channel to channel. You can look it up on Kodak's website—see *products/Film/Motion Picture Film*, and then check the characteristic curves.

rDGamma
The display gamma, according to Kodak, to compensate for the monitor lookup table. This was set to neutralize the Cineon system's standard monitor setting. Its inclusion here is more of a heritage thing. It is highly recommended that you leave it at 1.7.

rSoftclip
The roll-off values on the white point. The default is 0, which gives a Linear break. By increasing this value, the curve is smoothed.

Lookup

The *Lookup* node performs an arbitrary lookup on your image. It is extremely flexible, allowing you to mimic most other color-correction nodes, and is generally much faster than the *ColorX* node.

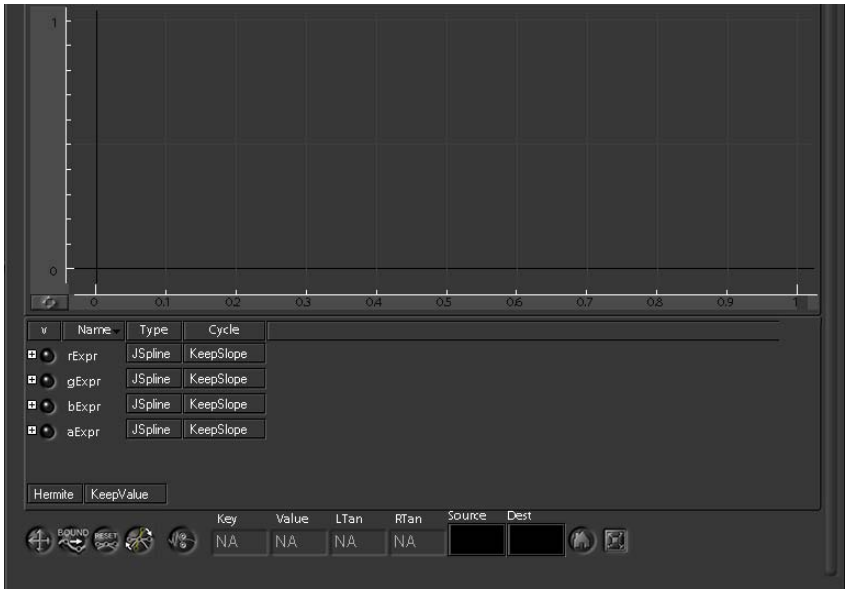
For information regarding the curve editor in *Lookup*, *LookupHSV*, and *LookupHLS*, see Chapter 10, "Parameter Animation and the Curve Editor," on page 291.

The *Lookup* is defined as a function f(x), where x represents the input color, ranging from 0 to 1. As you draw the graph of this function, x is on the X axis, and f(x) is on the Y axis.

Note: The *Lookup* node should not be used inside of macros.

Parameters

This node displays the following controls in the Parameters tab:



rExpr

Use this function to change the input red value, always represented by "x."

gExpr

Use this function to change the input green value, always represented by "x."

bExpr

Use this function to change the input blue value, always represented by "x."

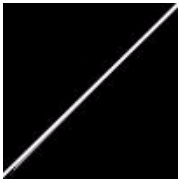
aExpr

Use this function to change the input alpha value, always represented by "x."

Sample Lookup Tables

The following table lists the *Lookup* equivalents of other Shake color-correction nodes.

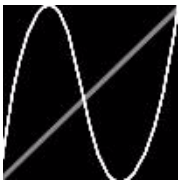
Function	Brightness	Invert
Math Expression	f(x) = x * value	f(x) = 1-x
Lookup Expression	x*1.5	1-x
Graph (white is result, gray is input)		

Function	Compress	Do Nothing
Math Expression	f(x) = x * (hi-lo) + lo	f(x) = x
Lookup Expression	"(x*.4)+0.3" (if lo = 0.3 and hi = 0.7)	"x"
Graph (white is result, gray is input)		

The following examples do custom lookups. The last two examples use Shake's curve formats, but use the Value mode (the V at the end of the curve name), and input x as the value. All "keyframes" are between 0 and 1, and can take any value.

When using the interface, this is the default behavior—click the Load Curve button in the Parameters tab to load the curve into the Curve Editor.

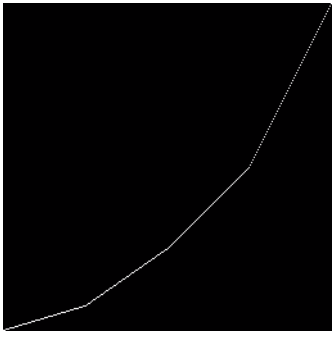
Function	Clipping	Dampening
Lookup Expression	x>.5?0:x	x*x
Graph (white is result, gray is input)		

Function	Spline Lookup	Linear Lookup
Lookup Expression	CSplineV(x,0, 0@0, 1@.25, 0@.75, 1@1)	LinearV(x,0, 0@0, 1@.25, 0@.75, 1@1)
Graph (white is result, gray is input)		

LookupFile

Use the *LookupFile* node to apply a lookup table to any image by reading a text file. The file should consist of an arbitrary number of rows, and each row can have three or four entries, corresponding to red, green, blue, and possibly alpha. Shake determines the range of the lookup to apply based on the number of rows in the file—with "black" always mapping to 0 and "white" mapping to (n-1), where n is the number of lines in the file. Therefore, if your file contains 256 rows, Shake assumes that your entries are all normalized to be in the range of 0 (black) to 255 (white). If you have 1024 lines in your file, then "white" is considered to be a value of 1023. Interpolation between entries is linear, so lookups with only a few entries may show undesirable artifacts. For example, the following simple five-line lookup file produces the following lookup curve:

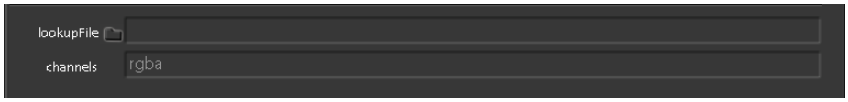
```
 0    0    0    0
.3   .3   .3   .3
 1    1    1    1
 2    2    2    2
 4    4    4    4
```



Because of this linear interpolation, you may want to instead use the standard *Lookup* node with lookups that do not have a large number of points.

Parameters

This node displays the following controls in the Parameters tab:



lookupFile

A text field where you enter the path to the lookup file.

channels

The channels to which the lookup operation is applied.

LookupHLS

The *LookupHLS* node performs exactly like *Lookup*, except it works on the HLS channels instead of RGB channels.

Parameters

This node displays the following controls in the Parameters tab:

sExpr

Use this function to change the input value, always represented by "x."

lExpr

Use this function to change the input value, always represented by "x."

hExpr
Use this function to change the input value, always represented by "x."

aExpr
Use this function to change the input value, always represented by "x."

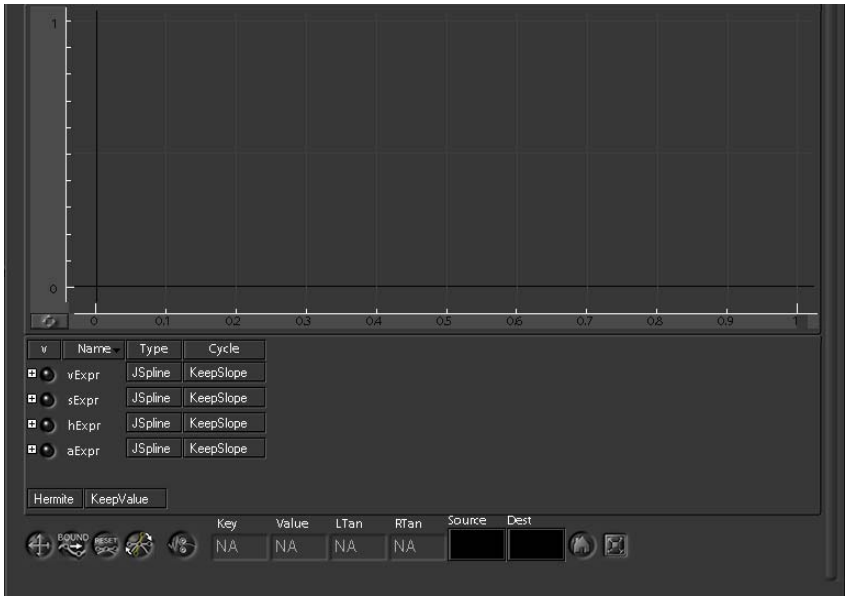
LookupHSV

The *LookupHSV* node performs exactly like *Lookup*, except that it works on the HSV channels instead of RGB channels.

Note: You cannot clone *LookupHSV* nodes in the node tree using the Paste Linked command.

Parameters

This node displays the following controls in the Parameters tab:



vExpr
Use this function to change the input value, always represented by "x."

sExpr
Use this function to change the input value, always represented by "x."

hExpr
Use this function to change the input value, always represented by "x."

aExpr
Use this function to change the input value, always represented by "x."

MDiv

The *MDiv* node divides the color channels by the alpha channel.

When you color correct a rendered (premultiplied) image, first apply an *MDiv* node to the image to make the image a non-premultiplied image, perform the color correction, and then add an *MMult* node to return the image to its premultiplied state. For more information on premultiplication, see "About Premultiplication and Compositing" on page 421.

Parameters

This node displays the following control in the Parameters tab:

ignoreZero

Tells Shake to ignore pixels with an alpha value of 0.

- 0 = Divide entire image.
- 1 = Ignore zero-value pixels.

MMult

The *MMult* node multiplies the color channels by the matte.

This node is used to premultiply an image. When compositing with the *Over* node, Shake expects all images to be premultiplied. Premultiplication is usually automatic with 3D-rendered images, but not for scanned images. Also, use this node to color correct a 3D-rendered image. Add an *MDiv* node to the image, perform your color corrections, and then add an *MMult* node into an *Over* operation. For more information on premultiplication, see "About Premultiplication and Compositing" on page 421.

Note: You can also choose to not insert a *MMult* node, and instead enable preMultiply in the *Over* node's parameters.

Parameters

This node displays the following control in the Parameters tab:

ignoreZero

Tells Shake to ignore pixels with an alpha value of 0.

- *0* = Multiplies entire image.
- *1* = Ignore zero-value pixels.

Reorder

The *Reorder* node lets you shuffle channels. The argument to this command specifies the new order. A channel can be copied to several different channels. The letter "l" refers to the luminance pseudo-channel which can be substituted in place of the RGBA. If an expression is on a channel that does not exist, Shake creates the channel. You can use the Z channel as well. For example:

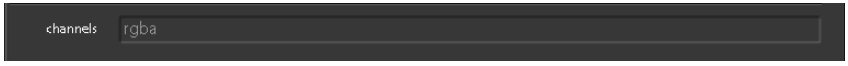
```
shake -reorder zzzz
```

places the Z channel into the RGBA channels for viewing.

To copy a channel from another image, use the *Copy* node.

Parameters

This node displays the following control in the Parameters tab:



channels

Indicates the new channel assignment. You can use any of the following:

- *r:* Set the pixels of this channel to the values of the red channel.
- *g:* Set the pixels of this channel to the values of the green channel.
- *b:* Set the pixels of this channel to the values of the blue channel.
- *a:* Set the pixels of this channel to the values of the alpha channel.
- *z:* Set the pixels of this channel to the values of the Z channel.
- *l:* Set the pixels of this channel to luminance of RGB.
- *0:* Set the pixels of this channel to 0.
- *1:* Set the pixels of this channel to 1.
- *n:* Remove this channel from the active channels.

Set

The *Set* node sets selected channels to a constant value. Alpha or depth channels can be added to images with *Set*. For example, *-set z .5* puts in a Z plane with a value of .5.

Parameters

This node displays the following controls in the Parameters tab:

channels
The channels to be set.

value
The value of the channel.

SetAlpha

The *SetAlpha* node is simply a macro for *Set* and *Crop* that sets the alpha to 1 by default. It exists because in the Infinite Workspace, color corrections extend beyond the frame of an image. By using the *Crop* in the macro, the *Set* node is cut off, making the image ready for transformations.

To remove the alpha channel from an image (turn an RGBA image into an RGB image), set the alpha value to 0.

Parameters

This node displays the following control in the Parameters tab:

alpha
From 0 to 1, this is the alpha value of the output image. By default, it is 1.

SetBGColor

The *SetBGColor* node sets selected channels to the selected color outside of the Domain of Definition (DOD). For example, if you create an Image–*RGrad*, the green DOD box appears around the *RGrad* image in the Viewer. Everything outside of the DOD is understood to be black, and therefore does not have to be computed. To change the area outside of the DOD, attach a *SetBGColor* to the node and change the color.

This node is often used for adjusting keys, as the keyer may be pulling a bluescreen, and therefore assigns the area outside of the DOD, which is black, as an opaque foreground. If the element is scaled down and composited, you do not see the background. To correct this, insert a *SetBGColor* before the keyed element is placed in the composite, for example, *ChromaKey* > *SetBGColor* > *Scale* > *Over*.

Parameters

This node displays the following controls in the Parameters tab:

mask

Specifies the channels that are reset.

Color

The new value to set the red, green, and blue channels to.

alpha

The new value to set the alpha channel to.

depth

The new value to set the Z channel to.

VideoSafe

For information on the *VideoSafe* node, see "*VideoSafe*" on page 208.

Consolidated Color Correctors

The consolidated color-corrector nodes are more complex than the other nodes. They are usually inappropriate for use on the command line, and have unique interfaces.

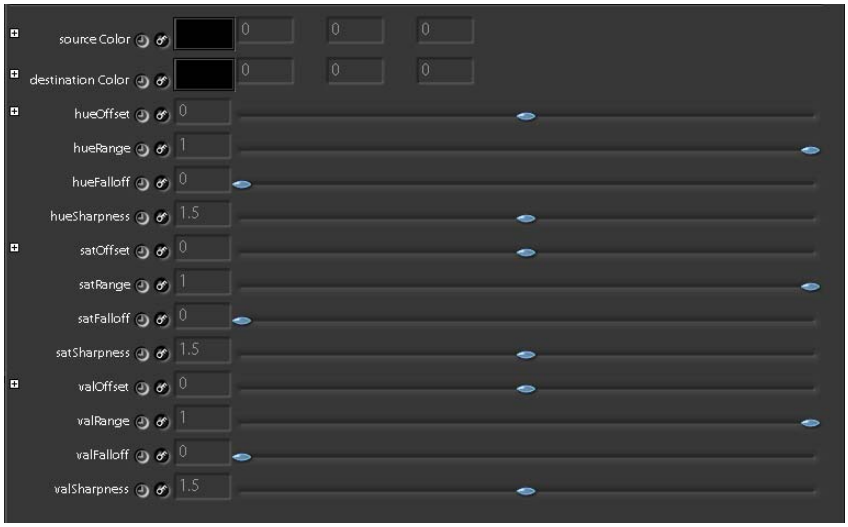
AdjustHSV

The *AdjustHSV* node takes a specified color, described by its HSV values, and offsets the color. For example, you can take a red spaceship and turn it blue without affecting the green alien on it. It works similarly to the *ChromaKey* node.

To change a color, scrub with the Color Picker to isolate its hue, saturation, and value. Next, scrub with the target Color Picker. For example, if you have a hue of 0 (red), and enter a hueOffset of .66, you slide it to blue. The Range, Falloff, and Sharpness sliders help control how much of a range you capture.

Parameters

This node displays the following controls in the Parameters tab:



sourceColor

These color controls let you select the HSV values of the target color you want to change.

destinationColor

The color you want to use as the replacement for the color value selected as the sourceColor.

hueOffset

A value that is added to the hue of the selected destination Color, thereby changing the color.

hueRange

The range of hue that is added to the HSV value selected in sourceColor to include a wider field of values.

hueFalloff

The amount of falloff from the affected amount of hue to the unaffected amount of hue. A greater hueFalloff value includes more color values at the edges of the hueRange.

hueSharpness

The drop-off curve of Falloff, creating a smoother or sharper transition between affected and unaffected regions of the image.

- 0 = linear drop-off
- 1.5 = smooth drop-off

satOffset
This is what is added to the saturation of the selected destination Color, thereby changing the intensity of the color.

satRange
The range of saturation that is added to the HSV value selected in sourceColor to include a wider field of values.

satFalloff
The amount of falloff from the affected amount of saturation to the unaffected amount of saturation. A greater satFalloff value includes more saturated values at the edges of the satRange.

satSharpness
The drop-off curve of Falloff, creating a smoother or sharper transition between affected and unaffected regions of the image.
- 0 = linear drop-off
- 1.5 = smooth drop-off

valOffset
A value that is added to that of the destination Color.

valRange
The range of value that is added to the HSV value selected in sourceColor to include a wider field of values.

valFalloff
The amount of falloff from the affected amount of value to the unaffected amount of value. A greater satFalloff value includes more saturated values at the edges of the satRange.

valSharpness
The drop-off curve of Falloff, creating a smoother or sharper transition between affected and unaffected regions of the image.
- 0 = linear drop-off
- 1.5 = smooth drop-off

ColorCorrect

The *ColorCorrect* node combines *Add*, *Mult*, *Gamma*, *ContrastRGB*, *ColorReplace*, *Invert*, *Reorder*, and *Lookup* in one node, and also gives you the ability to tune the image in only the shadow, midtone, or highlight areas.

You should keep in mind that the *ColorCorrect* node breaks concatenation with other color-correction nodes in the tree.

Note: The *ColorCorrect* node should not be used inside of macros.

The ColorCorrect Subtabs

The following table describes the *ColorCorrect* Parameter subtabs.

Subtab	Description
Master	Applies the same correction to the entire image.
Low Controls	Applies the correction primarily to the darkest portion of the image; the correction falls off as the image gets brighter.
Mid Controls	Applies the correction primarily to the middle range of the image.
High Controls	Applies the correction primarily to the highlights of the image; the correction falls off as the image gets darker.
Curves	Manual correction of the image using curves.
Misc	Secondary color correction, as well as invert, reorder, and premultiplication control.
Range Curves	Display of the different image ranges (shadows, midtones, highlights), their control curves, and the final concatenated curve of the color correction.

Note: You can only view one subtab at a time.

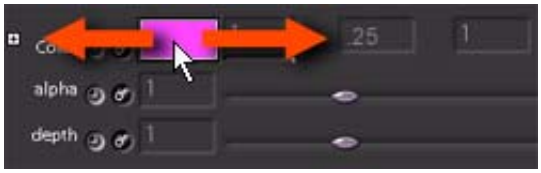
The Master, Low, Mid, and High Color Controls Tabs

The first four color control tabs (Master, Low, Mid, and High Controls) are identical except for the portion of the image you are modifying. Each has the same matrix to Add, Multiply (Gain), and apply a Gamma to the RGB channels, as well as apply contrast on a per-channel basis (Contrast, Center, SoftClip).

To tune the color with the matrix, do one of the following:

- Numerically enter a value in the RGB value fields.
- Use the slider below each value field.

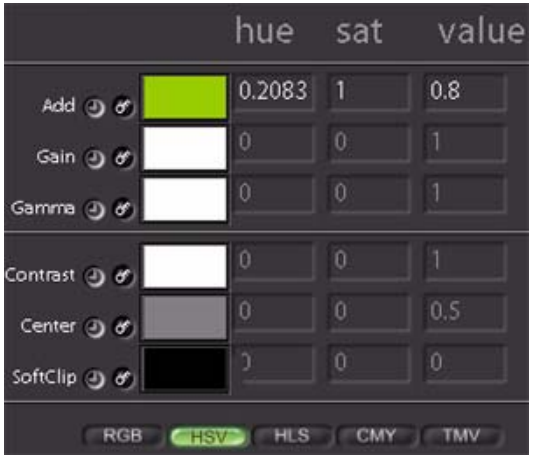
- Click the Color control, then select a color from the Viewer or the ColorWheel (in the Color Picker).
- Use the Virtual Color Picker. Press the relative key, R (red), G (green), B (blue), H (hue), S (saturation), V (value), L (luminance), M (magenta), or T (temperature), and drag left or right on the parameter line. (You can do this anywhere on the line, not just over the Color control.)



- To group the R, G, and B sliders, press V (value) and drag left or right.

  As an example, press R and drag right over the Add line. This adds to the red channel. When the color control is bright red, press H and drag left and right. The hue of the color shifts. This technique only modifies the color that is Added, Multiplied, and so on. So, dragging a color control while pressing S (saturation) does not decrease the saturation of the image, but only the saturation of the color that you are adding (or multiplying) to the image.

The bottom portion of the tab contains buttons to toggle the channels from RGB display to a different color space model. You can display RGB, HSV, HLS, CMY, and TMV. For example, if the current image is displayed in the RGB color model, click HSV and the numbers are converted to HSV space. Notice the Add color does not change—only the numerical value.



Note: The "TMV" color space is Temperature/Magenta-Cyan/Value.

When the *ColorCorrect* node is saved into a script, the values are always stored in RGB space.

Each color picker has an Autokey and View Curves button associated with all three channels for that parameter. All three channels are treated equally.

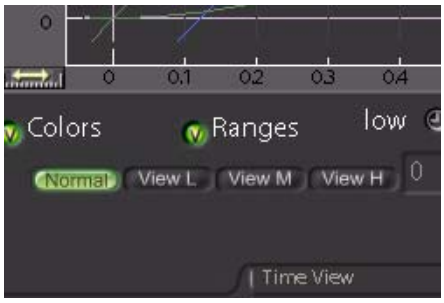
Working With Low, Mid, and High Ranges

The following section discusses the differences in working with low, mid, and high color ranges in the *ColorCorrect* node. The first image is the original image.

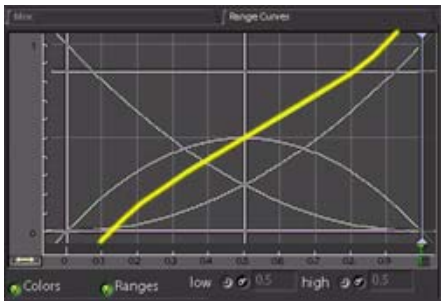
Thanks to Skippingstone for the use of images from their short film *Doppelganger*.

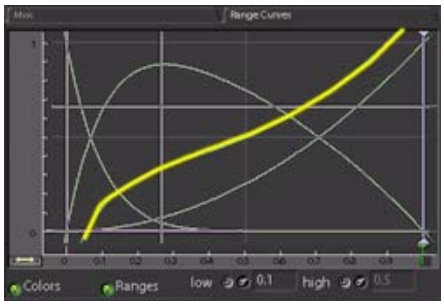
In the following examples, a Gain (multiply) of 2 is applied to each channel. The first example multiplies all pixels by 2. The pure blacks stay black, the whites flare out. However, when the gain is applied to the Low areas (the shadows), although the pure blacks stay black, the areas just above 0 are raised into the mid range, and this reduces the apparent contrast. A higher value solarizes the image. In the following images, a gain of 2 is applied in the Master tab, the Low, Mid, and the High tabs.

You can control the range of the image that is considered to be in the shadows, midtones, and highlights in the Range Curves subtab. This tab displays your final color lookup operator as a curve, your mask ranges (to turn on the display, click the Ranges button at the bottom), and controls for the center of the low and high curves. Also, you can toggle the output from the Normal, corrected image to a display of the Low areas boosted to 1 and all else black; the Mid areas boosted to 1 and all else black; or the High areas boosted to 1 and all else black. A colored display is used, rather than a display based on luminance, since different channels have different values. In the following, the range viewer controls, with Low, Mid, and High, are selected.



To control the mask areas, turn on the Ranges curve display at the bottom of the Range Curves tab. The left image below shows the default ranges. A curve of the final lookup is displayed in this illustration as a yellow line for clarity. Notice that the Low and High range curves' (gray curves sloping in from left and right) centers are set at .5. If you adjust the low or high values, you modify that range, as well as the mid-range curve. For example, the second image shows what happens when low is set down to .1. Notice the Low and Mid curves shift left, but the High curve remains unaffected.

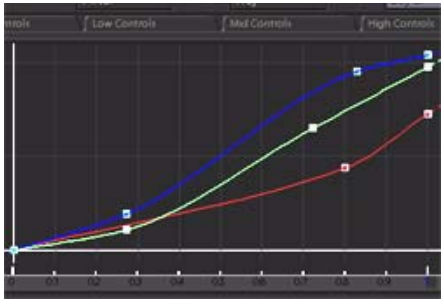




The Curves Tab

The Curves tab allows you to apply manual modifications to the lookup curve. Although this is generally used for manual adjustments, you can also apply functions using the standard lookup expressions.

Note: To insert a new control point, Shift-click a segment of the curve.



The Misc Tab

The Misc tab contains several functions.

- *Invert:* Invert uses the formula *1-x*, so float values may have odd results.
- *reorderChannels:* Enter a string to swap or remove your channels as per the standard *Reorder* method.

  For more information, see "*Reorder*" on page 657.

- *preMultiplied:* Enable preMultiplied if your image is premultiplied (typically, an image from a 3D render), and an *MDiv* is automatically inserted before the calculations, and an *MMult* is automatically added after the calculations.

  For more information on premultiplication, see "About Premultiplication and Compositing" on page 421.

- *Color Replace:* Use the Color Replace tools for secondary color correction, as per the *ColorReplace* node. In this example, the red color of the shutter is selected and replaced with blue. Sat Falloff is set to .2, Val Range to 0, and Val Falloff to 1.

Order of Calculations

Calculations are made in the following order:

- *MDiv (optional)*
- *ColorReplace*
- *Invert*
- *Lookup Curves*
- *Gamma*
- *Mult*
- *Add*
- *Contrast*
- *Reorder*
- *MMult (optional)*

ColorMatch

The *ColorMatch* node allows you to take an old set of colors (source color) in an image and match them to a new set (destination color). You can match the low, middle, and high end of the image. You can also perform Contrast, Gamma, and Add color corrections with Gamma as an inverse gamma to preserve highlights.

When you match color and use the Color controls, be aware of where you scrub. If you color correct an image and then feed it into the composite, you may have to jump down to view the color-corrected image to get proper source color; otherwise you pull in modified color. This occurs after you have already fed in the destination color, since they are linked to the source color. Therefore, a good workflow is to select all three source colors, and then select the destination colors. Another scrubbing technique is to ignore the node when scrubbing (select the node and press I in the Node View), then enable the node again when finished.

Parameters

This node displays the following controls in the Parameters tab:

lowSource
The low end of the RGB of the source color.

lowDest
The low end of the RGB destination color.

midSource
The middle of the RGB of the source color.

midDest
The middle of the RGB destination color.

highSource
The high end of the RGB of the source color.

highDest
The high end of the RGB destination color.

Contrast
Contrast values for the three channels.

Gamma
The Gamma values. Note this is an inverse gamma function, so you retain your highlights as you raise the gamma.

Mult
Multiplies the input image by this value.

Add
Adds color to the input image. Blacks are modified when this is raised.

min
Sets the clipping for the function.

max
Sets the clipping for the function.

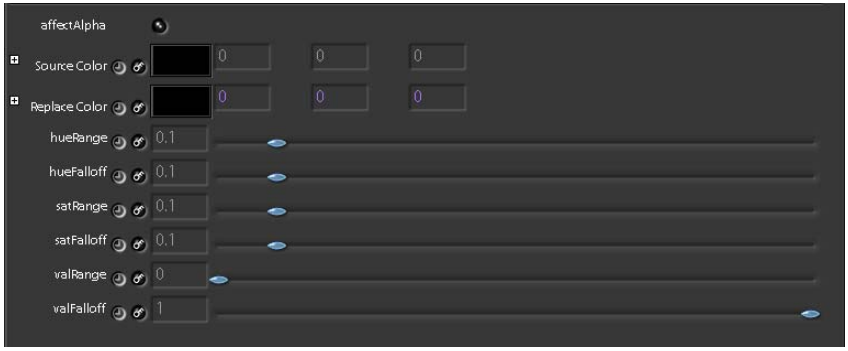
ColorReplace

The *ColorReplace* node allows you to isolate a color according to its hue, saturation, and value, and then replace it with a different color. Other areas of the spectrum remain unchanged. This is especially useful for spill suppression.

To pull a mask of the affected source color, enable affectAlpha in the *ColorReplace* parameters. To better understand the parameters, you can attach the *ColorReplace* node to a *ColorWheel* node to observe the effects graphically.

Parameters

This node displays the following controls in the Parameters tab:



affectAlpha
Toggles whether the alpha color is also adjusted by the color correction. If so, you can then easily use this as a mask for other operations.

sourceColor
These color controls let you select the target color you want to change.

destinationColor
The color you want to use as the replacement for the color value selected as the sourceColor.

hueRange
The range on the hue (from 0 to 1) that is affected. A range of .5 affects the entire hue range.

hueFalloff
This describes the amount of falloff from the affected to the unaffected hue region. A greater hueFalloff value includes more color values at the edges of the hueRange.

satRange
The range of the saturation from the Source color; 1 is the entire range.

satFalloff
This describes the amount of falloff from the affected amount of saturation to the unaffected amount of saturation. A greater satFalloff value includes more saturated values at the edges of the satRange.

valRange
Defines the range of value that is added to the HSV value selected in sourceColor to include a wider field of values. 1 is the entire range.

valFalloff
This describes the amount of falloff from the affected amount of value to the unaffected amount of value. A greater satFalloff value includes more saturated values at the edges of the satRange.

HueCurves

The *HueCurves* node allows you to perform various color corrections (Add, Saturation, Brightness) on isolated hues through the use of the Curve Editor. Typically, this tool is used for spill suppression, but you can also do color corrections once you understand how it is used.

Note: The *HueCurves* node should not be used inside of macros.

To color correct with the HueCurves node:

1. Find the hue of the area you want to color correct with the Color Picker. For example, if you have blue spill, the hue is approximately .66.
2. Load the parameter you want to use into the Curve Editor. For example, to load saturation into the Curve Editor, click the button to the left of "saturation" in the parameter list.

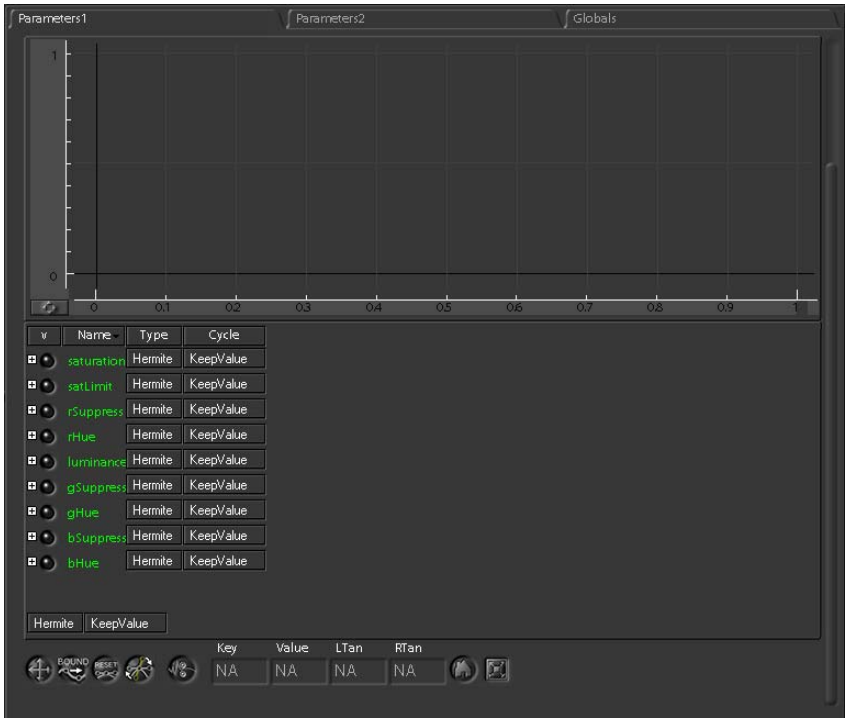
   When a parameter is loaded, the button is highlighted, and the curve appears in the Curve Editor.
3. Drag the control point near the hue (.66) down.

   The saturation is decreased in that particular hue, turning the pure blues to gray.

   By default, all curves have a value of 1 until you modify the value downward. Additionally, be careful with red-hued targets, as you may have to drag both the first and last control point on the curve.

Parameters

This node displays the following controls in the Parameters tab:



saturation
Removes saturation from the hue range you identify.

satLimit
Sets the limit for saturation values.

rSuppress
Removes red from the hue area you identify when you drag the control point downward.

rHue
Adds red to the hue range you identify.

luminance
Removes luminance from your area.

gSuppress
Removes green from the hue area you identify when you drag the control point downward.

gHue
Adds green to the hue range you identify.

bSuppress
Removes blue from the hue area you identify when you drag the control point downward.

bHue
Adds blue to the hue range you identify.

Other Nodes for Image Analysis

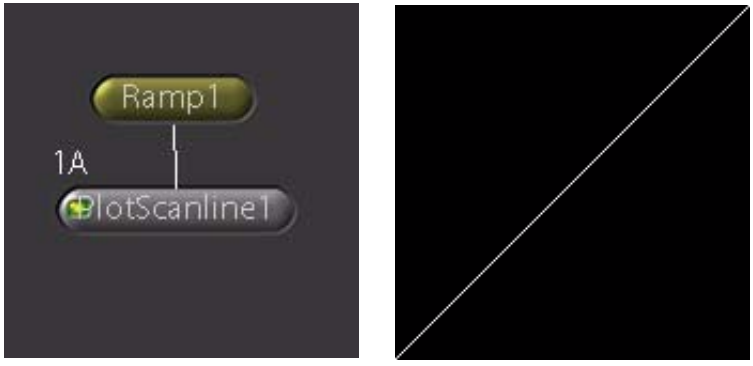
The *PlotScanline* and *Histogram* nodes, found in the Other tool tab, let you analyze images from the node tree to better understand how the data is being manipulated.

Note: You can also apply a *Histogram* or *PlotScanline* using the viewer scripts.

Using the PlotScanline to Understand Color-Correction Functions

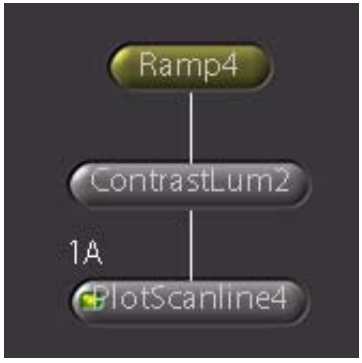
To better understand some of the Shake color-correction nodes, use the Other–*PlotScanline* node or the Plotscanline viewer script. The *PlotScanline* node, located in the Other Tool tab, looks at a single horizontal scanline of an image and plots the brightness value of a pixel for each X location.

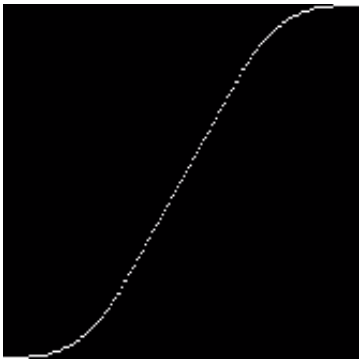
The most basic example of this is shown in the following illustration. The example begins with a simple horizontal gradient source image that varies linearly from 0 to 1. The *PlotScanline* resolution is set to 256 x 256 (for an 8-bit image).



The ramp ranges from black (on the left) to white (on the right). This is reflected in the graph as a linear line.

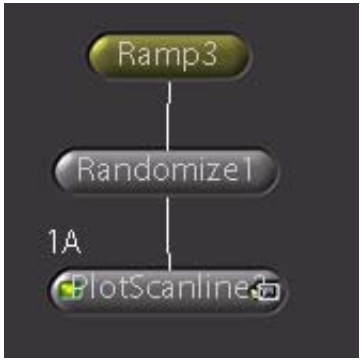
When a node such as *ContrastLum* is inserted above the *PlotScanline* node, you can begin to understand the node. In the *ContrastLum* node, value is set to 1.5 and the center and softClip parameters are adjusted.

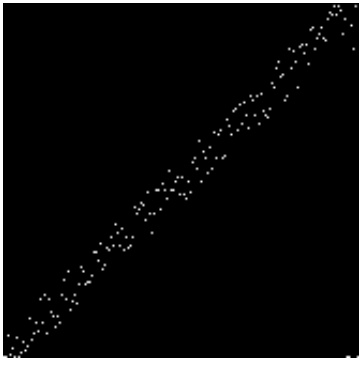




The effect on the ramp is reflected in the plot.

This also works for non-color correctors, and makes it an interesting analysis tool for the Filter–*Grain* or Warp–*Randomize* node.





PlotScanline and Histogram Viewer Scripts

You can also use the PlotScanline and Histogram viewer scripts to observe image data, but these are applied directly on your image. To load the parameters, right-click the Viewer Script button, then choose a Load Viewer Script Controls option from the shortcut menu.

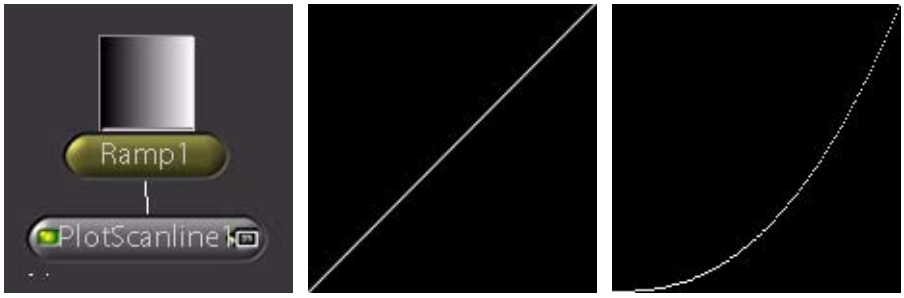
PlotScanline

The *PlotScanline* node is an analysis tool that examines a line of an image and graphs the intensity of each channel per X position. It greatly helps to determine what a color-correction node is doing. Although it can be attached to any image for analysis, it is often attached to a horizontal *Ramp* to observe the behavior of a color correction. Switch the Viewer to view the alpha channel (press A in the Viewer), and see the behavior of the alpha channel.

The following are some examples of using the *PlotScanline* node.

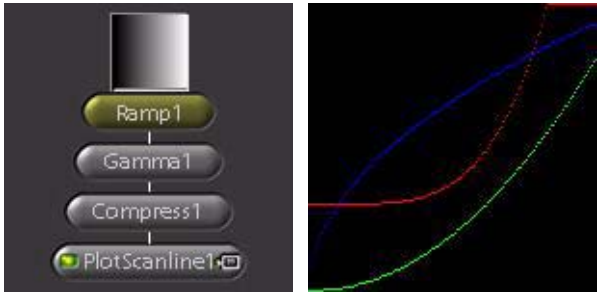
Example 1

A 256 x 256 8-bit black-and-white *Ramp*. Since there is a smooth gradation, with the center at .5, it is a straight line. Moving the center to .75 pushes the center to the right, making the entire image darker. This is reflected in the *PlotScanline* node.



Example 2

In this example, some color-correction nodes are inserted and the values are adjusted. The *PlotScanline* indicates exactly what data gets clipped, crunched, compressed, or confused.



Parameters

This node displays the following controls in the Parameters tab:

width

The width of the PlotScanline. You likely want to set the width to 256 on an 8-bit image to get one-to-one correspondence.

height

The height of the PlotScanline. You likely want to set the height to 256 on an 8-bit image to get one-to-one correspondence.

line

The Y-line of the image to be analyzed. On a horizontal ramp, this does not matter, as they are all identical.

Histogram

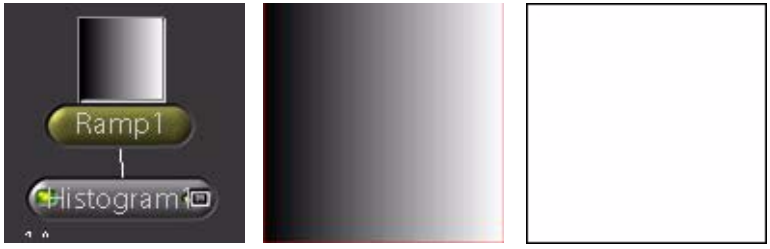
The *Histogram* node is an analysis tool that examines an image and graphs the occurrence per channel of each value. The X axis of the *Histogram* corresponds to the numerical value of a pixel. The Y axis is the percentage of pixels per channel with that value. The graph has nothing to do with the input pixel's original X or Y position in the image.

Note: The *Histogram* node should not be used inside of macros.

The following are some examples:

Example 1

A 256 x 256 8-bit black-and-white *Ramp*. Since there are equal amounts of the entire range of pixels, the result is a solid white field. The orientation of the ramp has no bearing on the graph.



Example 2

A 256 x 256 8-bit *Color*. Since the color is set to (approximately) .75, .5, .25, each channel exists at only one position in the *Histogram*.

Example 3

A 256 x 256 8-bit 4-corner *Grad*. The four corner values are of red (1, .5, .5, .5), of green (0, 1, .5, .5), and of blue (.5, 0, 0, 0). This reflects that most of red's values are around .5, ramping downward to 1, and no value is less than .5. In the Viewer, press R, G, B, or C to toggle through the different channels.

Example 4

Next, a *Brightness* of 2 is added between the *Grad* and the *Histogram*. The maxPerChannel parameter is enabled in the *Histogram* to better see the results. The result (the image is zoomed in here) is no odd values (all numbers are multiplied by 2), so there is a gap at every other value. It is a telltale sign of digital alteration when such regular patterns appear in a histogram.

Parameters

This node displays the following controls in the Parameters tab:

width

The width of the *Histogram*. You probably want to set the width to 256 on an 8-bit image for one-to-one correspondence.

height

The height of the *Histogram*. You probably want to set the height to 256 on an 8-bit image for one-to-one correspondence.

ignore

Tells Shake to ignore black or white pixels. For example, if you have a small element on a large black background, your histogram is skewed toward black. Disable black consideration to better analyze the image.

- *0* = No ignore.
- *1* = Ignore values of 0.
- *2* = Ignore values of 1.
- *3* = Ignore values of 0 and 1.

maxPerChannel

This determines how the graph channels relate to each other. If you have a large red component and a small blue component, the blue is very short, making it difficult to see. Toggle this to view the blue channel in relation to itself, rather than the red.

- *0* = Distribution of values relative to RGB total.
- *1* = Distribution of values relative to its own channel.

Keying

24

Shake provides powerful, industry-standard keying tools in the Primatte and Keylight nodes, along with additional keying nodes such as LumaKey and SpillSuppress. When combined with Shake's other filtering, masking, and color-correction nodes, you have detailed control over every aspect of the keying process.

About Keying and Spill Suppression

The first part of this chapter presents different strategies for pulling keys in Shake. Keying can be loosely defined as the creation of a new alpha channel based upon the pixel color (a pure blue pixel, for example), luminance (a very dark pixel), or Z depth (a pixel 300 Z units away, for example) in an image. Keying is discussed as a separate process from masking, which can be loosely defined as the creation of an alpha channel by hand through the use of painting, rotoshapes, or imported alpha masks from 2D or 3D renders in other software packages. For more information on these masking techniques, see Chapter 19, "Using Masks."

If you are not familiar with *Primatte* and *Keylight* (Shake's primary bundled keyers), you are encouraged to work through the keying lessons in the *Shake 4 Tutorials* book prior to reading through this chapter.

32-bit Support in Primatte and Keylight

As of Shake 4, the *Primatte* and *Keylight* nodes preserve 32-bit image data.

Pulling a Bluescreen or Greenscreen

In the Key Tool tab, the two primary nodes used to pull bluescreen and greenscreen keys are *Primatte* and *Keylight*. In the *Shake 4 Tutorials,* there are lessons devoted to each. Other functions in the Key tab include the *ChromaKey*, *DepthKey*, *DepthSlice*, *LumaKey*, and *SpillSuppress* nodes. These are discussed in the second half of this chapter. The *ColorReplace* node, although located in the Color tab, is also considered to be another key-pulling node.

Note: Although there is a *ChromaKey* node in the Key tab, it is not particularly useful. The same model and parameters appear in *ColorReplace*, but *ColorReplace* generally works much better.

Keying With Primatte and Keylight

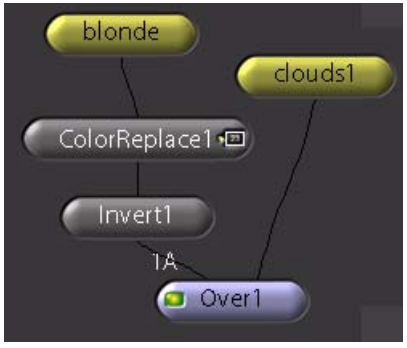
For lessons on how to use *Primatte* and *Keylight* to pull keys, see the *Shake 4 Tutorials.*

Keying With ColorReplace

ColorReplace is not the best keying tool, but it is a good node for making quick garbage masks. The following example uses *ColorReplace* to key the *blonde* image over the *clouds1* image.

To set up the key example:

1. Click the Image tab and select *FileIn*.
2. Go to the *$HOME/nreal/Tutorial_Media/Tutorial_06/images* directory and load the *alex_fg.jpg* and *clouds1.jpg* images. (The images are courtesy of Photron.)
3. Create the following tree:



4. Set the following parameters:
   - *ColorReplace node:* Click the SourceColor control, then scrub on the bluescreen in the *blonde* image. Set Replace Color to any other color (it just cannot be the same blue). Also, turn on affectAlpha so that you pull a key.
   - *Invert node:* Set the channels to "a" instead of rgba.
   - *Over node:* Enable preMultiply.

Because *ColorReplace* puts white in the SourceColor area of the alpha channel, use the *Invert* node to invert the image for the *Over* node.

The initial settings yield blue fringes. In the *ColorReplace* parameters, set satFalloff to 1 to correct this. Also, if pure black or pure white pixels start to show transparent, set the valRange and valFalloff numbers to approximately .2 and .5.

You can see there is some crunchiness, particularly in the hair. This demonstrates why *ColorReplace* is usually used to pull a key to mask other operations such as color correctors, rather than the actual composite. There are examples of using *ColorReplace* in the "Blue and Green Spill Suppression" section below.

Combining Keyers

You get the most flexibility when you combine keyers. Both *Primatte* and *Keylight* allow you to input a holdout matte. There are at least three typical ways of combining keys:

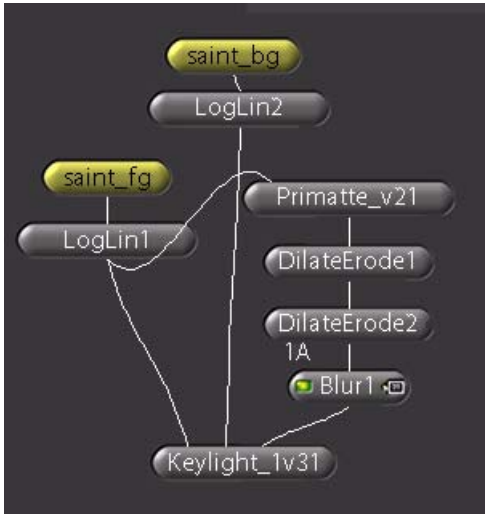
- Use the *Primatte* or *Keylight* holdout matte input.
- Use the *Primatte* arithmetic operator.
- Use the *Max*, *IAdd*, *IMult*, *Over*, *Inside*, or *Outside* node.

You can combine keys with the holdout matte input. Typically, you pull a basic key for soft edges and reflections. You also pull a second key which is very hard, and then soften it with filters.

In the following example, the first image shows the initial key coming out of the *Keylight* node. The reflections are good, but there is some transparency near the seat belt and steering wheel.

Next, a key is pulled with *Primatte* on the same bluescreen. With just foreground and background operators, the reflection is removed and the transparent areas filled in. This results in a very hard matte.

Next, filters are attached to the *Primatte* node. A *DilateErode* is added, and the xPixels parameter set to 1 (this closes up any holes in the alpha channel). You can also use a Median filter to do the same thing. A second *DilateErode* is applied, with the xPixels set to -5. This eats away at the matte. This filtering process cleans up the matte, making it more solid. A *Blur* softens it, and is fed into the HoldOutMatte input (the third input) of the *Keylight* node. The result is a solid matte with soft edges.

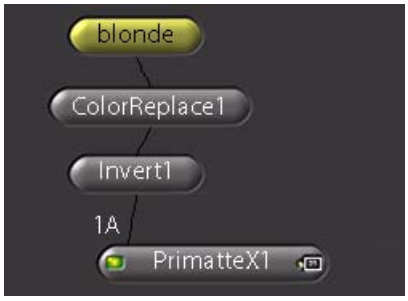


An Alternative for Making Hard Mattes

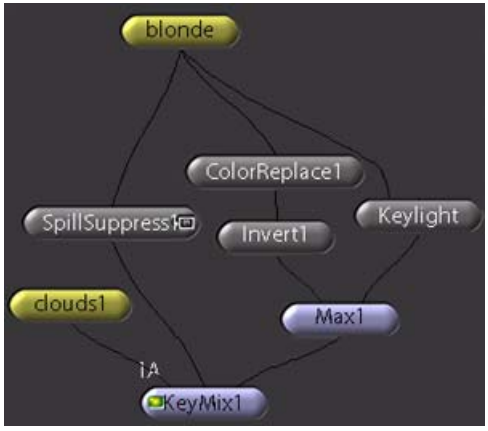
Copy your *Keylight* node and boost the screenRange to .3 to harden the matte.

A Matte Touch-Up Tool: The KeyChew macro, covered in Chapter 32, "The Cookbook," is also a good tool for duplicating the *DilateErode* chain in the above example.

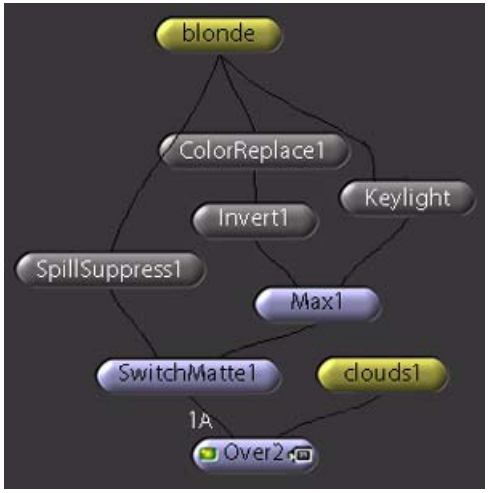
Another way to combine keys applies only to the *Primatte* node, which features a useful arithmetic parameter. Normally, when you pull a key in *Primatte*, the alpha mask is replaced in the foreground image. When the arithmetic parameter is switched from replace to add, multiply, or subtract, you can combine the mattes within *Primatte*. In the following node tree, an initial key is pulled with *ColorReplace,* with the affectAlpha button turned on. The resulting alpha channel is then inverted, without inverting the R, G, and B channels, using an *Invert* node. The inverted alpha channel is then combined with the *Primatte*-generated alpha channel by setting the *Primatte* arithmetic parameter to Add.



The final way to combine keys is to use layer nodes. In the following tree, keys are pulled using two different nodes and then combined with a *Max* node, which takes the greatest pixel value. The elements are combined with a *KeyMix* node. Note that the *KeyMix* does not get an alpha channel, since neither *clouds1* nor *blonde* has an alpha channel. *KeyMix* only mixes the two images through the mask you have pulled.



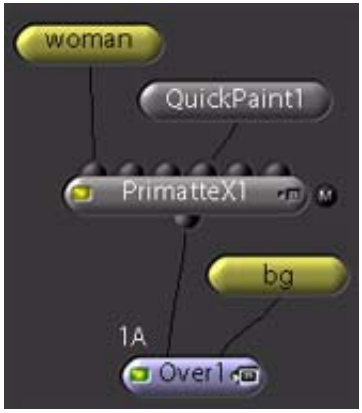
The following example uses a *SwitchMatte* node to assign the information from the combined keys to the foreground image. The resulting combined image data is then composited against the background using an *Over* node.



Blue and Green Spill Suppression

Once the composite is pulled from the keyers and put back into the *KeyMix* or *Over* node, you can start to work with spill suppression. Blue spill occurs when light is reflected off of the bluescreen and onto the foreground material.

The following examples use the *woman.iff* and *bg.jpg* files in the */Tutorial_08/images* directory. Notice that there is quite a bit of blue spill on the woman's shirt. In the following tree, the *Primatte* output is set to alpha only; a holdout matte is created for the line under her arm (with a *QuickPaint* node); and foreground and background scrubs are added. The composite is done with an *Over* node that has preMultiply enabled. Notice the blue spill that remains:

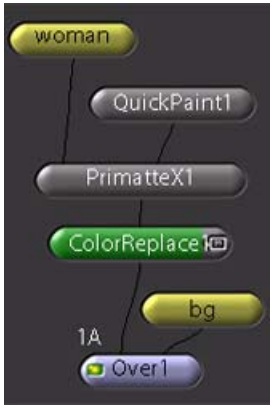


Although it's tempting to think that it would look better if you switch the *Primatte* output to comp and turn off the preMultiply in *Over*, this isn't the case. This has the unanticipated result of turning your blue edges into black edges that are actually more difficult to remove. Additional techniques to help correct this are discussed below.

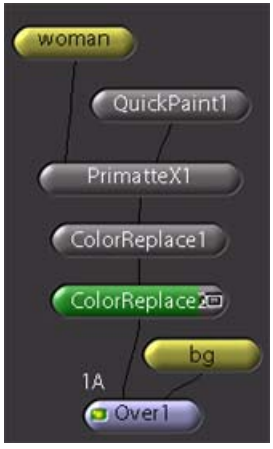
To avoid blue spill correction in your keyers, the following examples contain several sample trees to help you.

Using Color Replace—Method One

This technique is nice because it is fast, but often simply replaces blue spill with a different color. In the following tree, a *ColorReplace* node is applied to the foreground, and replaces the blue with the color of the wall. As always, increase the satFalloff to a value near1. The head looks great, but spill remains on the shirt, and there is now a yellow edge around the skirt.

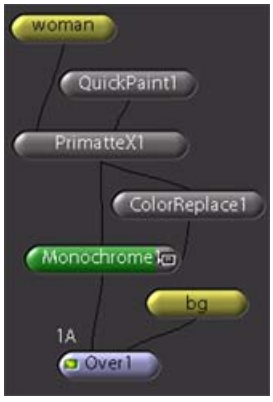


To correct this, drop the *ColorReplace* valRange to 0 and the valFalloff to approximately .4. Add a second *ColorReplace* node to eliminate the blue spill on the shirt. Carefully select the blue on the shirt, and then replace it with a beige-white color. Also, move all Range parameters to 0 and all Falloff parameters to approximately .3. The shirt looks good, but there is still a yellow ring around her skirt.



Using Color Replace—Method Two

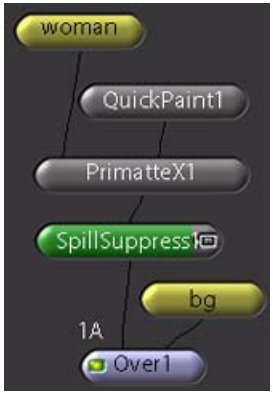
A better technique is to use *ColorReplace* to mask a color correction. Replace *ColorReplace2* with a *Monochrome* node, then pipe *Primatte* directly into it. Next, attach the output of *ColorReplace1* as the *Monochrome1* mask, and turn on the affectAlpha parameter in *ColorReplace*. This process turns off all saturation in the blue areas and turns those areas gray. This is good since the eye can detect luminance levels better than saturation levels. The skirt and the shirt now look good.



As an alternative to *Monochrome*, you can use the *AdjustHSV* or *Saturation* node.

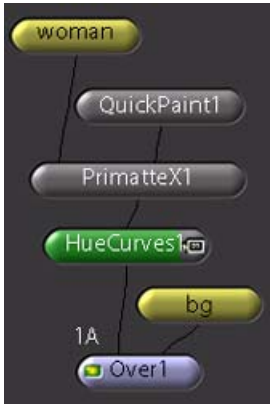
SpillSuppress

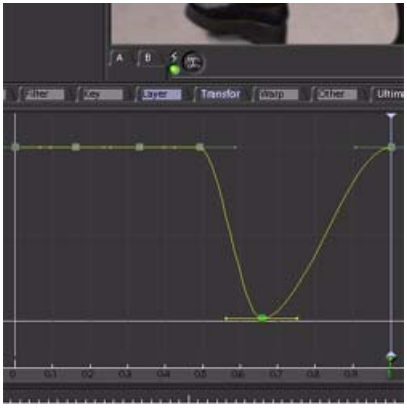
The *SpillSuppress* node mathematically evaluates each pixel and compares the blue and green strength. If the blue is significantly stronger than the green, the blue is suppressed. Because of the luminance difference between blue and green, the *SpillSuppress* node tends to work better on blue spill than green spill. It also tends to push your images to a yellow color.



HueCurves

The *HueCurves* node, located in the Color Tool tab, enables you to boost colors or saturation based on the hue of the pixel you want to affect. *HueCurves* works by loading a parameter into the Curve Editor and tuning it—ignoring the value fields in the node parameters. The X axis is the hue, and the Y axis depends on the parameter you are using. For the following example, load the saturation curve into the Curve Editor and grab the control point around .66, since blue has a hue of 66 percent. Drag the point down to decrease the saturation for the blue pixels.





Edge Treatment

Another typical problem when keying (OK, it is *the* problem with keying) is edge treatment.

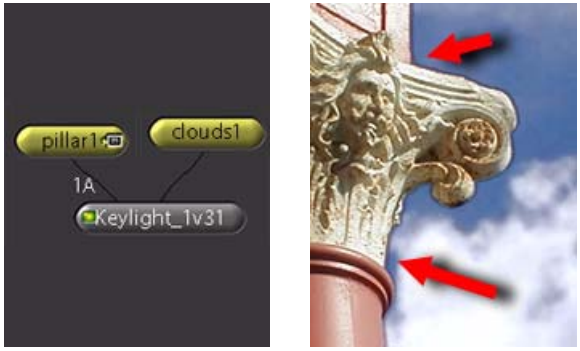
The following example uses two images from the *$HOME/nreal/Tutorial_Media/Tutorial_06/images* directory:

1. In the Image tab, click *FileIn*.
2. Go to the */Tutorial_06/images* directory and read in the *pillar1.jpg* and *clouds1.jpg* images.

   These images bring up an important tip for compositors: Supervisors are typically optimistic about the results possible from using the sky as a bluescreen.
3. In the Node View, select the *pillar1* node.
4. Click the Key tab, then click the *Keylight* node.

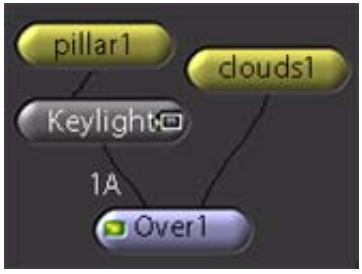
   The *pillar1* node is connected to the *Keylight* node's foreground input.
5. Connect the *clouds1* node to the *Keylight* node's background input.
6. In the *Keylight* node parameters, click the screenColour control, then scrub the sky near the horizon.

A black line appears around the pillar.



As mentioned earlier, it is better to composite after pulling your key because it gives you more flexibility.

7 So, rewire the tree with an *Over* node. Make sure you turn on preMultiply in the *Over* node parameters and set the *Keylight* output to unPremult.



8 Next, attach a *DilateErode* node to the *Keylight* node.

9 In the *DilateErode* channels parameter, enter "a" (replace rgba) to affect only the alpha, then set xPixels to -1. Note that the *DilateErode* node chews into the matte.

The next image illustrates a disabled preMultiply parameter in the *Over* node (because the mask/RGB premultiplied relationship is upset). White lines appear around the edges.

This introduces another problem: The electrical power lines have disappeared from the lower-right corner of the image. Because the details are so fine, the *DilateErode* node has chewed the lines away.

The easiest way of correcting this in this example is to use a manually painted mask, which limits the effect of the *DilateErode* node.

To restore fine detail to the composite:

1. Attach a *QuickPaint* node to the Mask input of the *DilateErode* node.

   A mask is attached to the *DilateErode*.

2. Paint across the electrical wires.

   But this only dilates areas where the wires exist—the opposite of what you want.

3 Open the Mask subtree in the *DilateErode* parameters, then enable invertMask.

The edges are now dilated everywhere except around the wires area.

For more information on the *QuickPaint* node, see "About the QuickPaint Node" on page 579.

Applying Effects to Bluescreen Footage

Problems can occur when you apply effects to keyed footage. For example, suppose you want to blur the foreground image, but not the background. Your first instinct would probably be to apply a *Blur* node to the foreground image, and then key and composite with *Keylight*. *This is inadvisable.* This section presents ways to find and avoid problems with this workflow.

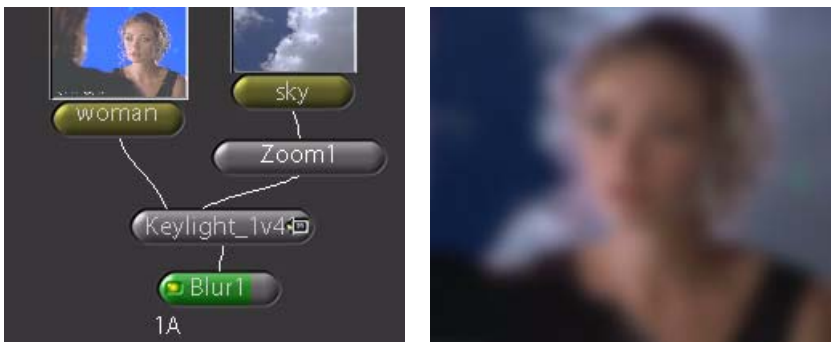
The following node tree uses the *woman.iff* and *sky.jpg* images in the *Tutorial_Media* directory. Special thanks to Skippingstone for the use of images from their short film *Doppelganger*.

Problem 1: Edge Ringing

When the blur is applied, a blue edge is introduced along the woman's neck line.

Problem 2: Accidentally Blurring Both Layers

One might be tempted to place the *Blur* node after the *Keylight* node in the tree. This also produces an incorrect result—the background is blurred as well.



Problem 3: Artifacts Introduced by Masking

Lastly, one might be tempted to mask the blur using the key from *Keylight*. This also produces an incorrect result.

Filtering Keys: The Correct Way

The problem with the three examples above is that the keying node, in this case *Keylight*, is being made to do too much—by having to simultaneously key, suppress spill, and composite within the same node, there is no good place to apply a filter.

The solution is to pull a key for purposes of creating a mask, but to use other nodes to perform the actual compositing. In the following example, the *Keylight* output parameter can be set to either "comp" or "on Black." The image and alpha that's output from the *Keylight* node can then be filtered, and the filtered result composited against the background using a simple *Over* node.

In general, it's good practice to use the *Primatte* and *Keylight* background inputs as test composites only, to be able to see how the key is looking while you're tuning it. Once you're done pulling the key, you can rewire the output image into a composite using an *Over* or a *KeyMix* node. This method of working gives you several advantages:

- You can apply filters and effects to the foreground material.
- You can transform the foreground material.
- You can color correct the foreground material.

Keying DV Video

DV footage, which is compressed with a 5:1 ratio as it's recorded with 4:1:1 (NTSC) or 4:2:0 (PAL) color sampling, is less than ideal as a format for doing any kind of keying. This is due to compression artifacts that, while invisible during ordinary playback, become apparent around the edges of your foreground subject when you start to key.

With a high-quality DV camera and good lighting, it's possible to pull a reasonable key using DV clips, but you cannot expect the kind of subtleties around the edges of a keyed subject that you can get with uncompressed or minimally compressed video (decent) or film (best). For example, while you may be able to preserve smoke, reflections, or wisps of hair when keying uncompressed footage, with equivalent DV footage this probably won't be possible.

On the other hand, if your foreground subject has slicked back hair, a crisp suit, and there are no translucent areas to worry about, you may be able to pull a perfectly acceptable key.

The following example presents an impeccably shot bluescreen image, recorded using a high-quality DV camera.

When you key the image and place it over a background (a red field is used in this example), the result is marred by blocky, aliased-looking edges all around your foreground subject. Unfortunately, that's the 4:1:1 color sampling of DV making itself seen.

DV bluescreen

Key pulled on DV bluescreen

Why is this happening? In the RGB color space, each pixel in your picture is represented with a value of Red, Green, and Blue.

When you shoot video—which uses the YUV (or YCrCb) color space, each pixel is represented by a luminance value (Y) and two chrominance values: red/green difference (Cr) and blue/green difference (Cb). This is done because green has a higher luminance value, and is therefore more likely to display artifacts if too much information is taken away.

In the video world, 4:4:4 color sampling means that for every four pixels in a row, you get four pixels each for Y, Cr, Cb. This is equivalent to 8-bit RGB. 4:2:2 means that you get four Y pixels and two each of Cr/Cb for every four pixels in a row. You get less information in a smaller package, giving you a little more speed and a usually acceptable loss of detail.

The DV format's 4:1:1 color sampling means that you get four Y (luminance) pixels and a single each of Cr/Cb (red/green and blue/green difference). In other words, luminance can change at every pixel, but color changes only once every four pixels.

If you look at the keyed DV footage, you see that each of those blocks around the edge of your subject is four pixels wide. A lot of data is saved. Unfortunately, bluescreen keyers pull keys from color, not luminance; hence the artifacts.

Although the information in video is transferred from the YUV colorspace into the RGB colorspace, you can still examine the original YUV channels. Attach a Color–*ColorSpace* node to the *FileIn* and set the outSpace to be YUV. With the pointer over the Viewer, press the R, G, and B keys to look at the new YUV channels.

Y (luminance)

U (Cr, or r/g difference)

V (Cb, or b/g difference)

As you can now see, the luminance channel gets all of the necessary information, since the human eye is more susceptible to differences in luminance than hue. But the two color channels display significant artifacts.

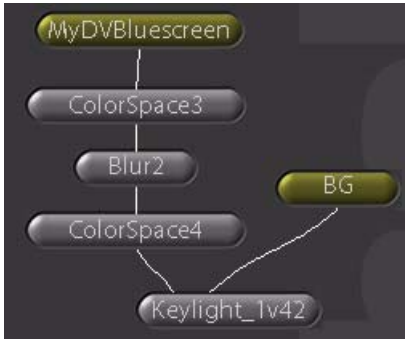
Here are two methods you can use to help you pull better keys from DV footage:

To correct a DV key (method 1):

1. Attach a *ColorSpace* node to the *FileIn* node containing the DV footage.
2. Set the outSpace parameter of the *ColorSpace* node to YUV.

   The colors in the image radically change, but don't worry, this is only temporary.
3. Attach a *Blur* node to the output of the *ColorSpace* node.
4. In the *Blur* node's channels parameter, blur only the U and V channels by switching the channels from rgba to gb.
5. Blur the isolated gb channels by adjusting the xPixels parameter by approximately 8, and adjusting the yPixels parameter independently by a much smaller value, approximately 1 pixel.
6. Attach a second *ColorSpace* node to the output of the *Blur* node, then set the inSpace parameter to YUV and keep the outSpace parameter set to RGB.

This converts the image back to RGB space.



The key is greatly improved. In particular, the original blockiness around the edge is gone. Unfortunately, you're still missing any fine detail that you might otherwise have had were the footage shot using a different format.

Without blur on UV channels

With blur on UV channels

Note: Using this method when you use straight YUV files, you can bypass the RGB to YUV conversion by turning off yuvDecode in the *FileIn* node. Apply the *Blur*, and then use one *ColorSpace* node to convert the image to RGB space.

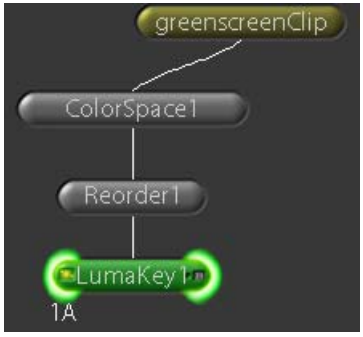
The next method will use a different image, one with more hair detail that takes a little more effort to preserve.

To correct a DV key (method two):

1. As in the first method, attach a *ColorSpace* node to the *FileIn* node containing the DV footage, then set the outSpace parameter to YUV.
2. Next, attach a *Reorder* node to the output of the *ColorSpace* node. Set the channels parameter to rrra to reassign the Y channel (the least compressed channel) to all three channels.

   The result will be a sharp grayscale image.
3. Attach a *LumaKey* node to the output of the *Reorder* node, and pull a key, concentrating on the edges of the subject.

   Don't worry if there are holes in the middle of the image. You're just trying to isolate as much of the edge of the subject as possible.



4. Now, go back up to the *FileIn* node at the top of the tree, and branch off another keying node, such as *Keylight*.
5. Pull a key, this time concentrating on the interior of the subjects.

   In essence, this second key is a holdout matte that you can combine with the *LumaKey* you created in step 3.

6 Now, attach the outputs of the *LumaKey* and the *Keylight* nodes to a *Max* node, to combine both alpha channels into one.

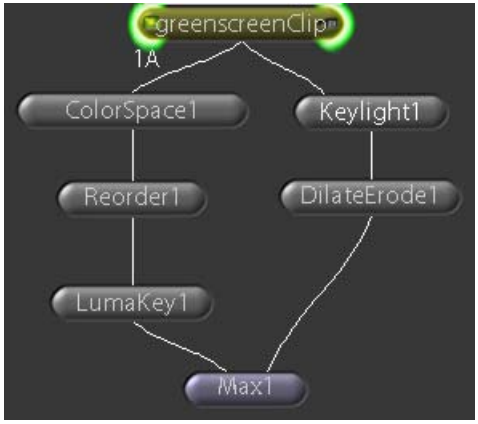
The LumaKey matte

The *Keylight* matte

The above screenshots show the results of each individual key, combined into the single alpha channel below. As you see, the holes in the *LumaKey* matte are filled by the *Keylight* matte, and the loss of detail around the edge of the *Keylight* matte is restored by that in the *LumaKey* matte. These keys can also be combined in many other ways using different layering nodes and rotoshape masks to better control the combination.

The combined LumaKey and Keylight matte

7 As an optional step, you may find it necessary to insert a *DilateErode* node between the *Keylight* and *Max* nodes in order to erode the output matte from the *Keylight* so it doesn't interfere with the edge created by the *LumaKey*.



This produces a mask that you can then recombine with the original foreground image and a background image using a *KeyMix* node. You'll probably have to deal with some spill suppression, but you can use the same techniques described in "Blue and Green Spill Suppression" on page 687.

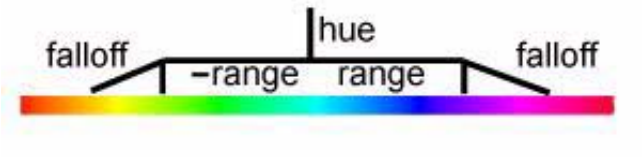
Note: If you're really detail-oriented, you can combine both of the above methods, in an effort to preserve more detail from the greenscreen key performed in method two.

Keying Functions

The following section details the keying nodes located in the Key Tool tab. The *ColorReplace* node, discussed above, is located in the Color Tool tab. For more information, see Chapter 23, "Color Correction."

ChromaKey

The *ChromaKey* node examines the HSV values of an image and pulls a matte based upon the parameters. In the interface, you can scrub a color in the Viewer. However, disable the matteMult parameter before you scrub. The hue, saturation, and value of an image each has a set of parameters to describe the exact HSV values you are keying, as a range from that midpoint, a falloff value, and a sharpness value to describe the falloff curve. This is illustrated in the following image:



The *ChromaKey* node is not Shake's strongest feature. It is recommended to use Color–*ColorReplace*; select your bluescreen color; choose your destination color (any other color); then enable affectAlpha.

A typical application is bluescreen or greenscreen removal. You can stack multiple *ChromaKey* nodes to extract different ranges. With the arithmetic parameter, you can choose to add to, subtract from, or replace the current mask.

Parameters

This node displays the following controls in the Parameters tab:

HSVColor
Picks the center value to be pulled on hue, saturation, and value.

hueRange
Plus and minus from the hue specified by the HSVColor parameter.

hueFalloff
Describes the falloff range from hueRange that is picked, with the values ramping down.

hueSharpness
Describes the falloff curve from hueRange to hueFalloff.

- 0 = linear drop-off
- 1 = smooth drop-off

satRange
Plus and minus from the saturation specified by the HSVColor parameter.

satFalloff
Describes the falloff range from satRange that is picked, with the values ramping down.

satSharpness
Describes the falloff curve from satRange to satFalloff.
- 0 = linear drop-off
- 1 = smooth drop-off

valRange
Plus and minus from the value specified by the HSVColor parameter.

valFalloff
Describes the falloff range from valRange that is picked, with the values ramping down.

valSharpness
Describes the falloff curve from valRange to valFalloff.
- 0 = linear drop-off
- 1 = smooth drop-off

matteMult
Toggle to premultiply the RGB channels by the pulled mask.
- 0 = no premultiply
- 1 = premultiply

arithmetic
This parameter lets you define how the mask is created by the key.
- 0 = replace existing mask
- 1 = add to existing mask
- 2 = subtract from existing mask

DepthKey

The *DepthKey* node creates a key in the alpha channel based on depth (Z) values. Values below loVal are set to 0, and values above hiVal are set to 1. Values in between are ramped. You also have roll-off control, plus a matteMult toggle. If there is no Z channel in your image, this node does not work.

If the Z channel is not directly imported with the *FileIn* node, you can copy it over with the Copy command, using z as your channel.

Note: When you import from Maya, there is a strange -1/distance Z setting. This basically means that the numbers in *DepthKey* are impractical. To correct this, go to www.highend2d.com and download the *MayaDepthKey* macro. The macro operates exactly like the normal *DepthKey*, but does the conversion math for you.

Parameters

This node displays the following controls in the Parameters tab:

loVal
Any pixel below this value (as calculated per its depth) turns black.

hiVal
Any pixel above this value (as calculated by its depth) turns white.

loSmooth
A roll-off factor to provide a smooth drop-off.

hiSmooth
A roll-off factor to provide a smooth drop-off.

matteMult
Toggle to premultiply the RGB channels by the pulled mask.
- 0 = no premultiply
- 1 = premultiply

DepthSlice

Similar to *DepthKey*, the *DepthSlice* node creates a slice in the alpha channel based on Z, as defined by a center point, and a drop-off range.

Parameters

This node displays the following controls in the Parameters tab:

center
The center Z depth from which the slice is measured.

lo
The distance included in the slice away from the center. lo adds distance toward the camera.

hi
The distance included in the slice away from the center. hi adds thickness away from the camera.

grad
When enabled (1), there is a gradation from hi to lo. Beyond, the slice is still black.

mirror
When enabled, the effect is mirrored in Z.

matteMult
Toggle to premultiply the RGB channels by the pulled mask.
- 0 = no premultiply
- 1 = premultiply

Keylight

Keylight is an Academy Award-winning keyer from Framestore CFC based in England. It accurately models the interaction of the bluescreen or greenscreen light with the foreground elements, and replaces it with light from the new background. With this approach, blue spill and green spill removal becomes an intrinsic part of the process, and provides a much more natural look with less tedious trial-and-error work. Soft edges, such as hair, and out-of-focus edges are pulled quite easily with the *Keylight* node.

To ensure the best results, try to always pull the key on raw plates. In the *Keylight* parameters, there is a colourspace control to indicate if the plate is in log, linear, or video color space. Therefore, you should not perform color correction on film plates when feeding the plates into *Keylight*.

For a hands-on example of using the *Keylight* node, see Tutorial 5, "Using Keylight," in the *Shake 4 Tutorials*.

Float Support in Keylight

The *Keylight* node now supports the preservation of 32-bit data. As a result, float images may require different keyer settings than 8-bit images. For example, highlights in the foreground subject of float images may produce unexpected areas of translucency due to barely perceptible green or blue casts that are preserved in float, but which would be clipped in 8 bit or 16 bit. This can be addressed by lowering the highlightGain parameter (to approximately -0.25) to strengthen weak areas in the interior of the key.

Parameters

This node displays the following controls in the Parameters tab:

output

You have the option to do your composite within the *Keylight* node, but there are other output options as well should you want to composite the foreground image using other nodes:

- *comp*: Renders the final composite, against the assigned background.
- *on Black*: Renders the foreground objects over black, creating a premultiplied output.
- *on Replace*: Renders the foreground objects over the replaceColour. This is a good mode to test your composite. Choosing a bright color allows you to instantly see if there are unwanted transparent areas in the foreground subject.
- *unpremult*: Renders the foreground without premultiplying the matte. Use this mode when you want to add transformations and color corrections after pulling the key. You then apply either a *MMult* node or enable preMultiply in an *Over* node to create the final composite.

- *status*: Displays an image with different colors, each of which indicates what portions of the foreground image are handled in which way by *Keylight*. This mode is useful for helping you to troubleshoot your key.
  - *Black pixels:* Areas that become pure background in the composite.
  - *Blue pixels:* Areas that become spill-corrected foreground.
  - *Green pixels:* A blend of foreground and background pixels.
  - *Pure green:* Mostly foreground and dark green is mostly background.

screenColour
The primary color to be pulled, which is usually blue or green.

Note: *Keylight* is tuned to the primary colors and is not effective on secondary colors (cyan, magenta, yellow). If trying to pull these colors, consider switching your image from RGB to CMY with the *ColorSpace* node, pull the key, and then switch back to RGB.

screenRange
Defines the range of colors that should be keyed out. The higher the number, the more of the background screen is removed. A value of 0 gives the smoothest key that retains the most fine detail; a value of .3 removes all the gray levels, and may result in coarser edges around the foreground subject.

fgBias
Foreground bias is used to reduce the blue spill on foreground objects. *Keylight* uses this color to calculate which shades the screen color passes through as it interacts with the foreground elements.

For example, blonde hair in front of a bluescreen tends to go through a magenta stage. Setting the FG Bias to the blonde color ensures the magenta cast is properly neutralized. This value affects both opacity and spill suppression. Return it to .5, .5, .5 to effectively deactivate this effect.

Avoid picking strong colors for the FG Bias. Muted shades work much better. Another way of looking at this parameter is as a way of preserving a foreground color that might otherwise be neutralized because it's too close to the key color. For example, a pale green object, such as a plant, in front of a greenscreen would normally become slightly transparent, with the background showing through instead of the pale green. By setting the FG Bias to the pale green, it is preserved in the composite.

Note: Please don't shoot plants in front of a greenscreen.

fineControl
The parameters in the fineControl subtree are used to make detailed adjustments to the matte that is created by the *Keylight* node.

- *shadowBalance, midtoneBalance, and highlightBalance:* Located within the fineControl subtree, these parameters help you when the screen area is slightly off from a completely pure primary color—for example, cyan instead of pure blue.

The transparency of the foreground is measured by calculating the difference between the dominant screen color (blue by default, otherwise the value of the screenColour parameter) and a weighted average of the other two colors (red and green).

With the example of a cyan screen, there is a greater difference between the blue and the red than between the blue and the green, since cyan has more green than red. Setting the balance to 0 forces *Keylight* to ignore the second-most dominant color in the screen, which is green in the example. When set to 1, the weakest screen color (red) is ignored. There are three controls to tune the low, medium, and highlight ranges.

- *shadowGain, midtoneGain, and highlightGain:* Located in the fineControl subtree, these parameters let you increase the gain to make the main matte more transparent. This tends to tint the edges the opposite of the screen color—for bluescreens edges become yellow. Decrease the gain to make the main matte more opaque.

  Note: You can lower the highlightGain parameter (to approximately -0.25) to strengthen weak areas in the interior of a mask that are due to green or blue casts in the highlights of foreground subjects in float images.

- *midTonesAt:* Located in the fineControl subtree, this parameter adjusts the effect of the balance and gain parameters by changing the level of the midtones they use. For example, If you are working on a dark shot, you may want to set the midtone level to a dark gray to make the controls differentiate between tones that would otherwise be considered shadows.

replaceColour
Spill can be replaced by the replaceColour. This occurs only in the opaque areas of the holdout matte. This is useful with blue areas in the foreground that you want to keep blue and opaque. The replaceColour is therefore blue.

fgMult
Allows color correction on the foreground element. This exactly mimics the Shake *Brightness* node.

fgGamma
Applies a gamma correction to the foreground element. This exactly mimics the Shake *Gamma* node.

saturation
Applies a saturation correction to the foreground element. This exactly mimics the Shake *Saturation* node.

colourspace

Keylight models the interaction of the blue/green light from the screen with the foreground elements. For these calculations to work correctly, you need to specify how pixel values relate to light levels. This is the function of the colourspace menu. Therefore, with Cineon plates (or other logarithmic files) you have the option to pull the key with or without a Delog operator before the key pull.

- *log:* Color spaces are designed so that a constant difference in pixel values represents a fixed brightness difference. For example, in the Cineon 10-bit file format, a difference of 90 between two pixels corresponds to one pixel being twice as bright as the other.
- *linear:* This color space has the brightness of a pixel proportional to its value. A pixel at 128 is twice as bright as a pixel at 64, for example.
- *video:* Color space has a more complicated relationship, but the brightness is approximately proportional to the pixel value raised to the power of 2.2.

plumbing

The subparameters in the plumbing subtree allow you to adjust how the input images are used to create the final output image and matte.

- *useHoldOutMatte:* If the third image input is used for a holdout matte, toggles the holdout matte on and off.
- *holdOutChannel:* Specifies which channel of the image is being used as the holdout matte.
- *useGarbageMatte:* If the fourth image input is used for a garbage matte, this button toggles the garbage matte on and off.
- *garbageChannel:* Specifies which channel of the image is being used as the garbage matte.
- *bgColor:* Either pulls a key on the area outside of the frame (0), or asserts the background as the background color (for example, usually black).
- *clipMode:* Sets the output resolution of the node—either the foreground image (1) or the background image (0) resolution.

LumaKey

The *LumaKey* node creates a key in the alpha channels based on overall luminance. Values below loVal are set to zero, and values above hiVal are set to 1. Values in between are ramped. You also have roll-off control, and an mMult toggle.

This is a fast way to place the luminance of an image into the alpha layer, but this operation can also be done with a *Reorder* node set to rgbl.

Parameters

This node displays the following controls in the Parameters tab:

loVal

Any pixel below this value (as calculated by its luminance) turns black.

hiVal
Any pixel above this value (as calculated by its luminance) turns white.

loSmooth
A roll-off factor to provide a smooth drop-off.

hiSmooth
A roll-off factor to provide a smooth drop-off.

matteMult
Toggle to premultiply the RGB channels by the pulled mask.
- 0 = no premultiply
- 1 = premultiply

Primatte (Plug-in)

The *Primatte* plug-in is the latest update of Photron's Primatte keying software. The Shake *Primatte* node allows you to scrub across an image to determine matte areas in order to pull a key (or alpha channel) for a composite. The plug-in also works on the RGB channels to suppress spill (the leaking of blue or green color onto the foreground objects).

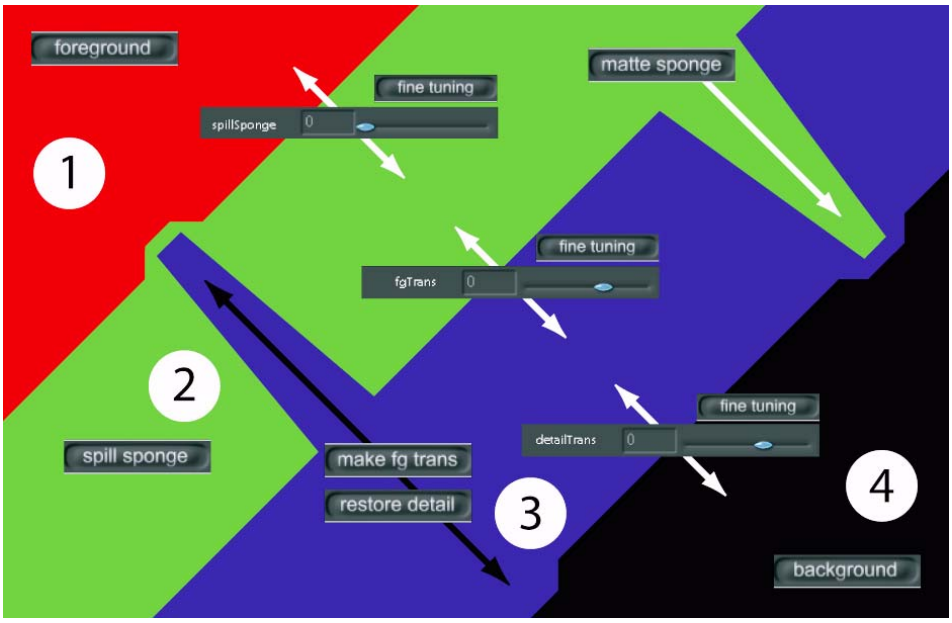
Note that Shake scripts pass special data to the *Primatte* plug-in. This data is encoded, which means that *Primatte* must be set up using the graphical interface.

To learn how to maximize the effectiveness of this node by combining it with other functions, see Tutorial 6, "Using Primatte," in the *Shake 4 Tutorials,* as well as this manual's section on "Blue and Green Spill Suppression" on page 687.

Supplying the Background Image

Although you can output *Primatte* with a premultiplied foreground with no background image, you should supply a background input if possible, as *Primatte* still factors in some of this information. If you do not supply this image, black ringing appears around the edges. If this is impractical, toggle the replaceMode parameter to use color and supply an appropriate Replace Color.

In *Primatte*, you assign color to one of four zones by clicking one of the eight large operator buttons, then scrubbing for a color in the Viewer. These four zones are arranged around a center point in 3D color space, with each zone situated like a layer of an onion. The following diagram shows the operator/zone assignments. Note that the "decolor all" button scales the entire 3D space, and shifts all color either toward or away from the foreground. Therefore, it does not involve picking a color, only moving a slider.



Parameters

This node displays the following controls in the Parameters tab:

clipMode

Sets the resolution to that of either the foreground (0) or the background (1).

output

You are not obliged to use *Primatte* for your composite, especially when you need to make transformations to the foreground object after the matte is pulled. Pull the matte on the full-resolution image prior to any scaling. The output setting determines what is changed by *Primatte*:

- *alpha only*: Only the matte is affected.
- *on black*: The foreground image and the matte are changed.
- *comp*: If there is a second input image, this composites that background in.

- *status*: Presents an image with different colors, displaying which parts of the image fall into the four *Primatte* zones. This mode is useful to help you troubleshoot your key.
  - *Black–Zone 1*: All background
  - *Blue–Zone 2*: Transparent foreground
  - *Green–Zone 3*: Suppressed foreground
  - *Red–Zone 4*: All foreground

arithmetic
Determines how *Primatte* affects the foreground matte channel.
- *Replace (0)*: Replaces the fg matte channel completely.
- *Subtract (1):* Subtracts the *Primatte*-derived matte from the incoming matte.
- *Multiply (2)*: Multiplies the two mattes together.
- *Add (3)*: Adds the two mattes together.

processBGColor
This button tells Shake whether or not to consider the pixels outside of the frame. When disabled (default mode), the area outside of the image is assumed to be 100 percent transparent. When enabled, the area outside of the frame is treated the same way black pixels are treated by your *Primatte* scrubs.

gMatteChannel
The channels to be used for the garbage matte. If no garbage matte is assigned, these parameters have no effect.

hMatteChannel
The channels to be used for the holdout matte. If no holdout matte is assigned, these parameters have no effect.

replaceMode
When you perform spill suppression through the use of the spillsponge operator or the fine tuning operator, these suppressed areas are shifted from blue (or whatever your center value is) toward a different color. There are two modes:
- *use image:* The default mode. This mode uses colors from either the background image, or the image connected to the *replaceImage* input knot, if one has been assigned.
- *use color:* Lets you pick a specific color using the ReplaceColor parameter.

ReplaceColor
The color used in the spill suppressed area if the replaceMode parameter is set to "use color."

operator

Each button that appears in the group of controls labelled "operator" allows you to modify the key created by *Primatte*, using a color you select with the Color control. The effect of all the operators you click is cumulative, and each operation you perform is saved in a history of operations that's accessible via the currentOp slider.

Every time you click an operator, an additional operation is added to the history of operations represented by the currentOp slider. To modify the currently selected operation, instead of adding a new one, select an operation using the currentOp slider, click the color control bearing that operation's name, then scrub the image to select a new color range.

The eight operator buttons include the following:

- *background:* Assigns pixels to the background. Areas of the foreground that you select with the background operator become100-percent transparent, with no spill suppression.
- *restore detail:* Removes transparency on background material. It is useful for restoring lost details such as hair. It is the equivalent of detailTrans in the "fine tuning" subtree, but with no slider.
- *fine tuning:* When the fine tuning mode is clicked, three additional parameters appear at the bottom of the Parameters tab.
  - *spillSponge*: When this mode is selected, the pointer motion in the Decolor slider performs a color adjustment of the sampled color against the background. After sampling a color region from the image, the more to the right the pointer moves, the less of the background color component (or spill) is included in that color region. The more to the left the pointer moves, the closer the color component of the selected region is to the spill suppress color.
  - *fgTrans*: Adjusts the transparency of the matte against the sampled color. After sampling a color region from the image, the more to the right the pointer moves, the more transparent the matte becomes in that color region. This is equivalent to the "make fg trans" button, but offers more control.
- *detailTrans*: Determines the transparency of the sampled color when it is close to the background color. The slider in this mode is useful for restoring the color of pixels that are faded due to similarity to the background color. If you slide to the left, picked areas are more opaque. This is equivalent to the "restore detail" button, but offers more control.
- *foreground:* When this mode is selected, the sampled pixels within the image window become 100 percent foreground. The color of the sampled pixels is the same color as in the original foreground image. The matte is completely white.
- *make fg trans:* Allows you to select regions within foreground elements in order to make them more transparent. This operation is used to adjust elements like clouds or smoke. It is the equivalent of the fgTrans conrol in fine tuning mode, except that it features no slider control.

- *decolor all:* When this mode is selected, the value parameter appears at the bottom of the Parameters tab.

  Adjusting the value parameter shrinks or expands the polyhedron between zone 4 (all foreground) and zone 3 (foreground plus spill suppress). Positive values expand the shell, effectively shifting across the entire image more color from the foreground into the suppressed area. Negative values contract the shell, thereby slipping more values away from the suppressed area into the foreground area. Because the shells cannot intersect, if you shrink the shell too much, you also crush the smaller interior shells, causing all values to shift toward the background.
- *spill sponge:* Affects the color of the foreground, but not the matte. This operation suppresses the color you pick. "spill sponge" is usually used on spill areas that are known to be opaque—for example, blue spill on the face or body. If the change is too drastic, supplement or replace the operation with fine tuning–spillSponge adjustment.
- *matte sponge:* Used to restore foreground areas lost during spill suppression. "matte sponge" only affects the alpha channel.

currentOp
Each operator you use to perform a scrub operation is maintained separately in memory. The history of all the operations you've performed within the *Primatte* node is accessible using the currentOp slider. Moving the slider to the left returns the you to any previous scrub operation you've performed, allowing you to re-adjust or delete it. To see which operation you've selected, look at the name that appears at the top of the color control that appears below the slider.

delete op
Click "delete op" to delete the operation that's currently selected in the currentOp parameter.

Color Control (initially set to "center")
This control isn't dedicated to any one parameter. Instead, the color control lets you assign a color to whichever operator you've clicked in the *Primatte* node. Additionally, this control displays the color that's currently assigned to whichever operator you've selected using the currentOp slider.

When you first assign a *Primatte* node to an image, this control is set to "center." It is the first operation you use when you begin keying with the *Primatte* node. Scrub pixels in the background of the image to be keyed to define the starting range of color to be keyed out. Doing so determines the center of the 3D polyhedron (described above), and is therefore extremely important. When the center operation is selected, the multiplier parameter appears at the bottom of the Parameters tab.

This initial pixel scrub that defines the *center* is always operation 0 in the currentOp slider. To readjust the center, move the currentOp slider all the way to the left, to operation 0.

Note: Readjusting the center operation will change the effect of all subsequent operations you have already performed.

As you click additional operators (for example, the "background," "foreground," and "spill sponge" operations), this color control lets you choose a color from the foreground image to assign to the currently selected operator.

multiplier
When the color control is set to center, the multiplier slider appears. This parameter lets you modify the size of the center area in 3D space. Therefore, a higher multiplier value expands the size of the background space, sucking in more of that color. The result is more transparent. A value lower than 1 reduces the amount of color sucked out by the initial background color pick.

evalToEnd
As you adjust the currentOp slider, you may want to see the composite only up to that point. Turn off Eval to End to view the key using only the operators at or before the current operation selected by the currentOp slider.

active
Instead of clicking delete op to eliminate an operator from the currentOp slider, you can disable any operation by turning off the active control. This disables the operation that's currently selected by the currentOp slider.

SpillSuppress

The *SpillSuppress* node suppresses blue or green spill, with controls for gain in the other color channels. *SpillSuppress* uses a color-correction algorithm, so the entire image is modified.

Parameters

This node displays the following controls in the Parameters tab:

rGain
Since the spill suppress darkens the overall image, you can use this control to slightly compensate for the brightness in the other two channels.

gGain
Since the spill suppress darkens the overall image, you can use this control to slightly compensate for the brightness in the other two channels.

lumGain
An overall luminance gain.

screenColor

Select color to suppress.

- *0* = Suppress blue
- *1* = Suppress green. gGain then converts to bGain.

Image Tracking, Stabilization, and SmoothCam

25

Shake provides several methods of tracking, stabilizing, and smoothing moving subjects in your scripts. Additional tools are provided to process the keyframed results of these operations, giving you even more detailed control.

About Image Tracking Nodes

Shake provides four image tracking and stabilization nodes: *Tracker*, *Stabilize*, *MatchMove,* and *SmoothCam*.

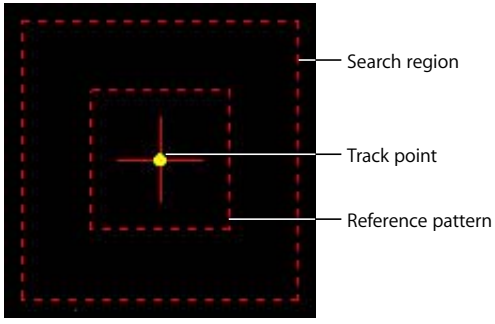
- *Stabilize*: The *Stabilize* node is used to correct unwanted movement in a shot. You can also use the *Stabilize* node for matchmoving using the inverseTransform parameter. Use of the *Stabilize* node's inverseTransform parameter is recommended rather than use of the *MatchMove* node, since it gives you the option to composite later with an *Over* node. This technique also gives you proper pass-through of onscreen controls for more intuitive control.
- *Tracker*: The *Tracker* node does not transform the input image. It is used only to generate tracks that can then be referenced by the *MatchMove* and *Stabilize* nodes, or nodes such as *Move2D* and *Pan* with standard linking techniques.
- *MatchMove*: The *MatchMove* node allows two input images—a Foreground (the first node input) and a Background (the second node input). By tracking one, two, or four points on the background, the foreground can be set to match the movement of the background. Placing a logo on the side of a moving truck is the classic example of a matchmove. The single advantage the *MatchMove* node has over the *Stabilize* node is that you can immediately test your track results.

  Note: You can also attach a tracker to a rotoshape or paint stroke. For more information, see "Attaching a Tracker to a Paint Stroke" on page 586 and "Attaching Trackers to Shapes and Points" on page 562.

- *SmoothCam:* This node differs from the others above in that it doesn't track small groups of pixels. Instead, it evaluates the entire frame, using motion analysis to derive the movement of the camera. Once derived, this node has two modes. It can smooth the shot, eliminating unwanted jitter while maintaining the general motion of the camera. It can also lock the shot, stabilizing a subject within the frame that's isolated with a mask. This node can affect translation, rotation, zoom, and perspective, making it more flexible for certain operations than the other tracking nodes. For more information on using the *SmoothCam* node, see "The SmoothCam Node" on page 754.

How a Tracker Works

A tracker works by analyzing an area of pixels over a range of frames in order to "lock onto" a pattern as it moves across the screen. You specify the "snapshot" of pixels in one or more reference frames, then Shake proceeds to "track" that snapshot for a specified duration of time. In Shake, that snapshot is known as a *reference pattern*, and its area is defined by the inner box of the onscreen tracker control:

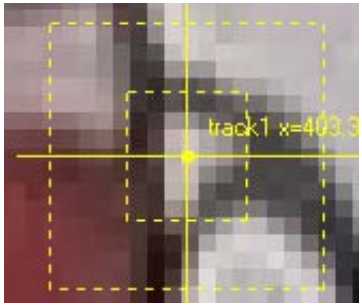


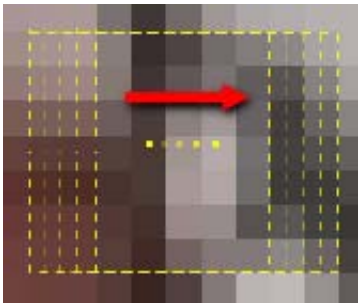
Ideally, the reference pattern should be some easily identifiable detail with high contrast—this makes it easier to track.

The tracker advances to each subsequent frame, sampling the area inside the *search region,* which is represented by the outer box of the onscreen tracker control. The tracker positions a box the same size as the reference pattern at the pixel in the first row, at the first column of the search region, and takes a sample. The tracker then advances to the next pixel (or subpixel) column in the search region and takes a second sample. For every sample the tracker takes, it assigns a correlation value by comparing the current sample to the previously designated reference pattern. When all of the samples have been taken, Shake assigns the new tracking point to the sample with the highest correlation value. This process is then repeated, every frame, until the end of the track range has been reached.

Using referenceBehavior

The referenceBehavior parameter controls if and when the reference pattern is ever updated. By default, the reference pattern is set to use the start frame (the first frame at which you start tracking) throughout the entire track, so even the samples within the last frame of the track are compared to the very first frame. You can change this behavior to periodically update the reference pattern when various criteria are met, which can help you to track subjects that change shape due to perspective or motion.

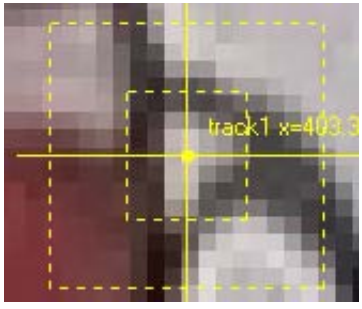


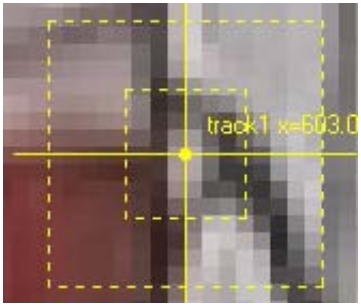


Setting subPixelResolution

The number of samples taken in the search region is determined by the subPixelResolution parameter. A subPixelResolution of 1 positions the reference pattern box at every pixel to find a sample. This is not very accurate, because most movement occurs at the subpixel level—the movement is more subtle than one pixel across, and therefore is factored in with other objects in the calculation of that pixel's color. The next resolution down in subPixelResolution, 1/4, advances the reference pattern box in .25 pixel increments, and is more accurate.

The example here is a theoretical 1/2 resolution since the pattern is advanced in .5 pixel increments. Keep in mind that the lower the number, the more samples taken. At 1/4 resolution, it takes 16 times more samples per pixel than at a resolution of 1. At 1/64, it takes 4096 times more samples per pixel.





For this reason, most trackers don't handle significant rotational movement very well—they (Shake's included) only test for panning changes, not rotational. If they did, they would have to multiply the amount of panning samples by the amount of degrees for the number of samples to take, which would be prohibitively costly at this stage. If you are tracking an object with rotational movement, try using a referenceBehavior set to update every frame. This means that the reference pattern is updated at every frame, so you are only comparing a frame with the frame before it, and not the first frame.

Also keep in mind that manual adjustments are a standard solution for many tracking problems.

This section discusses tracking in depth, including interface features, workflow issues, and tips on successful tracking and manipulation of tracking data. For specific formats of each node, see the functions listing at the end of this chapter. For a tutorial on how to use the Tracker, see Tutorial 7, "Tracking and Stabilization," in the *Shake 4 Tutorials*.

Image Tracking Workflow

The following is a general overview of the steps required to generate a track. The steps are further detailed in subsequent sections.

To generate a track:

1. Apply a motion tracking node to an image.
2. Double-click the tracking node to load its image into the Viewer and its parameters into the Parameters tab.

   Note: If you don't load the motion tracking node into the Viewer, the track will not be performed.
3. Play your background clip several times to determine a good tracking point.
4. Make sure that the onscreen controls are visible in the Viewer.
5. Go to the frame that you want to start the track.
6. Position the tracker on the point you want to track, then adjust the reference pattern and the search region boxes used to identify the desired tracking point.
7. Ensure the trackRange parameter reflects the frame range you want to track.

   For *FileIn* nodes, the default is the range of the clip. For Shake-generated elements, you must provide a frame range, for example, 1-50.
8. Click the Track Forward or Track Backward button (in the Viewer shelf) to begin processing.

   The track generates animation curves.
9. To stop a track, click the mouse button.

Stabilize Additions

If you are using the *Stabilize* node, include the following additional steps with the general steps above.

1 Determine if you need one-, two-, or four-point tracking.

Note: For two- or four-point tracking, select "2 pt" or "4 pt" in the trackType parameters.

In the *Stabilize* node parameters, set applyTransform to active in order to stabilize the plate.

2 If you are using the *Stabilize* node for matchmoving, toggle the inverseTransform parameter from stabilize to match.

MatchMove Additions

If you are using the *MatchMove* node, include the following additional steps with the general steps above.

1 Attach the foreground element to the first input of the *MatchMove* node.

2 Determine if you need one-, two-, or four-point tracking.

3 In the *MatchMove* parameters, set the outputType to Over (or another compositing operation).

4 Set the applyTransform parameter to active (to match the motion).

Note: If you are four-point tracking (make sure "4 pt" is enabled in the trackType parameter), and you want to attach four arbitrary points on the foreground image to the background, click the BG/FG button in the Viewer shelf to show the foreground.

Next, position the four corners on the foreground element. These represent the corners that are plugged into the four tracking points. Click the BG/FG button again to show the background.

5 You may have to set the outputType parameter to Over again (or whichever compositing mode you selected in step 3).

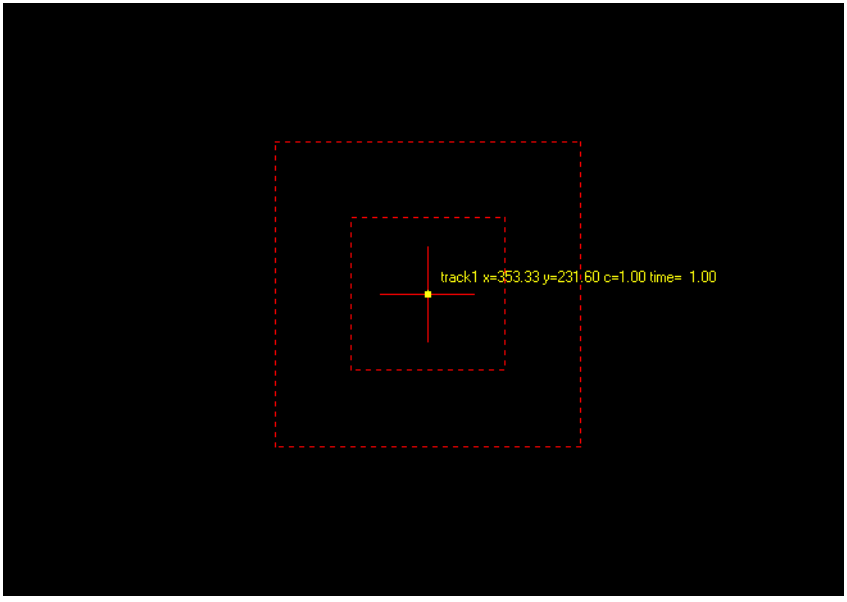
Adjusting the Onscreen Tracker Controls

The *Tracker*, *Stabilize*, and *MatchMove* nodes share common onscreen interface controls in the Viewer.

A tracker consists of three onscreen controls:

- *Search region:* The outer box
- *Reference pattern:* The inner box

- *Track point:* The center crosshairs



To move the tracker:

- Click a blank area inside of the search region or the track point, then drag.

To resize the tracking region or the reference pattern:

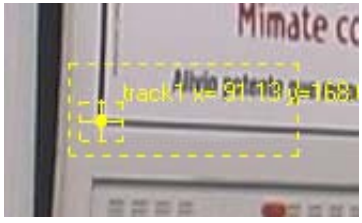
- Drag a corner and the boxes uniformly scale in the X or Y axes.

  The larger the search region, the slower the track.

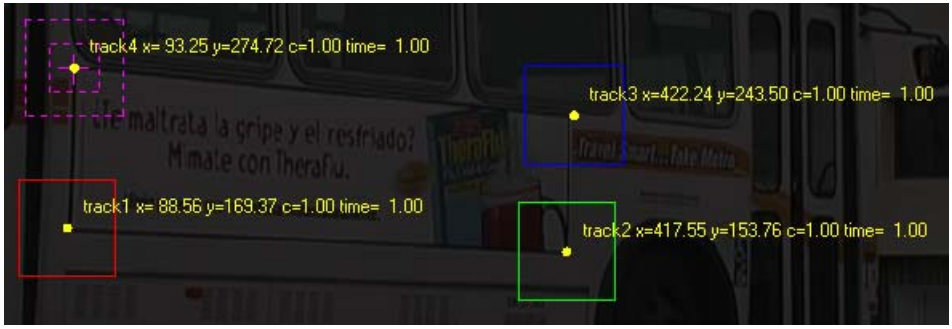
To scale the search region non-uniformly:

- Drag an edge of the search region.

  This is good for "leading" the track point. For example, a bus in a clip moves to the right. Scale the tracker search region to the right as well, since it doesn't make sense to scan to the left of the current track pattern for a pattern match. (This is demonstrated in Tutorial 7, "Tracking and Stabilization," in the *Shake 4 Tutorials*.)



Note: For four-point *MatchMove* and *Stabilize* operations, the trackers should be positioned in a counterclockwise order, starting in the lower-left corner. This ensures the proper alignment of your element when the transformation is applied.



If the reference pattern you are tracking goes offscreen, or becomes obscured by another object, Shake's default behavior is to stop the tracking analysis. You can use the Offset Track button to reposition the reference pattern in the Viewer. When you restart the tracking process, Shake will continue its motion analysis based on the new reference pattern, but will update (and keyframe) the original tracking point.

To offset (move) the onscreen tracker control to an unobstructed area of the image:

1 Cick the Offset Tracker button in the Viewer shelf.

2 Drag the onscreen tracker control (but not the tracking point crosshairs) to a better position in the Viewer.

3 Click the Track Forward or Track Backward button in the Viewer shelf to restart the motion analysis.

Shake continues to keyframe the movement of the original tracking point, based on the movement of the new, offset reference pattern.

Note: When you use the Offset Track function, be sure that the new reference pattern is in the same perspective plane as the originally tracked feature. If the two features are not approximately the same distance from the camera, the parallax effect will result in inaccurate motion tracking.

4 To reset the search area back to the original tracking point, click the Reset Track button.

You can turn off a tracker using the controls in the Parameters tab.

To turn off a specific tracker:

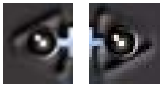
- Click the Visibility button located next to the track name in the Parameters tab.

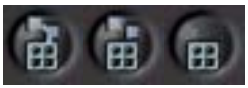
All visible trackers are processed when you click the track button, regardless of whether they have previously tracked keys. When Visibility is disabled for a tracker, that track is not processed.

A limitation of the *MatchMove* node is that it does not handle the pass-through of upstream onscreen controls, so those controls are in their untransformed state.

Viewer Shelf Controls

The following table describes the Viewer shelf buttons that become available with the tracking nodes.

Button		Description
	Track Backward/ Track Forward	Click to start the tracking process. All visible trackers attempt to lay down new tracking keyframes. Tracking continues until one of the trackRange frame limits is reached—the upper limit if you are tracking forward or the lower limit if you are tracking backward, or until your correlation falls below your failureTolerance setting.
	Offset Track–Off	The track search region and the tracking point are linked. If you move one, the other follows.
	Offset Track–On	The track search region and the tracking point are offset from each other. If your original sample pattern becomes obscured, offset the search region to a different area. Your keyframes are still saved relative to the original spot.
	Reset Track	Returns an offset track search region back to the track point.

Button		Description
	Track Display	Click to toggle between track display options: • The left button displays all trackers, curves, and keyframes. • The middle button displays just the trackers and keyframes. • The right button displays just the trackers. You can also control visibility of individual trackers as described below.
	FG/BG	These buttons appear on the Viewer shelf when the *MatchMove* node is active. Click FG/BG to toggle between adjusting the track points on the background (BG), or the corners of the foreground (FG) that are pinned to the track points (when "4 pt" [four-point tracking] is enabled in the trackType parameter).

Next to each track name is a Color Picker and a Visibility button.

To change the color of a tracker, click the color swatch.

Tracking Parameters

All trackers share the following parameters:

Parameter	Description	
trackRange	The trackRange parameter is the potential frame range limit of your track. By default, the range is set to the clip range. For generated elements such as *RGrad*, the default takes a range of 1. You can set new limits using Shake's standard range description, for example, 10-30x2. If you stop tracking and start again, processing starts from the current frame and proceeds to the lower or upper limit of your trackRange (depending on whether you are tracking forward or backward).	
subPixelResolution	The subPixelResolution parameter determines the resolution of your track. The smaller the number, the more precise the track. Possible values:	
	1	Area is sampled at every pixel. Not very accurate or smooth, but very fast.
	1/4	Area is sampled at every .25 pixels (16 times more than with a sampling of 1).
	1/16	Area is sampled at every .0625 pixels (256 times more than with a sampling of 1).
	1/32	Area is sampled at every .03125 pixels (1024 times more than with a sampling of 1).
	1/64	Area is sampled at every .015625 pixels (4096 times more than with a sampling of 1).

Parameter	Description	
matchSpace	The pixels are matched according to the correlation between the selected color space—luminance, hue, or saturation. When an image has roughly the same luminance, but contrasting hues, you should switch to hue-based tracking. You can also adjust the weight of the color channels in the matchSpace subtree.	
referenceTolerance	A tracking correlation of 1 is a perfect score—there is an exact match between the original reference frame and the sampled area. When the referenceTolerance is lowered, you accept greater inaccuracy in your track. If tracked keyframes are between the referenceTolerance and the failureTolerance, they are highlighted in the Viewer. Also, in some cases, referenceBehavior is triggered if the tracking correlation is below the referenceTolerance.	
referenceBehavior	This behavior dictates the tracking area reference sample. By default, the reference pattern is the first frame that the track is started, not necessarily the first frame of the trackRange. The last two behaviors in the referenceBehavior list measure the tracking correlation and match it to the referenceTolerance to decide an action.	
	use start frame	The new samples are compared to the reference pattern from the first frame of the track. If you stop tracking midway, and start again at a later frame, the later frame is used as the reference sample.
	update every frame	The source sample is updated from the previous frame. This usually creates an inherent drift in the track, as tiny errors accumulate. This method is for movements that have drastic changes in perspective and scale.
	update from keyframes	If you are using a failureBehavior of "predict location and don't create keys" or "don't predict location," a keyframe is not necessarily saved every frame. In this case, you may only want to update from the last frame with a valid keyframe.
	update if above reference tolerance	This updates the reference sample from the previous frame if the correlation is above the referenceTolerance. The intent is to update every frame unless you know the point is obscured. If you use a predict mode and know there are obstructions, it keeps the reference area from updating if the point is completely obscured.

Parameter	Description	
	update if below reference tolerance	This updates the reference sample from the previous frame if the correlation is below the referenceTolerance. This basically says, "If I can't get a good match, then resample." This is excellent for gradual perspective and scale shifts in the tracking area.
failureTolerance	If the correlation of a track falls below this value, it initiates the failureBehavior.	
failureBehavior	What occurs when the correlation drops below the failureTolerance:	
	stop	The tracker stops if the correlation is less than the failureTolerance. You can also press Esc to manually stop tracking.
	predict location and create key	If a failure is detected, then the tracker predicts the location of the keyframe based on a vector of the last two keyframes, and continues tracking in the new area.
	predict location and don't create key	Same as above, but it merely predicts the new search area and does not create new keyframes until a high correlation is obtained. This is excellent for tracked objects that pass behind foreground objects.
	don't predict location	In this case, the tracker merely sits in the same spot looking for new samples. New keyframes are not created.
	use existing key to predict location	This allows you to manually create keyframes along your track path. You then return to the start frame and start tracking. The search pattern starts looking where the preexisting motion path is.
limitProcessing	Creates a Domain of Definition (DOD) around the bounding boxes of all active trackers. Only that portion of the image is loaded from disk when tracking, so the track is faster. This has no effect on the final output image.	
preProcess	Toggles on the preprocessing for the tracking area. This applies a slight blur to reduce fluctuations due to grain. To control the blur amount, open the preProcess subtree.	
blurAmount	The amount of blur applied when preprocessing.	
trackNName	The name of the track. To change the name, click in the text field and enter the new name.	
trackNX/Y	The actual track point in X and Y. Use this to link a parameter to a track point.	

Parameter	Description
trackNCorrelation	The correlation value of that key to the original sample. A score of 1 is a perfect score. 0 is a completely unusable score.
trackNWindow Parameters	These multiple parameters control the windowing of the tracking box, and are not relevant to exported values.

Tracking Shortcut Menu

Right-click in the text field of a trackName to open a shortcut menu with options for manipulating your tracks.

Menu Item	Description
Copy/Paste	The standard copy and paste commands.
Load Track	Displays a list of all currently existing tracks. Select one, and a copy of that track is loaded into both the X and Y parameters of the current tracker.
Link Track	Displays a list of all currently existing tracks. A link is made from the current tracker to the tracker you select in the Select Track pop-up window. Therefore, if you modify the tracker selected with the Link command, the current tracker is updated. If you delete the tracker you linked to, you lose your tracking data.
Average Tracks	Displays a list of all tracks. Select up to four tracks that you want to average together, then click OK. An expression is entered into the current trackX and Y fields that links back to your averaged tracks. You cannot choose to average your current track—it is deleted by this action.
Smooth Track	Displays a slider to execute a blur function on your tracking curves; the smooth value is the number of keyframes considered in the smoothing operation. To view the curves, open the trackName subtree and click the Load Curve button to load the parameters into the Curve Editor. If you use Smoothing and Averaging operations as your standard workflow, it is recommended that you generate your tracks first with the *Tracker* node, and then link the tracks to a *MatchMove* or a *Stabilize* node.
Load Track File	Loads a Shake-formatted track file from disk. See the end of the tracking overview for the format of a track file.
Save Track File	Saves a Shake-formatted track file to disk.
Clear Track	Resets the current tracker. To reset the entire function, right-click an empty area of the Parameters tab, then choose Reset All Values from the shortcut menu.

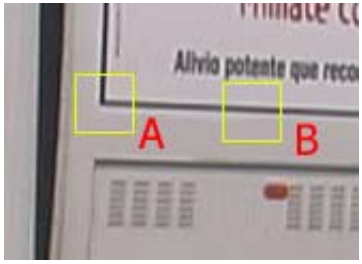
Strategies for Better Tracking

Unfortunately, tracking is rarely a magic bullet that works perfectly on the first attempt. This section discusses some strategies to help you get accurate tracks in various situations.

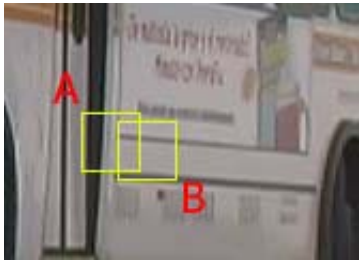
Picking a Good Reference Pattern

The ideal reference pattern is one that doesn't change perspective, scale, or rotation, and does not move offscreen or become obscured by other objects. The ideal pattern also maintains overall brightness or color, is very high contrast, and is distinct from other patterns in the same neighborhood. Meanwhile, in the real world, we have to contend with all of these factors in our footage.

In the following example, two possible reference pattern candidates include the corner at area A, or anywhere along the line near B. Area A is the better choice, since B can be matched anywhere along the horizontal black line, and probably on the dark line that is below B. One rule is to avoid similar horizontal or vertical patterns when selecting a candidate pattern. If you can easily find similar patterns, so can the tracker.

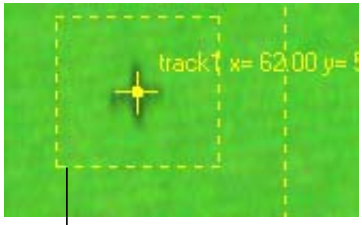


Another potential candidate is a word in the sign. However, as the clip advances, the text becomes an indecipherable blur. The A and B points also move closer together—the clip has significant scaling on the X axis and some scaling in Y due to the perspective shift. Although the overall brightness has dropped, contrast remains relatively high at the A point. The A point is a good candidate for tracking with referenceBehavior set to "update if below reference tolerance" or "update every frame."



The reference sample should also be relatively constant and unique over time. Flickering lights, for example, are not good reference patterns. If the lights were regular enough, you could try to set the trackRange to match the flicker, that is, 1-5, 10-15, 20-25, and so on. Granted, this is awkward. A better solution is to set your failureBehavior to "predict location and don't create key."

The following example shows a track marker placed on a TV screen so the client could place an image on the TV. The default tracker reference pattern is unnecessarily large. The only interesting detail is the black cross in the middle. Otherwise, most of the pattern matches up with most of the rest of the image—green. To adjust for this, limit the reference pattern to more closely match the black crosshairs.



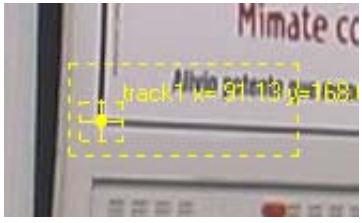
Default tracker reference pattern

Adjusted tracker reference pattern

Picking a Good Search Region

You should also suit your search region to match the clip's movement and the patterns near the reference pattern. With the default settings, the bus example clip does not track the lower-left corner of the sign very well. This is because the middle black horizontal line can easily be matched with the black line at the very base of the search region in later frames of the clip. The X axis is squeezed so much that vertical details disappear. Additionally, since the bus is moving to the right, there is no point in wasting cycles to the left of the point. Remember, the larger the search region, the more samples the tracker must take.

The corrected search region, illustrated below, is now high enough to not include the lower black line, and extends minimally to the left.



Manually Coax Your Track

Another technique you can use is to manually insert tracking keyframes. For example, if you have 100 frames to track, you can put in a keyframe every 5 or 10 frames with the Autokey feature. A helpful trick is to set an increment of 5 or 10 in the Time Bar. Press the Left Arrow or Right Arrow to jump by the increment amount.

Note: To add a keyframe without moving an onscreen control—for example, to create a keyframe at frame 20 with the same value as a keyframe at frame 19—turn Autokey off and then back on.

Once your keyframes are manually entered, return to frame 1, and set the failureBehavior to "use existing key to predict location." The tracker searches along the tracker's preexisting motion path to find matching patterns.

Identify the Color Channel With the Highest Contrast

The tracker works best with a high-contrast reference pattern. The human eye sees contrast as represented by value. However, you may sometimes have higher contrast in saturation or hue, so switch to a different color space with the matchSpace parameter in the tolerances subtree. A shot may also have a higher contrast in a specific RGB channel than in the other channels. For example, the blue channel may have a larger range than the red or green channel. In such a case, apply a Color–*Reorder* node to the image, set the channels parameter to *bbb*, and then perform the track with luminance selected as the matchSpace.

Delog Logarithmic Cineon Files Prior to Tracking

Because the logarithmic to linear conversion increases contrast, you may have better results on linear data than on logarithmic data.

Avoid Reducing Image Quality

Ideally, you should track an image with the most amount of raw data. This means if you apply a *Brightness* node set to .5 to your image, you lose half of the color information that could have been used for tracking. It is far better to track the image before the *Brightness* function is applied.

Do Not Track Proxies

Don't ever track an image with proxy settings enabled. Proxies are bad for two reasons: First, you filter the image so detail is lost. Second, you automatically throw data away because of data round-off. For example, if using 1/4 proxy, you automatically throw away four pixels of data in any direction, which means an 8 x 8 grid of potential inaccuracy.

Increasing Contrast and Preprocessing the Image

It is often helpful to apply a *Monochrome* node to an image, and drop the blue channel out if you have particularly grainy footage. Another good strategy is to activate the preProcess flag in the tracker. This applies a small blur to the footage to reduce irregularities due to video or film grain.

In some cases, you may want to modify your images to increase the contrast in the reference pattern, either with a *ContrastLum* node or *ContrastRGB* node. Since you're only using this image to generate tracks, you can disable or remove these nodes after you've finished the track.

Tracking Images With Perspective, Scale, or Rotational Shifts

For images with significant change in size and angle, you can try two different referenceBehaviors, "update if below reference tolerance" or "update every frame." The second choice is the more drastic, because you get an inherent accumulation of tiny errors when you update every frame. Therefore, try "update if below reference tolerance" first.

Another strategy is to jump to the midpoint frame of the clip and track forward to the end frame of the clip. Then return to the midpoint frame and track backward to the beginning of the clip.

A second strategy is to apply two *Stabilize* nodes. The first *Stabilize* node can be considered a rough stabilize. The second *Stabilize* then works off of the first, and therefore has a much better chance of finding acceptable patterns. Since the *Stabilize* nodes concatenate, no quality is lost.

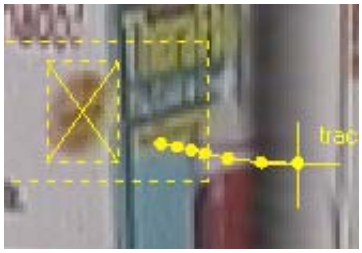
Tracking Obscured or Off-Frame Points

There are two basic techniques to correct track points that are obscured by moving off screen or an object passing in front of them.

The first strategy is to use a different failureBehavior, either "predict location and create key" or "predict location and don't create key." The first setting is good for predictable, linear motion behavior—it continues to lay down keyframes following the vector of the last two valid keyframes, so the two frames prior to the pattern become obscured. It is excellent for points that go off screen and never reappear. The second setting is a better choice for patterns that reappear because it continues to search on a vector, but only creates a keyframe if it finds another acceptable pattern. You have an even interpolation between the frame before the pattern was obscured, and the frame after it is revealed again. These strategies only work when the clip contains nice linear movement.

The second strategy is to use the Offset Tracker button (in the Viewer shelf). When the reference pattern becomes obscured, turning on the Offset Tracker button lets you move the tracking control, picking a new reference pattern and search region in a different area from the original reference pattern. The offset between the original reference pattern and the new one is calculated in order to maintain continuity in the resulting track path.

In the following example, the track is obscured by a lamp post, so the search region (not the point, just the region) is moved to a nearby pattern and tracking continues until the original pattern reappears. Even though one region is examined, the points are saved in another region. The second tracking pattern should travel in the same direction as your original pattern.



Modifying the Results of a Track

A track can be modified in several ways to massage the data from a less-than-perfect track. You can manually modify a track in the Viewer or in the Curve Editor, average tracks together, smooth tracks to remove noise, or remove jitter to smooth out a camera movement.

Manually Modifying Tracks

To manually adjust a tracking point onscreen, turn on the Autokey button in the Viewer shelf.

Note: To add a keyframe without moving an onscreen control, for example, to create a keyframe at frame 20 with the same value as a keyframe at frame 19, turn Autokey off and then back on.

Use the + and – keys (next to the Delete or Backspace key) to zoom in and out of the clip. The zoom follows the pointer, so place the pointer on the key point in the Viewer and zoom in. Press the Home key (near the Page Up and Page Down keys on your keyboard) or click the Home button in the Viewer shelf to return to normal view.

You can also adjust a tracking curve in the Curve Editor. In the tracking node parameters, open the trackName subtree and click the Load Curve button to load a parameter into the Curve Editor. In the following example, track1X (only the X parameter) is loaded into the Editor.

Averaging Tracks

A common technique is to track forward from the first frame to the last, create a second track, and track backward from the last frame to the first. These two tracks are then averaged together to (hopefully) derive a more accurate track. If you plan to use this method, it is recommended that you use the *Tracker* node, and then load your tracks into *Stabilize* or *MatchMove* with the Load or Link Track functions (in the trackName text field shortcut menu).

To average tracks:

1. Apply a *Tracker* node, and in the bottom of the *Tracker* parameters, click Add.
2. Click Add again.

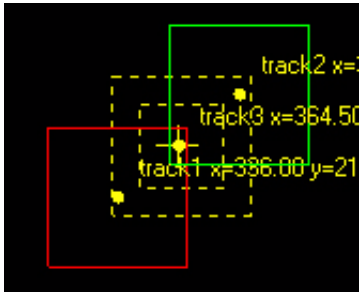
   There are a total of three trackers.

   Note: You could also potentially use tracks from any other *Stabilize*, *MatchMove,* or *Tracker* node.
3. Create tracks on track1 and track2.
4. Right-click track3, then choose Average Tracks from the shortcut menu.

5 Select Tracker1.track1 and Tracker1.track2 as the first two inputs, respectively, and leave the last two inputs set to none. The following illustration shows *Stabilize1* to remind you that any tracking node can be a track source.

6 Click OK.

The third track, track3, is in the middle of the first two tracks.



This works by creating an expression in both the track3X and track3Y parameters. The expression for the X parameter looks like this:

```
(Tracker1.track2X+Tracker1.track1X)/2
```

Because these are linked to track1 and track2 on the *Tracker1* node, do not delete them. For more information on linking, see "Linking to Tracking Data" on page 737.

You can average up to four tracks at one time, but you can of course continue to manipulate your tracks with further functions, including Average Tracks.

Smoothing Track Curves

You can smooth a track with the Smooth Tracks function in the *Tracker* parameters. Prior to smoothing the curve, you may want to copy the track (as a backup) to another tracker with the Load Track function on the second tracker.

To smooth a track curve:

1 Right-click the track you want to smooth, then choose Smooth Tracks from the shortcut menu.

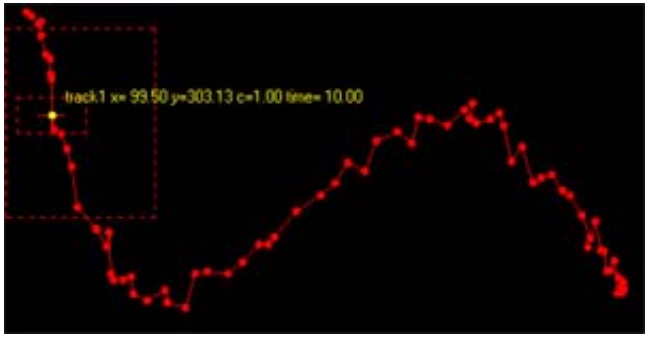
The Smooth Track window appears.

2 Enter a value (or use the slider) in the smoothValue field.

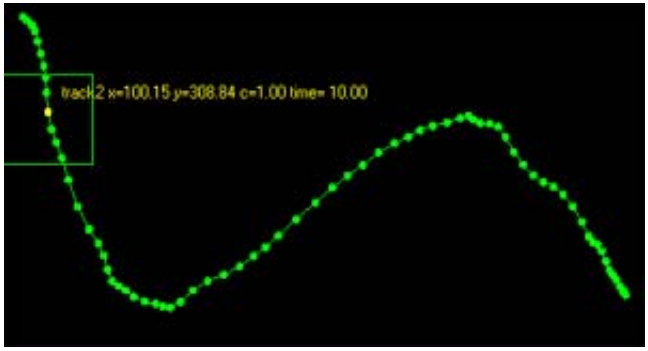
The default is 5, which means that 5 track points centered on the currently evaluated point are used to compute the current point's new, smoothed value. This is a standard Gaussian (bell-curve type) filter. In other words, if you leave it at 5, when the value of frame 12 is computed, frames 10, 11, 12, 13, and 14 are considered. If set to 3, it uses frames 11, 12, and 13.

The larger the smoothValue, the more points are considered (and thus more calculations done) for every point in the curve. Even values for smoothValue use the next largest odd number of frames, but the end ones do not contribute as much.

As an example, the following is a noisy track curve (before smoothing):



After the track curve is smoothed:



Linking to Tracking Data

Referencing track point data works similarly to referencing any other parameter within Shake. The twist here is that since you can rename the track point, you can change the name of the referred parameter. For example, if you have a *Tracker* node named *Tracker1*, and a track point set to its default name "track1," do one of the following:

- To reference the X track data, use: *Tracker1.track1X*.
- If you change the name of the track point to "lowerleft," then the reference is changed to *Tracker1.lowerleftX*.

  This applies to the Y data as well.
- You can also use Link Track (in the trackName text field shorcut menu) to link one track to another track, simultaneously linking the X and Y curves.

Removing Jitter on a Camera Move

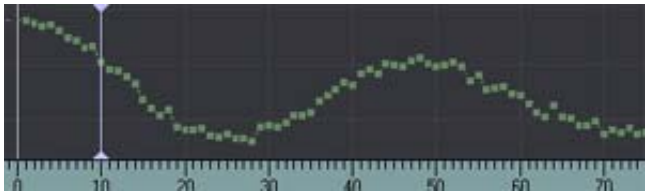
The following technique is useful when the clip contains a camera move that you want to preserve, but which has a lot of jitter. You need to stabilize the shot, but only by the small amount that is the actual jitter. To do this, you can combine the techniques mentioned above.

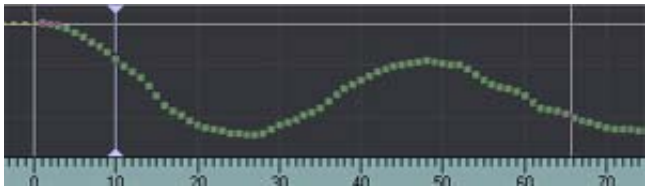
Note: The *SmoothCam* node was specifically developed to smooth out or lock irregular camera movement within a shot in a much simpler way. For more information, see "The SmoothCam Node" on page 754.

To remove jitter and preserve the camera move:

1. Track the plate with a *Tracker* node (for this example, called *Tracker1*).
2. In the *Tracker1* parameters, click Add to create a second tracker in the same node.
3. Load track1 into track2 using Load Track (right-click in the track2 text field).
4. Right-click track2, then choose Smooth Track from the shortcut menu.
5. Enter a smooth value, click Apply, then click Done.

At this point, you have a track and a smoothed version of that track. The following example shows the Y curves of the two tracks.





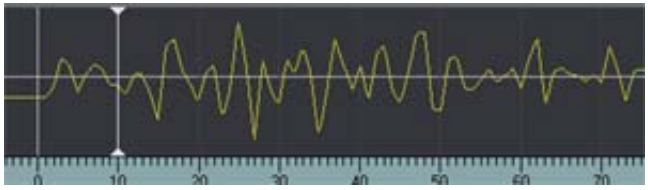
6 Create a *Stabilize* node.

7 In the *Stabilize* node, expand track1, then enter the following expression in the track1X and track1Y parameters:

- In track1X, enter: *Tracker1.track1X - Tracker1.track2X*
- In track1Y, enter: *Tracker1.track1Y - Tracker1.track2Y*

Thus, you get only the difference between the two curves—the jitter.

Note: This illustration is scaled differently in Y than the above illustrations.



8 In the *Stabilize* node parameters, set applyTransform to active.

The plate is only panned by the amount of the jitter, and maintains the overall camera move.

Working With Two-Point Tracking

There are several additional options available when working with two-point tracking. You can choose to pan, scale, and/or rotate the image. When setting the applyScale and applyRotate parameters, you have three choices: "none," "live," and "baked."

In the applyScale and applyRotate parameters, enable "live" to use the mathematical calculation of the four curves (track1X, track1Y, track2X, and track2Y)—live mode takes the track1 and track2 expressions and creates scale or rotational curves you can view in the Curve Editor.

To convert curves into editable data (generate keyframes), click "baked" in the applyScale and applyRotate parameters.

Note: Two-point (or four-point) tracking is only available in the *MatchMove* and *Stabilize* nodes (not the *Tracker* node).

Saving Tracks

Once a track is complete, you can save the track (in the *Tracker*, *MatchMove*, or *Stabilize* node) for use by another tracking function, a paint stroke, or to a *RotoShape* node.

Note: Although you cannot attach a saved track file to a paint stroke or *RotoShape* node, you can add a tracking node to your script, load the saved track file in the tracking node, then apply the track to your shape or paint stroke. For more information on attaching trackers to strokes and shapes, see "Attaching a Tracker to a Paint Stroke" on page 586 and "Attaching Trackers to Shapes and Points" on page 562.

To save a track file:

1. Create your track using the *Tracker*, *MatchMove*, or *Stabilize* node.
2. In the tracker Parameters tab, right-click the trackName field, then choose Save Track File from the shortcut menu.
3. In the "Save tracking data to file window," navigate to the directory in which you want to save the track file, then enter the track file name.
4. Click OK.

   The track file is saved.

To load a track file:

1. Add a *Tracker*, *MatchMove*, or *Stabilize* node.
2. In the tracker Parameters tab, right-click the trackName field, then choose Load Track File from the shortcut menu.
3. In the "Load tracking data from file" window, navigate to the track you want to load.
4. Click OK.

   The track is applied to the tracker.

Tracking File Format

The following is a sample saved track file for use with the Save Track File or Load Track File command (right-click in the track name text field to access the shortcut menu).

TrackName track1

Frame	X	Y	Correlation
1.00	462.000	210.000	1.000
2.00	405.000	192.000	1.000
etc...			

Note: The Save Track File or Load Track File command is different from the Load Expression command. The Load Expression command is available in the shortcut menu of any node's value field and looks for formatted Shake expressions. For example, to load the above information into a *Move2D* node, you must load two files, one for the xPan parameter and one for the yPan parameter. Their formats are similar to the following:

```
Linear(0,0462@1, 405@2, ....)
```

and

```
Linear(0,210@1, 192@2, ...)
```

Tracking Nodes

The following section includes the tracking functions located in the Transform Tool tab. For information on other Transform functions, see Chapter 26, "Transformations, Motion Blur, and AutoAlign," on page 763.

MatchMove

The *MatchMove* node is a dedicated tracking node to match a foreground element to a background element using one-point (panning), two-point (panning, scaling, or rotation), or four-point (corner pinning) tracking. Unlike the *Tracker* or *Stabilize* node, *MatchMove* can perform the compositing operation, or you can pass the transformed foreground out for further modifications (blur, color corrections, and so on) before you do a composite.

The *MatchMove* node can generate up to four tracking points, or you can load in other tracks (created in Shake or on disk). To load another track, right-click in a trackName text field, then choose Load Track from the shortcut menu.

Parameters

This node displays the following controls in the Parameters tab:

applyTransform

The foreground element is only transformed if applyTransform is active.

trackType

You can do one-point, two-point, or four-point matchmoves. Different options appear with the different types.

- *1 pt:* Pans the foreground element to match the tracking point. You can optionally turn off the X or Y movement.
- *2 pt:* Pops up two additional parameters, with the option of matching scaling and rotation of the background element (with the matchScale and matchRotate parameters).
- *4 pt:* Performs corner-pin matchmoving on the foreground element. The pan, scale, and angle parameters disappear. Toggle to FG mode in the Viewer to adjust the 4 points that are pinned to the background. These points appear as *sourceNX/YPosition*.

applyX

Turning off the *applyX* parameter prevents the foreground element from following the horizontal movement of the tracked subject.

applyY

Turning off the *applyY* parameter prevents the foreground element from following the vertical movement of the tracked subject.

transform parameters

This parameter contains the following subparameters:

- *xFilter, yFilter:* The transformation filter used. Different filters can be used for horizontal and vertical transformations. For more information on the different kinds of filters that are available and how they work, see "Filters Within Transform Nodes" on page 862.
- *motionBlur:* Enables the motion blur for the foreground element. A value of 0 is no blur; 1 is the high setting. A mid-value is a trade-off between speed and quality. This value is multiplied by the motionBlur parameter in the Globals tab.
- *shutterTiming:* A subparameter of motionBlur. Shutter length. 0 is no blur, whereas 1 represents a whole frame of blur. Note that standard camera blur is 180 degrees, or a value of .5. This value is multiplied by the shutterTiming parameter in the Globals tab.
- *shutterOffset:* A subparameter of motionBlur. The offset from the current frame that the blur is calculated. Default is 0; previous frames are less than 0. The Global parameter shutterOffset is added to this.
- *aspectRatio:* This parameter inherits the current value of the defaultAspect global parameter. If you're working on a nonsquare pixel or anamorphic image, it is important that this value is set correctly to avoid unwanted distortion in the image.

refFrame
The reference frame that is used to calculate the null state of the transformation. For example, scale has a value of 1 and rotate has a value of 0 at the reference frame.

outputType
A pop-up menu that lets you choose the compositing operation used to combine the foreground element you're adding to the scene against the background that you're tracking. Each menu option follows the standard Shake operator of the same name. To pass on a tracked foreground without compositing, select Foreground. You can also use this when modifying the foreground corner points, as the FG/BG button on the Viewer shelf switches this setting.

clipMode
Selects the output resolution of the node from the Background (1) or the Foreground (0).

applyScale
Opening the applyScale parameter reveals the scale parameter. In two-point mode, the toggle control next to the scale slider toggles scaling of the foreground on and off. Under this parameter is a subparameter that returns the actual scaling and rotation value used in the transformation. You have the option to view the live calculated curve, or to bake the curve to create editable data (keyframes).

scale
A slider that determines the calculated scale for two-point matching. The scale at the refFrame is equal to 1, and all other frames are in reference to that frame.

applyRotate
Opening the applyRotate parameter reveals the rotate parameter. In two-point mode, the toggle control next to the scale slider toggles rotation of the foreground on and off. You have the option to view the live calculated curve, or to bake the curve to create editable data (keyframes).

rotate
A slider that determines the calculated rotation for two-point matching. The angle at the refFrame is equal to 0, and all other frames are calculated with reference to that frame.

subPixelResolution
The resolution of your track. The smaller the number, the more precise and slower your tracking. Possible values include:

- *1:* Area is sampled at every pixel. Not very accurate or smooth, but very fast.
- *1/4:* Area is sampled at every .25 pixels (16 times more than with a sampling of 1).
- *1/16:* Area is sampled at every .0625 pixels (256 times more than with a sampling of 1).

- *1/32:* Area is sampled at every .03125 pixels (1024 times more than with a sampling of 1).
- *1/64:* Area is sampled at every .015625 pixels (4096 times more than with a sampling of 1).

tolerances

The tolerances subtree contains subparameters that let you control this node's behaviors when the tracking quality goes down.

- matchSpace: The pixels are matched according to the correlation between the selected color space—luminance, hue, or saturation. When an image has roughly the same luminance, but contrasting hues, you should switch to hue-based tracking.

  You can also adjust the weight of the color channels in the matchSpace subtree:

  - *redWeight, greenWeight, blueWeight:* Three subparameters with sliders let you weight how closely the tracking operation follows each color channel of the image being tracked. In general, the color channels with the most contrast for the feature you're tracking should be weighed most heavily. Color channels with minimal contrast for the feature you're tracking should be de-emphasized.
- *referenceTolerance:* A tracking correlation of 1 is a perfect score—there is an exact match between the original reference frame and the sampled area. When the referenceTolerance is lowered, you accept greater inaccuracy in your track. If tracked keyframes are between the referenceTolerance and the failureTolerance, they are highlighted in the Viewer. Also, in some cases, referenceBehavior is triggered if the tracking correlation is below the referenceTolerance.
- *referenceBehavior:* A pop-up menu that sets the tracking area reference sample. By default, the reference pattern is the first frame from which the track is started, not necessarily the first frame of the trackRange. The last two behaviors in the referenceBehavior list measure the tracking correlation and match it to the referenceTolerance to decide an action.

  The referenceBehavior pop-up menu contains the following options:

  - *use start frame:* The new samples are compared to the reference pattern from the first frame of the track. If you stop tracking midway, and start again at a later frame, the later frame is used as the reference sample.
  - *update every frame:* The source sample is updated from the previous frame. This usually creates an inherent drift in the track, as tiny errors accumulate. This method is for movements that have drastic changes in perspective and scale.
  - *update from keyframes:* If you are using a failureBehavior of "predict location and don't create keys" or "don't predict location," a keyframe is not necessarily saved every frame. In this case, you may only want to update from the last frame with a valid keyframe.

- *update if above reference tolerance:* This option updates the reference sample from the previous frame if the correlation is above the referenceTolerance. The intent is to update every frame unless you know the point is obscured. If you use a predict mode and know there are obstructions, this option keeps the reference area from updating if the point is completely obscured.
- *update if below reference tolerance:* This option updates the reference sample from the previous frame if the correlation is below the referenceTolerance. This option basically says, "If I can't get a good match, then resample." This approach is excellent for gradual perspective and scale shifts in the tracking area.

- *failureTolerance:* If the correlation value of the tracker's analysis falls below this value, Shake initiates the failureBehavior.
- failureBehavior: A pop-up menu containing the following options:
  - *stop:* The tracker stops if the correlation is less than the failureTolerance. You can also press Esc to manually stop tracking.
  - *predict location and create key:* If a failure is detected, then the tracker predicts the location of the keyframe based on a vector of the last two keyframes, and continues tracking in the new area.
  - *predict location and don't create key:* Same as above, but it merely predicts the new search area and does not create new keyframes until a high correlation is obtained. This is excellent for tracked objects that pass behind foreground objects.
  - *don't predict location:* In this case, the tracker merely sits in the same spot looking for new samples. New keyframes are not created.
  - *use existing key to predict location:* This allows you to manually create keyframes along your track path. You then return to the start frame and start tracking. The search pattern starts looking where the preexisting motion path is.

limitProcessing

This button sets a Domain of Definition (DOD) around the bounding boxes of all active trackers. Only that portion of the image is loaded from disk when tracking, so the track is faster. This function has no effect on the final output image.

preProcess

This button toggles preprocessing on and off for the tracking area. This applies a slight blur to reduce fluctuations due to grain. To control the blur amount, open the preProcess subtree.

- *blurAmount:* A subparameter of preProcess. Sets the amount of blur applied when preprocessing.

trackRange
The trackRange parameter is the potential frame range limit of your track. By default, the range is set to the clip range. For Shake-generated elements such as *RGrad*, it takes a range of 1. You can set new limits using Shake's standard range description, for example, 10-30x2. If you stop tracking and start again, the process starts from the current frame until it reaches the lower or upper limit of your trackRange, depending on whether you are tracking forward or backward.

track1Name, track2Name…
The name of the track. To change the name, click in the text field and type a new name. The number of trackXParameters corresponds to the number of points you selected in the *trackType* parameter. Each trackName parameter contains the following subparameters:

- *track1X:* The actual X value of the keyframed track point at that frame. Use this to link a parameter to a track point. This parameter defaults to the expression width/3.
- *track1Y:* The actual Y value of the keyframed track point at that frame. Use this to link a parameter to a track point. This parameter defaults to the expression height/3.
- *track1Correlation:* The correlation value representing how closely that keyframe matched the original sample. A score of 1 is a perfect score. 0 is an unusable score.
- *track1 Window Parameters:* These multiple parameters control the windowing of the tracking box, and are not relevant to exported values.
  - *track1CenterX:* Determines the horizontal position of the track point. This parameter defaults to the expression width/3.
  - *track1CenterY:* Determines the vertical position of the track point. This parameter defaults to the expression height/3.
  - *track1Visible:* This parameter is the same as the visibility button that's immediately to the right of the trackName parameter, and toggles visibility of that tracker box and its keyframes off and on.
  - *track1Enabled:* If trackEnabled is not turned on, that tracker will not be used during the next track analysis.

Stabilize

The *Stabilize* node is a dedicated tracking node that locks down an image, removing problems such as camera shake or gate weave. You can do one-point (panning), two-point (panning, scaling, or rotation), or four-point (corner-pinning) stabilization. Tracks can be generated in the *Stabilize* node, or can be read in. To read in a track, right-click in the trackName text field, then choose Load Track from the shortcut menu.

Parameters

This node displays the following controls in the Parameters tab:

applyTransform
The foreground element is only transformed if *applyTransform* is active.

inverseTransform
Inverts the transformation. Use this to "unstabilize" the shot. For example, you stabilize a shot with a *Stabilize*, apply compositing operations, and then copy the first *Stabilize* to the end of the node tree. By inverting the transformation, you return the shot to its former unstable condition.

trackType
You can do one-point, two-point, or four-point matchmoves. Different options appear with the different types.

- *1 pt:* Pans the foreground element to match the tracking point. You can optionally turn off the X or Y movement.
- *2 pt:* Opens two additional parameters, with the option to match scaling and rotation of the background element (with the matchScale and matchRotate parameters).
- *4 pt:* Performs cornerpin matchmoving on the foreground element. The pan, scale, and angle parameters disappear. Four track points are used to make the transformation.

applyX
Turning off the applyX parameter prevents the foreground element from following the horizontal movement of the tracked subject.

applyY
Turning off the applyY parameter prevents the foreground element from following the vertical movement of the tracked subject.

transform parameters
This section contains subparameters that control how the input image is transformed according to the derived tracking information.

- *xFilter, yFilter:* A pop-up menu that sets the transformation filter used. Different filters can be used for horizontal and vertical transformations. For more information on the different kinds of filters that are available and how they work, see "Filters Within Transform Nodes" on page 862.
- *transformationOrder:* A pop-up menu that sets the order that transformations are executed. The default setting is trsx. This means that transformations are performed in the following order: translate, rotate, scale, shear.
- *motionBlur:* Enables the motion blur for the foreground element. A value of 0 is no blur; 1 is the high setting. A mid-value is a trade-off between speed and quality. This value is multiplied by the motionBlur parameter in the Parameters tab.
- *shutterTiming:* A subparameter of motionBlur. Controls shutter length. 0 is no blur, whereas 1 represents a whole frame of blur. Note that standard camera blur is 180 degrees, or a value of .5. This value is multiplied by the global shutterTiming parameter.

- *shutterOffset:* A subparameter of motionBlur. Controls the offset from the current frame that the blur is calculated. Default is 0; previous frames are less than 0. The global shutterOffset parameter is added to this.
- *aspectRatio:* This parameter inherits the current value of the defaultAspect global parameter. If you're working on a nonsquare pixel or anamorphic image, it is important that this value is set correctly to avoid unwanted distortion in the image.

refFrame
The reference frame that is used to calculate the null state of the transformation. For example, scale has a value of 1 and rotate has a value of 0 at the reference frame.

applyScale
A parameter that becomes available when the tracker is set to two-point mode. Three buttons that allow you to disable the transform (click "none"), view the live calculated curve (click "live"), or to bake the curve (click "bake") to create editable data (keyframes).
- *scale:* Opening the applyScale subtree reveals the scale slider, which determines the calculated scale for two-point matching. The scale at the refFrame is equal to 1, and all other frames are in reference to that frame.

applyRotate
A parameter that becomes available when the tracker is set to two-point mode. Three buttons that allow you to disable the transform (click "none"), view the live calculated curve (click "live"), or to bake the curve (click "bake") to create editable data (keyframes).
- *rotate:* Opening the applyRotate subtree reveals the rotate parameter. A slider determines the calculated rotation for two-point matching. The angle at the refFrame is equal to 0, and all other frames are calculated with reference to that frame.

subPixelResolution
The resolution of your track. The smaller the number, the more precise and slower your tracking analysis. The possible values are:
- *1:* Area is sampled at every pixel. Not very accurate or smooth, but very fast.
- *1/4:* Area is sampled at every .25 pixels (16 times more than with a sampling of 1).
- *1/16:* Area is sampled at every .0625 pixels (256 times more than with a sampling of 1).
- *1/32:* Area is sampled at every .03125 pixels (1024 times more than with a sampling of 1).
- *1/64:* Area is sampled at every .015625 pixels (4096 times more than with a sampling of 1).

Note: Most current computers are fast enough to quickly compute motion tracking even at 1/64 resolution, so don't feel the need to be conservative with this setting.

tolerances
The tolerances subtree contains subparameters that let you control this node's behaviors when the tracking quality decreases.

- *matchSpace:* The pixels are matched according to the correlation between the selected color space—luminance, hue, or saturation. When an image has roughly the same luminance, but contrasting hues, you should switch to hue-based tracking.

You can also adjust the weight of the color channels in the matchSpace subtree.

- *matchSpace (subtree):* Three subparameters with sliders let you weight how closely the tracking operation follows each color channel of the image being tracked. In general, the color channels with the most contrast for the feature you're tracking should be weighted most heavily. Color channels with minimal contrast for the feature you're tracking should be de-emphasized.
- *referenceTolerance:* A tracking correlation of 1 is a perfect score—there is an exact match between the original reference frame and the sampled area. When the referenceTolerance is lowered, you accept greater inaccuracy in your track. If tracked keyframes are between the referenceTolerance and the failureTolerance, they are highlighted in the Viewer. Also, in some cases, referenceBehavior is triggered if the tracking correlation is below the referenceTolerance.
- *referenceBehavior:* A pop-up menu that sets the tracking area reference sample. By default, the reference pattern is the first frame at which the track analysis begins, not necessarily the first frame of the trackRange. The last two behaviors in the referenceBehavior list measure the tracking correlation and match it to the referenceTolerance to decide an action.

  The referenceBehavior pop-up menu contains the following options:

  - *use start frame:* The new samples are compared to the reference pattern from the first frame of the track. If you stop tracking midway, and start again at a later frame, the later frame is used as the reference sample.
  - *update every frame:* The source sample is updated from the previous frame. This usually creates an inherent drift in the track, as tiny errors accumulate. This method is for movements that have drastic changes in perspective and scale.
  - *update from keyframes:* If you are using a failureBehavior of "predict location and don't create keys" or "don't predict location," a keyframe is not necessarily saved at every frame. In this case, you may only want to update from the last frame with a valid keyframe.
  - *update if above reference tolerance:* This option updates the reference sample from the previous frame if the correlation is above the referenceTolerance. The intent is to update every frame unless you know the point is obscured. If you use a predict mode and know there are obstructions, this option keeps the reference area from updating if the point is completely obscured.
  - *update if below reference tolerance:* This option updates the reference sample from the previous frame if the correlation is below the referenceTolerance. This basically says, "If I can't get a good match, then resample." This option is excellent for gradual perspective and scale shifts in the tracking area.

- *failureTolerance:* If the correlation value of the tracker's analysis falls below the value in this field, Shake initiates the failureBehavior.
- *failureBehavior:* A pop-up menu containing the following options:
  - *stop:* The tracker stops if the correlation is less than the failureTolerance. You can also press Esc to manually stop tracking.
  - *predict location and create key:* If a failure is detected, then the tracker predicts the location of the keyframe based on a vector of the last two keyframes, then continues tracking in the new area.
  - *predict location and don't create key:* Same as above, but this option merely predicts the new search area and does not create new keyframes until a high correlation is obtained. This option is excellent for tracked objects that pass behind foreground objects.
  - *don't predict location:* In this case, the tracker merely sits in the same spot looking for new samples. New keyframes are not created.
  - *use existing key to predict location:* This option allows you to manually create keyframes along your track path. You then return to the start frame and begin tracking. The search pattern starts looking along the preexisting motion path.

limitProcessing

This button creates a Domain of Definition (DOD) around the bounding boxes of all active trackers. Only that portion of the image is loaded from disk when tracking, so the track analysis is faster. This setting has no effect on the final output image.

preProcess

This button turns on preprocessing for the tracking area, applying a slight blur to reduce fluctuations due to grain. To control the blur amount, open the preProcess subtree.

- *blurAmount:* A subparameter of preProcess that sets the amount of blur applied when preprocessing to improve the quality of tracks in clips with excessive grain or noise.

trackRange

The trackRange parameter is the potential frame range limit of your track. By default, the range is set to the clip range. For Shake-generated elements such as *RGrad*, this parameter takes a range of 1. You can set new limits using Shake's standard range description, for example, 10-30x2. If you stop tracking and start again, the analysis starts from the current frame until it reaches the lower or upper limit of your trackRange, depending on whether you are tracking forward or backward.

track1Name, track2Name…
The name of the track. To change the name, click in the text field and type a new name. The number of tracks corresponds to the number of points you selected in the trackType parameter. Each trackName parameter contains the following subparameters:

- *track1X:* The actual X value of the keyframed track point at that frame. Use this to link a parameter to a track point. This parameter defaults to the expression width/3.
- *track1Y:* The actual Y value of the keyframed track point at that frame. Use this to link a parameter to a track point. This parameter defaults to the expression height/3.
- *track1Correlation:* The correlation value representing how closely that keyframe matched the original sample. A score of 1 is a perfect score. 0 is an unusable score.
- *track1 Window Parameters:* These multiple parameters control the windowing of the tracking box, and are not relevant to exported values.
  - *track1CenterX:* Determines the horizontal position of the track point. This parameter defaults to the expression width/3.
  - *track1CenterY:* Determines the vertical position of the track point. This parameter defaults to the expression height/3.
  - *track1Visible:* This parameter is the same as the visibility button that's immediately to the right of the trackName parameter, and toggles visibility of that tracker box and its keyframes off and on.
  - *track1Enabled:* If trackEnabled is not turned on, that tracker will not be used during the next track analysis.

Tracker

The *Tracker* node is used only to generate and contain tracking information. Unlike the *MatchMove* and *Stabilize* functions, it has no capability to alter the input image. As such, the *Tracker* node is used to create tracks that are referenced either in other tracking nodes or in non-tracking nodes such as *Move2D*, *Rotate*, and so on.

While *MatchMove* and *Stabilize* are limited to creating one, two, or four track points, the *Tracker* node can hold as many trackers as you want. To add a tracker, click the Add button at the bottom of the Parameters tab. If your workflow continually uses Smoothing and Averaging of tracks for application in a *Stabilize* or *MatchMove*, you should probably generate the tracking data using a *Tracker* node.

With the *Tracker* node, you can delete trackers, and save the tracks to disk. To save an individual track, right-click that tracker's trackName field, then choose Save Track from the shortcut menu.

> ***Warning:*** You cannot clone a *Tracker* node in the Node View using the Paste Linked command.

Parameters

This node displays the following controls in the Parameters tab:

trackRange

The trackRange parameter is the potential frame range limit of your track. By default, the range is set to the clip range. For Shake-generated elements such as *RGrad*, this parameter takes a range of 1. You can set new limits using Shake's standard range description, for example, 10-30x2. If you stop tracking and start again, the analysis starts from the current frame until it reaches the lower or upper limit of your trackRange, depending on whether you are tracking forward or backward.

subPixelResolution

The resolution of your track. The smaller the number, the more precise and slower your tracking analysis. The possible values are:

- *1:* Area is sampled at every pixel. Not very accurate or smooth, but very fast.
- *1/4:* Area is sampled at every .25 pixels (16 times more than with a sampling of 1).
- *1/16:* Area is sampled at every .0625 pixels (256 times more than with a sampling of 1).
- *1/32:* Area is sampled at every .03125 pixels (1024 times more than with a sampling of 1).
- *1/64:* Area is sampled at every .015625 pixels (4096 times more than with a sampling of 1).

matchSpace

The pixels are matched according to the correlation between the selected color space—luminance, hue, or saturation. When an image has roughly the same luminance, but contrasting hues, you should switch to hue-based tracking.

referenceTolerance

A tracking correlation of 1 is a perfect score—there is an exact match between the original reference frame and the sampled area. When the referenceTolerance is lowered, you accept greater inaccuracy in your track. If tracked keyframes are between the referenceTolerance and the failureTolerance, they are highlighted in the Viewer. Also, in some cases, referenceBehavior is triggered if the tracking correlation is below the referenceTolerance.

referenceBehavior

This behavior dictates the tracking area reference sample. By default, the reference pattern is the first frame at which the track analysis begins, not necessarily the first frame of the trackRange. The last two behaviors in the referenceBehavior list measure the tracking correlation and match it to the referenceTolerance to decide an action. The referenceBehavior pop-up menu contains the following options:

- *use start frame:* The new samples are compared to the reference pattern from the first frame of the track. If you stop tracking midway, and start again at a later frame, the later frame is used as the reference sample.

- *update every frame:* The source sample is updated from the previous frame. This usually creates an inherent drift in the track, as tiny errors accumulate. This method is for movements that have drastic changes in perspective and scale.
- *update from keyframes:* If you are using a failureBehavior of "predict location and don't create keys" or "don't predict location," a keyframe is not necessarily saved every frame. In this case, you may only want to update from the last frame with a valid keyframe.
- *update if above reference tolerance:* This option updates the reference sample from the previous frame if the correlation is above the referenceTolerance. The intent is to update every frame unless you know the point is obscured. If you use a predict mode and know there are obstructions, this option keeps the reference area from updating if the point is completely obscured.
- *update if below reference tolerance:* This option updates the reference sample from the previous frame if the correlation is below the referenceTolerance. This option basically says, "If I can't get a good match, then resample." It is excellent for gradual perspective and scale shifts in the tracking area.

failureTolerance
If the correlation value of the tracker's analysis falls below the value in this field, Shake initiates the failureBehavior.

failureBehavior
A pop-up menu containing the following settings:
- *stop:* The tracker stops if the correlation is less than the failureTolerance. You can also press Esc to manually stop tracking.
- *predict location and create key:* If a failure is detected, then the tracker predicts the location of the keyframe based on a vector of the last two keyframes, then continues tracking in the new area.
- *predict location and don't create key:* Same as above, but this option merely predicts the new search area and does not create new keyframes until a high correlation is obtained. This option is excellent for tracked objects that pass behind foreground objects.
- *don't predict location:* In this case, the tracker merely sits in the same spot looking for new samples. New keyframes are not created.
- *use existing key to predict location:* This allows you to manually create keyframes along your track path. You then return to the start frame and start tracking. The search pattern starts looking where the preexisting motion path is.

limitProcessing
This button creates a Domain of Definition (DOD) around the bounding boxes of all active trackers. Only that portion of the image is loaded from disk when tracking, so the track is faster. This setting has no effect on the final output image.

tolerances
The tolerances subtree contains subparameters that let you control this node's behaviors when the tracking quality decreases.

matchSpace
Not to be confused with the matchSpace parameter above—the matchspace subtree has three subparameters with sliders that let you weight how closely the tracking operation follows each color channel of the image being tracked. In general, the color channels with the most contrast for the feature you're tracking should be weighted most heavily. Color channels with minimal contrast for the feature you're tracking should be de-emphasized.

preProcess
This button turns on preprocessing for the tracking area, applying a slight blur to reduce fluctuations due to grain. To control the blur amount, open the preProcess subtree.

- *blurAmount:* A subparameter of preProcess that sets the amount of blur applied when preprocessing to improve the quality of tracks in clips with excessive grain or noise.

track1Name, track2Name…
The name of the track, itself a subtree of parameters containing the data for that particular tracking region. To change the name, click in the text field and enter the new name. The number of tracks corresponds to the number of tracking regions you've created using the Add and Delete buttons. Each trackName parameter contains the following subparameters:

- *track1X:* The actual X value of the keyframed track point at that frame. Use this to link a parameter to a track point. This parameter defaults to the expression width/3.
- *track1Y:* The actual Y value of the keyframed track point at that frame. Use this to link a parameter to a track point. This parameter defaults to the expression height/3.
- *track1Correlation:* The correlation value representing how closely that keyframe matched the original sample. A score of 1 is a perfect score. 0 is an unusable score.
- *track1 Window Parameters:* The parameters within the track1Window Parameters section define the size and position of the tracker boxes used to perform the motion tracking analysis. Each of the position and sizing parameters in this section is transient, meaning they're not saved with the project. If you close a project, these parameters return to their defaults when that project is reopened. The following are the default expressions that each parameter is set to.
  - *track1Left:* width/2-height/30
  - *track1Right:* width/2+height/30
  - *track1Bottom:* height/2-height/30
  - *track1Top:* height/2+height/30
  - *track1leftSearch:* width/2-height/15
  - *track1RightSearch:* width/2+height/15

- *track1BottomSearch:* height/2-height/15
- *track1TopSearch:* height/2+height/15
- *track1CenterX:* width/2
- *track1CenterY:* height/2
- *track1Visible:* This parameter is the same as the visibility button that's immediately to the right of the trackName parameter.
- *track1Enabled:* If trackEnabled is not turned on, that tracker will not be used during the next track analysis.

Add, Delete, Save, Load
These buttons allow you to create and remove additional tracking regions. You can also save tracking data, and read it back in using the Save and Load buttons.

CornerPin
The *CornerPin* node can be used to push the four corners of an image into four different positions, or to extract four positions and place them into the corners. For more information on the *CornerPin* node, see "*CornerPin*" on page 795.

The SmoothCam Node
This transformation node differs from the other tracking nodes described previously in this chapter in that it doesn't focus the track on small groups of pixels. Instead, it evaluates the entire image at once, using advanced motion analysis techniques to extract transformation data.

Once this information is derived, this node has two modes. It can smooth the shot, eliminating unwanted jitter while maintaining the general motion of the camera, or it can lock the shot, stabilizing the subject. This node can affect translation, rotation, zoom, and perspective, making it more flexible for certain operations than the other tracking nodes.

The *SmoothCam* node is primarily useful for removing unwanted trembling from less than stable crane or jib arm moves, eliminating teetering from handheld walking shots, or reducing vibrations in automotive shots. Secondarily, the *SmoothCam* node can be used to stabilize shots that may be difficult to lock using the *Stabilize* node.

As useful as the *SmoothCam* node is, be aware that motion blur that is present in the image will remain, even though the subject in the shot is successfully smoothed or locked. This may or may not affect your approach to the composite.

Important: Interlaced images *must* be deinterlaced prior to use with the *SmoothCam* node. For more information, see "Setting Up Your Script to Use Interlaced Images" on page 196.

Masking Important Features

The *SmoothCam* node has two inputs. The first one is for the input image to be processed. The second input is for an optional matte with which you can isolate a subject or area that you want the *SmoothCam* node to ignore while performing its analysis.

When creating a matte to use with the *SmoothCam* node, white areas are ignored, and black areas are analyzed.

Using the SmoothCam Node

The *SmoothCam* node works similarly to the *AutoAlign* node, in that the input image must be analyzed first in order to derive the data that is used to smooth or lock the image. This data is stored within the Shake script itself, so that each clip needs to be analyzed only once.

To analyze the media using the *SmoothCam* node:

1. Attach a *SmoothCam* node to the image you want to process.
2. Load the *SmoothCam* node's parameters into the Parameters tab.
3. If necessary, set the analysisRange to the frame range you want to process.

   By default, the analysisRange is set to the number of frames possessed by the media of the *FileIn* node to which it is attached.
4. Choose an analysisQuality mode. Start with Normal, which provides excellent results in most cases.

   Note: If, at the end of this process, the result is not what you'd hoped, then you should try again with the analysisQuality mode set to high. Be aware this will significantly increase the time it takes to perform this analysis.
5. Click the "analyze" button.

   The frames within the frame range are analyzed, and this data is stored within your script. To stop the analysis at any time, press Esc.

After the analysis has been performed, it's time to choose the mode with which you want the *SmoothCam* node to process the image.

To smooth a shot using the *SmoothCam* node:

1. Once the analysis has concluded, set steadyMode to smooth.
2. Open the smooth subtree.

3 Adjust the translationSmooth, rotationSmooth, and zoomSmooth sliders to increase or decrease the amount of smoothing that is attempted. At 0, no smoothing is attempted in that dimension. At higher values, *SmoothCam* attempts to smooth a wider range of variations in the movement of the image from one frame to the next.

- *translationSmooth:* Smooths the X and Y motion of the shot.
- *rotationSmooth:* Smooths rotation in the shot.
- *zoomSmooth:* Smooths a zoom within the shot.

Important: It's faster, and will yield more accurate results, if you set the smoothing parameters of dimensions in which you know the camera is *not* moving to 0. For example, if you know for a fact that the camera is neither rotating nor zooming, set rotationSmooth and zoomSmooth to 0. In particular, if the camera is not zooming, increasing the zoomSmooth value may actually cause unwanted transformations in the image.

To lock and match shots using the *SmoothCam* node:

1 Once the analysis has concluded, set steadyMode to lock.

2 Open the lockDown subtree.

3 The setting of the inverseTransform parameter depends on what you're trying to do:

- If you want to lock the analyzed shot itself, leave inverseTransform set to lock.
- If you want to apply the motion from this shot to another image, attach this node to the other image, then set inverseTransform to match.

4 Turn on the locking parameters for the dimensions of motion you want to affect.

- *translateLock:* Locks the subject in the X and Y dimensions.
- *rotateLock:* Locks the rotation of the subject.
- *zoomLock:* Removes zoom from the shot, maintaining the relative size of the subject.
- *perspectiveLock:* Locks the perspective of the subject, performing the equivalent of a four-corner warp to maintain the size and shape of the subject.

Troubleshooting SmoothCam Effects

If the output is unsatisfactory, there are several things you can try to improve the result.

Change the Smoothing Parameters

If you're trying to smooth the motion in a shot, you should first try changing the smoothing parameters. This can be accomplished without having to reanalyze the clip.

Reanalyze the Media at a Higher Quality

Next, try changing the analysisQuality to high, and reanalyze the media.

Try Editing the Analysis Data

If neither of the prior solutions helps, try loading the confidence parameter into the Curve Editor, then look for frames where the confidence parameter falls to 0. If the image transformation at these frames stands out, you can try loading the translationX, translationY, rotation, and/or zoom parameters within the motion subtree into the Curve Editor, then delete any keyframes that create unusual spikes at those frames.

Removing Black Borders Introduced by Smoothing and Locking

When you use the *SmoothCam* node, the resulting transformations that are made to the input image either to smooth or lock the shot cause moving black borders to appear around the edges of the output image. While this is necessary to achieve the desired effect, you probably don't want these black borders to appear in the final shot.

There are several ways you can choose to handle this border.

Using the clipMode Parameter

The clipMode parameter provides different ways you can treat the size of the output frame in order to include or exclude this black region. After you've analyzed the input image and picked the settings necessary to smooth or lock your shot, you can choose one of three clipMode parameters.

union

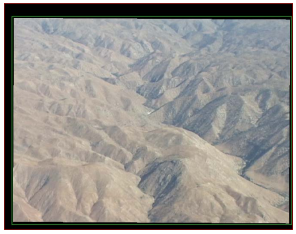
This option expands the rendered frame to include the full area within which the input image is transformed. If you scrub through the *SmoothCam*'s output with this option turned on, the image appears to float about within a black frame larger than itself. The frame size of the output image is larger than that of the input image.

intersection

This option contracts the rendered frame to exclude any black area. The result is a stable output image that fills the frame, but with a frame size that's smaller than the input image.

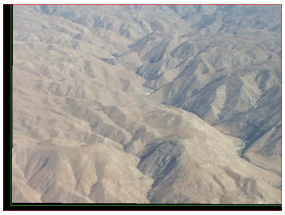
in

This option maintains the frame size of the input image as that of the output image. The result is a moving black area that encroaches around the edges of the output image.



Union

Intersection



In

You can use one of the above three clipMode options to produce the output image most useful for your purposes.

Note: Whichever clipMode you use, areas of the image that end up being clipped are preserved by Shake's Infinite Workspace, available for future operations.

Scaling the Output Image to Fit the Original Frame Size

If you need to output the resulting image at the same size as the original, the quickest fix is to leave the clipMode Parameter set to "in," and use a *Scale* node after the *SmoothCam* node to enlarge the image to the point where all instances of black borders fall outside the edge of the frame. The disadvantage of this method is the resulting softening of the image, depending on how much it must be enlarged.

Painting the Edges Back In

A more time-intensive, but possibly higher-quality solution would be to paint the missing edge data with a *QuickPaint* node. A variety of techniques can be employed, including using the clone brush to sample other parts of the image, or using the Reveal Brush to paint in parts of a secondary clean-plate image that you create separately.

Warping the Edges

One last suggestion would be to experiment with the different warping nodes, to stretch the edges of the image to fill any gaps. For example, you can experiment with the *LensWarp* node to stretch the edges of the image out. This solution is highly dependent on the type of image, and may introduce other image artifacts that may or may not be acceptable.

Parameters in the SmoothCam Node

This node displays the following controls in the Parameters tab:

analysisRange

The range of frames of the input image to analyze. By default, this parameter inherits the number of frames in the source media file represented by the *FileIn* node to which it's connected, not the timeRange in the Globals tab.

processedRange

This is not a user-editable parameter. It indicates the overall frame range that has been analyzed.

analysisQuality

Defines the level of detail for the motion analysis. There are two levels of quality, "normal" and "high." In most situations, "normal" should produce an excellent result. Should you notice any artifacts, set this parameter to "high."

analyze

Click this button to analyze the frame range defined by the analysisRange parameter. Pressing Esc while Shake is analyzing stops the analysis.

clipMode

The clipMode parameter provides different ways you can treat the size of the output frame in order to include or exclude black regions introduced by the *SmoothCam* node's transformations. The clipMode parameter has three options:

- *union:* Expands the rendered frame to include the full area within which the input image is transformed. If you scrub through the *SmoothCam* node's output with this option turned on, the image appears to float about within a black frame larger than itself. The frame size of the output image is larger than that of the input image.
- *intersection:* Contracts the rendered frame to exclude any black area. The result is a stable output image that fills the frame, but with a frame size that's smaller than the input image.

- *in:* Maintains the frame size of the input image as that of the output image. The result is a moving black area that encroaches around the edges of the output image.

steadyMode

The *SmoothCam* node has two modes: Smooth and Lock.

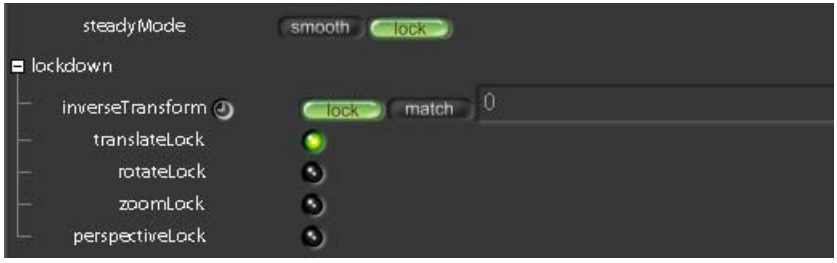
- *Smooth:* This mode smooths the apparent motion of the camera, while allowing the general movement in the frame to proceed. It's useful for removing jitter from a camera movement. When enabled, this mode has three sliders for each of the dimensions that can be smoothed.

The amount of smoothing can be increased or decreased along a sliding scale.

- *translationSmooth:* Smooths motion in both the X and Y dimensions.
- *rotationSmooth:* Smooths image rotation.
- *zoomSmooth:* Smooths an uneven zoom.

Note: Don't turn on zoomSmooth unless you're positive that the image is being zoomed. Otherwise this parameter will not have the desired effect.

- *Lock:* This mode attempts to lock the motion of the principal subject in the shot to eliminate motion. As a result, the background will appear to move around the subject being tracked.



- *InverseTransform:* This control lets you invert the effect of this node, so that other images can be matchmoved using the same motion.
- *translateLock:* Locks the image in both X and Y dimensions.
- *rotateLock:* Locks the rotation of the image.

- *zoomLock:* Locks an image that is being zoomed.

  Note: Don't turn on zoomLock unless you're absolutely positive that the image is being dynamically zoomed.

- *perspectiveLock:* Locks an image experiencing a change in perspective, similar to a reverse corner-pin.

Motion

The values within the parameters in the motion subtree are not meant to be editable, or even directly intelligible. They're available to be loaded into the Curve Editor so that you can find possible spikes in the analysis data that, along with a low confidence value, might indicate an error. In this case, you have the option of deleting the offending keyframe, in order to see if the resulting smoothing or locking operation becomes more acceptable.

- *confidence:* Unlike the tracking nodes, the confidence value is restricted to one of three possible values. You can load the confidence parameter into the Curve Editor to quickly find problem areas in the analysis. The values are:
  - *1:* Indicates an analysis in which Shake has high confidence. A keyframe is generated at these frames.
  - *0.5:* Indicates an uncertain analysis. Shake also generates a keyframe at these frames.
  - *0:* Indicates that Shake has no confidence in the analysis. No keyframe is generated.
- *translationX, translationY:* Contains the X and Y translation data.
- *rotation:* Contains the rotation data.
- *zoom:* Contains the zoom data.

Transformations, Motion Blur, and AutoAlign

26

Shake's transformation nodes provide many ways to geometrically manipulate the position, size, and orientation of images in your composition. The parameters within these nodes can also be animated—either manually or using expressions—to create motion and accompanying motion-blur effects.

About Transformations

Shake has a wide variety of nodes that can be used to create various kinds of transformations. Many of the transformation nodes—found in the Transform Tool tab—perform simple operations, such as pan, rotate, scale, shear, and corner-pin (four-corner warping). These nodes are all linear operations, and turning on motion blur for these nodes results in realistic blur for any parts of the image affected by animated transform parameters.

The Transform tab also contains nodes that perform changes in resolution, including the *Resize, Viewport, Window,* and *Zoom* nodes. These nodes are covered in more detail in "Controlling Image Resolution" on page 180.

The *Move2D* and *Move3D* nodes are the most flexible operators, since they include most of the parameters contained in the simple transform nodes. The processing requirements are the same whether you use a *Move2D* or a *Pan* node to move an image, since the *Pan* operation is simply a macro of the *Move2D* node. Given this choice, however, you'll find that using the *Move2D* node for multiple transformations is more efficient than simultaneously applying *Pan*, *Rotate* , and *Scale* nodes one after the other, due to the way Shake handles the internal computations.

Because Shake has an Infinite Workspace, effects are not clipped when they move in and out of frame with a second operation. For more information, see "Taking Advantage of the Infinite Workspace" on page 405.

Concatenation of Transformations

Many of the transform nodes concatenate, similar to the way color-correction nodes concatenate. Like color corrections, compatible transform nodes that are connected to one another are concatenated so that each operation is collapsed into a single calculation, optimizing both processing time and image quality. You can tell which transform nodes concatenate by the letter "C" in the upper-left corner of the icon in the Transform tab. In the screenshot below, you can see that the *CameraShake*, *CornerPin*, and *Move3D* nodes all concatenate, but the *Orient* node does not.

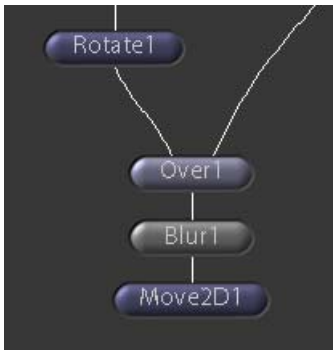
Thanks to concatenation, you can apply a *Move2D* node, a *Rotate* node, and a *CornerPin* node, and Shake resolves all three nodes by collapsing them into a single internal calculation, thus executing the end result in one operation. Not only is the render optimized, but the resulting image is of much higher quality (since the image is filtered one time instead of three times).

For example, if you apply a *Rotate* node to an image of the earth so that the lit side faces the sun, and then use a second *Rotate* node to orbit the earth around the sun, Shake determines the combined transformation and calculates it in one pass. This is important to understand because it means you spend (in this case) half of the processing time, and only filter the image once.

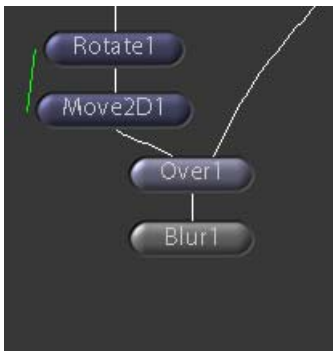
Note: To see the destructive effects of repetitive filtering, try panning an image 20 times in another compositing application.

It is to your advantage to employ transform concatenation whenever possible. As with color-correction nodes, transform nodes only concatenate when they're connected together, so arrange the operations in your node tree wisely. Any other type of node inserted between other transform nodes breaks the concatenation.

For example, you cannot apply a *Rotate* node, an *Over* node, a *Blur* node, and then a *Move2D* node and have the *Rotate* and the *Move2D* concatenate. Instead, it's best to make sure that the *Rotate* and *Move2D* nodes are placed together in the node tree. In many cases, a simple change like this results in dramatic improvements in the speed and quality of your images.



Transform nodes that aren't concatenated



Transform nodes that are concatenated

See "Creating Motion Blur in Shake" on page 778 for another good example of using concatenation to your advantage.

The following nodes concatenate with one another:

- *CameraShake*
- *CornerPin*
- *Move2D*
- *Move3D*
- *Pan*
- *Rotate*
- *Scale*
- *Shear*
- *Stabilize*

Making Concatenation Visible

You can set showConcatenationLinks to "enhanced" in the enhancedNodeView subtree of the Globals tab, then enable the enhanced Node View to see which nodes are currently concatenating. A green line connects transform nodes that concatenate, which makes it easy to see where the chain may be broken. For more information, see "Using the Enhanced Node View" on page 221.

Inverting Transformations

The *Move2D* and *CornerPin* nodes have an inverseTransform parameter. This aptly named parameter inverts the effect of the transformation, numerically. For example, a pan of 100 with inverseTransform activated becomes a pan of -100. The parameters themselves are not changed, just their effects on the composition.

In the case of *Move2D*, you can use inverseTransform to turn imported tracking data into stabilization data. In the case of *CornerPin*, you can set the four corners to map to four arbitrary points. The classic example of this is to pin an image onto the front of a television screen. Using the inverseTransform parameter, you could extract the original image that appears angled on the television screen, and remap it into a flat image. This technique is helpful for generating texture maps from photos for 3D renders.

Onscreen Controls

Most of the transform nodes have onscreen controls that let you interactively manipulate your images directly in the Viewer. These controls appear whenever that node's parameters are loaded.

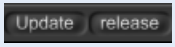
Note: Many of these controls are also shared by other nodes with similar functionality.

If an image moves offscreen—that is, beyond the boundaries of the Viewer—its transform controls remain visible in the Viewer area. When an image undergoes an extreme transformation and moves "offscreen," the best way to locate it is to zoom out in the Viewer until you see its transform controls.

Accelerating Viewer Interactivity

There are two fast ways you can speed up Shake's performance when using onscreen transform controls to perform transformations:

To quickly scrub through an animation, set the Update mode (located in the upper-right corner of the interface) to "release" then move the playhead. To select release from the Update mode list, click and hold the button labeled, "manual" or "always," then select "release." Because the image does not update until you release the mouse button, this setting lets you manipulate the controls freely without being slowed down by constant image processing.



You can also lower the Global interactiveScale parameter. Doing so dynamically lowers the resolution of images as they're being manipulated in Shake. This allows you to see the changes you're making to the image as you're making them, albeit at lower resolution. When you release the mouse button, the image returns to the current resolution of the project.

Viewer Shelf Controls

When you use an active node with onscreen controls, additional controls also appear in the Viewer shelf (a row of buttons that appears directly underneath the Viewer).

Viewer shelf of the *Move2D* node

The following table shows the common onscreen control buttons.

Button		Description
	Onscreen Controls–Show	Displays the onscreen controls. Click to toggle between Show and Hide mode.
	Onscreen Controls–Show on Release	Hides onscreen controls while you modify an image. To access this mode, click and hold the Onscreen Controls button, then choose this button from the pop-up menu, or right-click the Onscreen Controls button, then choose this option from the shortcut menu.
	Onscreen Controls–Hide	Turns off the onscreen controls. To access this mode, click and hold the Onscreen Controls button, then choose this button from the pop-up menu, or right-click the Onscreen Controls button, then choose this option from the shortcut menu.
	Autokey	When Autokey is on, a keyframe is automatically created each time an onscreen control is moved. To enable, click the button, or right-click it, then choose this option from the shortcut menu. To manually add a keyframe without moving the onscreen controls, click Autokey off and on.
	Delete Keyframe	Deletes the keyframe at the current frame. This button is useful because some nodes (such as *Move2D*) create multiple keyframes when you modify a single parameter. This button deletes the keyframes from all associated parameters at the current frame. To delete all keyframes for a parameter, right-click the Delete Keyframe button, then choose Delete All Keys from the shortcut menu.
	Lock Direction–Off	Allows dragging of onscreen controls in both the X and Y directions.
	Lock Direction to X	Allows dragging of onscreen controls in the X direction only. To enable, click and hold the Lock Direction button, then choose this button from the pop-up menu.
	Lock Direction to Y	Allows dragging of onscreen controls in the Y direction only. To enable, click and hold the Lock Direction button, then choose this button from the pop-up menu.
	Onscreen Color Control	Click this swatch to change the color of the onscreen controls.
	Path Display–Path and Keyframe	Displays the motion path and the keyframe positions in the Viewer. You can select and move the keyframes onscreen.

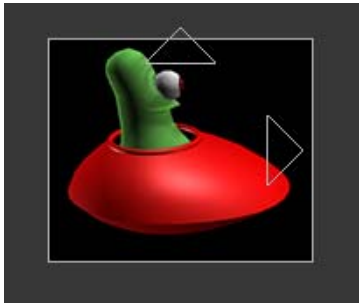
Button		Description
	Path Display–Keyframe	Displays only the keyframe positions in the Viewer. To access this mode, click and hold the Path Display button, then choose this button from the pop-up menu.
	Path Display–Hide	The motion path and keyframes are not displayed in the Viewer. To access this mode, click and hold the Path Display button, then choose this button from the pop-up menu.

Transform Controls

The most commonly used transform node is *Move2D*. The *Move2D* node combines the controls of the *Rotate*, *Pan*, and *Scale* nodes into a single operation. These nodes share a number of onscreen transform controls that enable you to manipulate images directly in the Viewer.

Pan

The pan controls provide two ways to move the image within the Viewer. Drag anywhere within the image bounding box to reposition the image freely. Drag the vertical arrowhead to restrict position changes to the yPan parameter. Drag the horizontal arrowhead to restrict position changes to the xPan parameter.

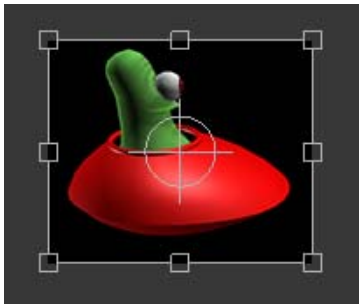


You can also press Q or P while dragging anywhere within the Viewer to pan an image without having to position the pointer directly over it.

Scale

Drag any of the corner controls to scale the entire image up or down, affecting the xScale and yScale parameters. Drag the top or bottom controls to scale the image vertically, affecting only the yScale parameter. Drag the left or right controls to scale the image horizontally, affecting only the xScale parameter.

Drag the center control to move the point about which scaling is performed, affecting the xCenter and yCenter parameters.

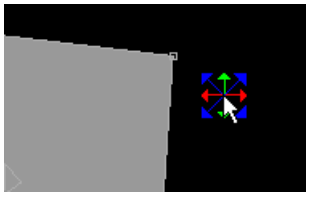


There is an additional method you can use to scale images in the Viewer.

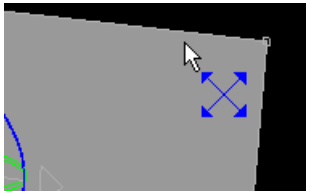
To scale an image in the Viewer without using the scale handles:

1. Select an image.
2. With the pointer positioned over the Viewer, press E or I.
3. When the dimension pointer appears, drag in the direction you want to scale the image.

   The colors in the dimension pointer correspond to the pan controls.

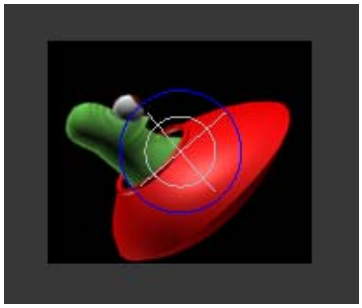


When you drag, the dimension pointer turns into an axis arrow, indicating the dimension in which you are scaling the image -- horizontal, vertical or diagonal.



Rotate

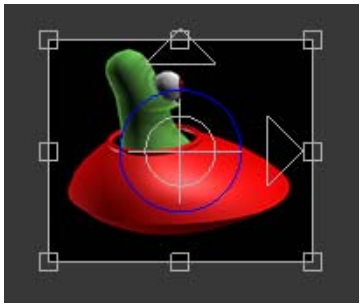
Drag the blue rotate control to rotate the image about the center point, affecting the angle parameter. Drag the white center control to move the center point itself, affecting the xCenter and yCenter parameters.



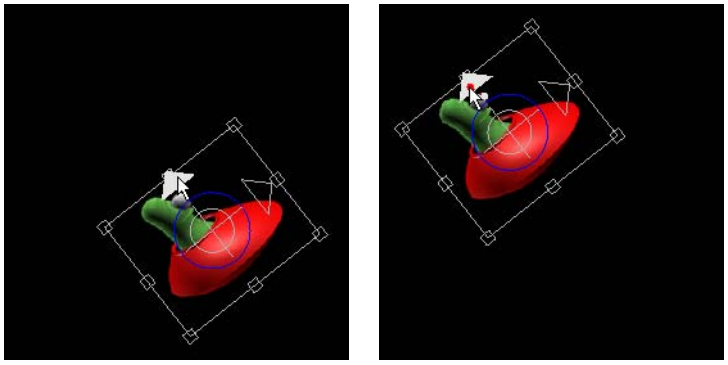
To rotate an image without positioning the pointer directly over it, press W or O, then drag in the Viewer. The dimension pointer appears, allowing you to to rotate the image in any direction.

Move2D

The *Move2D* node combines the onscreen transform controls for the *Pan, Scale*, and *Rotate* nodes into a single, all-purpose transformation node.

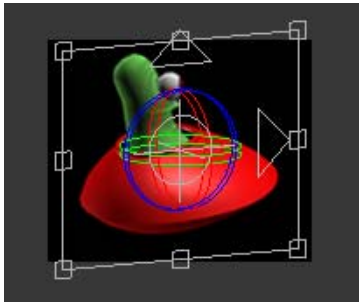


After an image is rotated with the *Move2D* node, the horizontal and vertical panning controls (arrowheads) lock movement to the new orientation of the image.



Move3D

Similar to the *Move2D* node, the *Move3D* node adds three colored dimensional angle controls to control xAngle (red), yAngle (green), and zAngle (blue) parameters. This lets you simulate 3D transformations with 2D images.



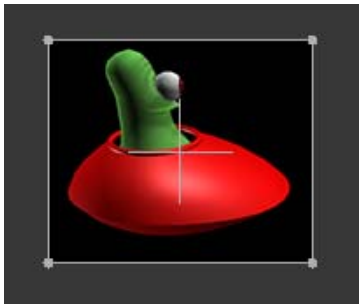
As with the *Move2D* node, when an image is panned with the *Move3D* node, the horizontal and vertical panning controls lock movement to the new orientation of the image, instead of the absolute orientation of the Viewer frame.

There are keyboard shortcuts (identical to those used within the *MultiPlane* node) that you can use to manipulate images without having to position the pointer directly over them. For more information, see "Transforming Individual Layers" on page 500.

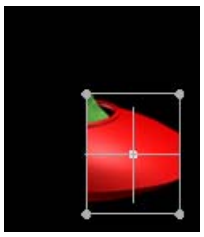
Note: Unlike similar controls available for layers connected to the *MultiPlane* node, the constrained pan controls do not move the image forward or back if the yAngle is tilted. To manipulate images within a more fully featured 3D environment, connect them to a *MultiPlane* node, and use the available controls. For more information, see Chapter 18, "Compositing With the MultiPlane Node," on page 485.

Crop

This onscreen transform control, available in the *Crop* node, lets you drag any corner to crop two sides of an image at once. Drag any outside edge to crop that edge by itself. This affects the cropLeft, cropBottom, cropRight, and cropTop parameters.

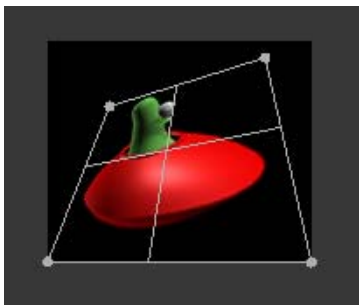


Drag the crosshairs at the center to move the entire crop box (simultaneously affecting all four crop parameters), while the image remains in place.



CornerPin

This onscreen transform control, available in the *CornerPin* node, lets you drag any corner to distort the image relative to that corner. Drag any edge to distort the image by moving both corners on that side. Drag the crosshairs to reposition the entire image.

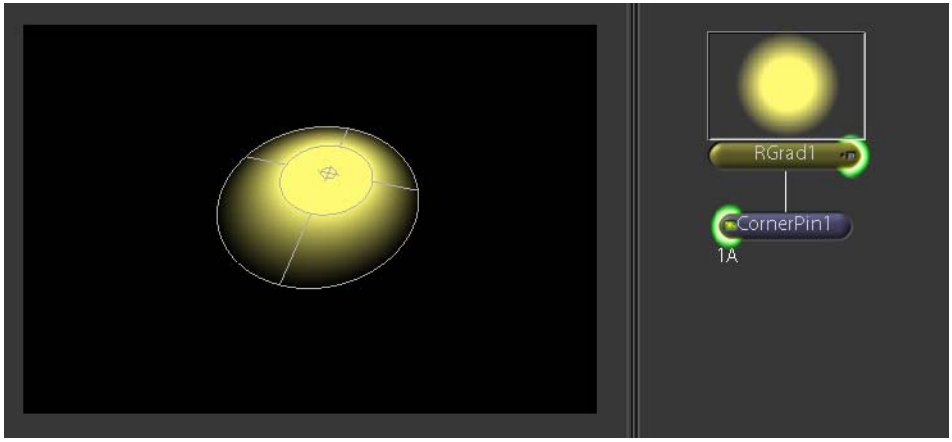


Onscreen Controls Across Multiple Transformations

If you apply multiple transformations to an image, all downstream onscreen controls are transformed along with the image. This lets you accurately visualize that control's effect on the image.

Note: This is also true for the onscreen controls of other nodes, like the *RGrad* and *Text* nodes.

In the following example, an *RGrad* node is connected to a *CornerPin* node. The *CornerPin* node is used to place the *RGrad* in perspective.

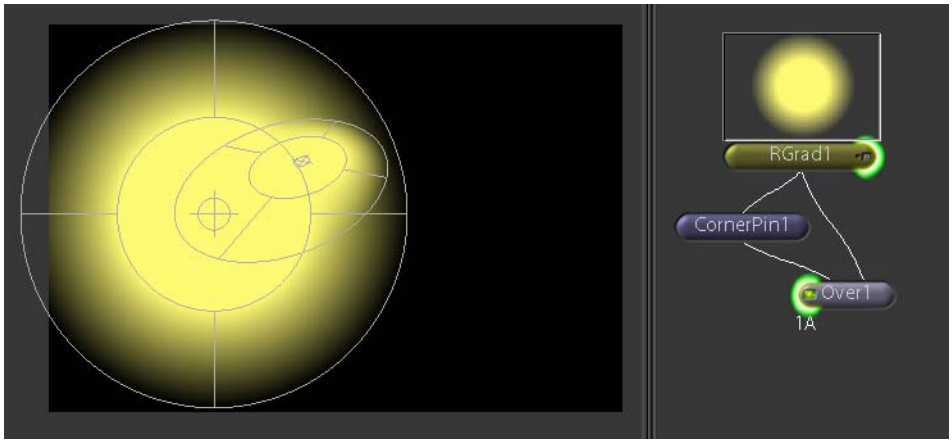


In the screenshot above, the image output by the *CornerPin* node is loaded in the Viewer, but the parameters of the *RGrad* node above it are loaded in the Parameters tab. As a result, the *RGrad* controls inherit the perspective shift from the *CornerPin* node.

Displaying Multiple Versions of the Same Control

If you create a node tree where several versions of the same node are visualized, transform controls are displayed for each copy of the node. This can be used to your advantage, to provide you with multiple perspectives of the same control, allowing for precise manipulation of the image.

In this example, the *CornerPin* node is composited over the original *RGrad* node.



As shown in the above image, manipulating the *RGrad1* node while viewing the *Over1* node results in the display of multiple controls. Changes made with one control modify both. To break this link, copy the original *RGrad* node and connect it to the new node.

Important: The *MatchMove* node is the only exception to this behavior. For nodes above the *MatchMove* node, the onscreen controls appear without transformation.

Scaling Images and Changing Resolution

There are two types of scaling functions:

- Functions that scale the size of the image in the frame, but do not change the actual resolution (*Scale*, *Move2D*)
- Functions that scale the size of the image in the frame, and also change the output resolution (*Resize*, *Zoom*, *Fit*)

Additionally, the *Crop*, *Window*, and *ViewPort* nodes can be used to change the image resolution by cutting into an image or expanding its borders.

Finally, the *SetDOD* node crops in an area of interest, called the Domain of Definition (DOD).

The following tables discuss the differences between the scaling functions.

Node	Changes Resolution	Changes Pixel Scale	Breaks Infinite Workspace	Changes Relative Aspect Ratio
Scale, *Move2D*	No	Yes	No	Yes
Crop	Yes	No	Yes	No
Window	Yes	No	Yes	No
ViewPort	Yes	No	No	No

Node	Changes Resolution	Changes Pixel Scale	Breaks Infinite Workspace	Changes Relative Aspect Ratio
SetDOD	No	No	Yes	No
Resize	Yes	Yes	No	Yes
Fit	Yes	Yes	Yes	No
Zoom	Yes	Yes	No	Yes

Node	Example	Example Parameters	Notes
(No node)		Resolution = 100 x 100 pixels	The unmodified image
Scale, Move2D		xScale = .5 yScale = .5	*Scale* is a subset of the *Move2D* function. There is no processing speed increase when using *Scale* instead of *Move2D*.
Scale, Move2D		xScale = 1.6 yScale = 1.6	
Crop		-33 , -33, 133, 133	When using the default parameters, 0, 0, width, height, *Crop* breaks the Infinite Workspace, and resets the color beyond the frame to black.
Crop		33, 33, 67, 67	

Node	Example	Example Parameters	Notes
Window		-33, -33, 166, 166	*Window* is identical to *Crop*, except that you specify the output resolution in the third and fourth parameters.
Window		33, 33, 34, 34	
ViewPort		33, 33, 67, 67	*ViewPort* is identical to *Crop*, except that it does not cut off the Infinite Workspace, and is therefore primarily used to set a resolution.
SetDOD		33, 33, 67, 67	Used to limit the calculation area of the node tree to within the DOD; considerably speeds up renders.
Resize		72, 48	
Fit		72, 48	*Fit* and *Resize* are identical, except that *Fit* preserves the pixel aspect ratio, padding the edges with black.
Zoom		.72, .48	*Zoom* and *Resize* are identical, except that *Zoom* gives a scaling factor and *Resize* gives a resolution for its parameters.

Creating Motion Blur in Shake

Motion blur can be applied to any animated transformation. Each transform node has its own motion blur settings, so you can fine-tune each node's effect individually. There is also a global set of motion blur parameters that adjusts or replaces the existing values, depending on the parameter.

Note: You can also use the global motion blur parameter to temporarily turn motion blur off.

You can control the quality of the blur using the motionBlur slider, found in the Parameters tab of most transform nodes.

- 0 disables motion blur for that operation.
- 0.5 = good quality.
- 1 = excellent quality.
- Higher values may be used for even better quality.

Opening the motionBlur subtree reveals two additional parameters that control the look of the resulting motion blur:

shutterTiming
This parameter controls the duration of the blur. By default, it goes from the current frame to halfway toward the next frame, which is a value of .5, equivalent to 180 degrees of camera shutter. Real-world film cameras are generally at 178 degrees.

shutterOffset
This parameter controls the frame at which the shutter opening is considered to start. This value is 0 by default, so it starts at the current frame and calculates the motion that occurs up until the next frame. When set to -1, the motion from the previous frame up to the current frame is calculated.

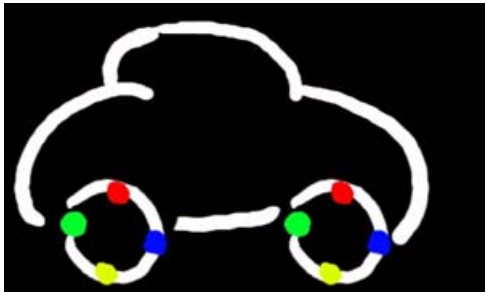
Motion Blur for Concatenated Nodes

When multiple transform nodes are concatenated, the highest blur settings from that collection of nodes is used for the overall calculation.

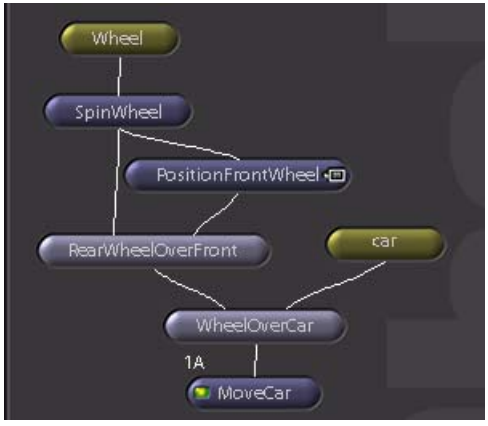
Multiple Elements With Independent Motion Blur

Sometimes you have to place multiple *Move2D* nodes somewhat counterintuitively in order to take advantage of transform node concatenation. For example, an *Over* node placed between two *Move2D* nodes breaks concatenation. To get around this, apply *Move2D* nodes one after the other, then use the *Over* node afterwards.

In the following example, two elements are composited together to simulate a car moving forward with spinning wheels: an image of a car body and a single graphic representing the wheel graphic. The wheel graphic is used twice, once for the back wheel and once for the front wheel. The colored dots on the wheels will illustrate the improper and proper arrangements of nodes necessary to produce realistic motion blur for this simulation.



In the following node tree, a *Move2D* node (renamed *SpinWheel*) is animated to rotate the wheel. The *PositionFrontWheel* node is a non-animated *Move2D* that pans a copy of *SpinWheel* to the front wheel position. The two wheel nodes are composited together (*RearWheelOverFront*), and the result is composited over the car body (*WheelsOverCar*). To animate the car forward, a *Move2D* node (*MoveCar1*) is applied, which pans the entire image to the right.

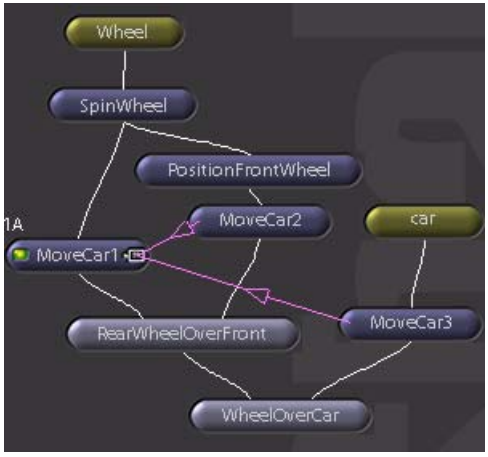


The result is inaccurate when motion blur is applied. This is because the *SpinWheel* node applies the blur for the turning wheel, and then three nodes later, the *MoveCar* node applies the blur to the already-blurred wheels. Instead of the individual paths of the color dots on the wheels, the result is a horizontal smear.

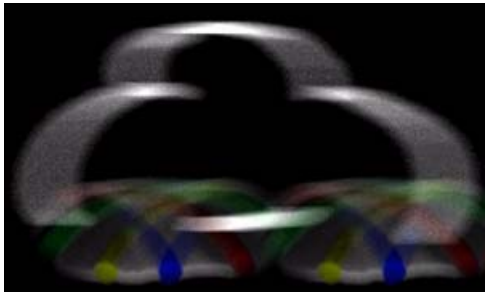
The following image shows the *WheelsOverCar* node immediately before the entire car is panned. The result is bad because the wheels are blurred without consideration of the forward momentum added in a later node.

To correct this, it is more efficient to apply three *Move2D* nodes to pan the three elements separately, and then composite them together. In the following tree, the *MoveCar1*, *MoveCar2*, and *MoveCar3* nodes are identical, and *MoveCar2* and *MoveCar3* are clones of *MoveCar1*.

Note: To create a cloned node, copy the node (press Command-C or Control-C) and clone the node using the Paste Linked command in the Node View shortcut menu (or press Command-Shift-V or Control-Shift-V).



Because the *SpinWheel* and *MoveCar1* nodes are transformations, these nodes concatenate. The *SpinWheel*, *PositionFrontWheel*, and *MoveCar2* nodes also concatenate. The result is three transformations, the same amount as the previous tree, but with an accurate blur on the wheels.



Applying Motion Blur to Non-Keyframed Motion

Ordinarily, motion blur only appears when you keyframe a parameter of a transform node in order to create Shake-generated motion. If you simply read in an image sequence with a moving subject within the frame, such as a man running along a road, Shake will not create any motion blur, since nothing is actually animated.

To create motion blur for such non-keyframed elements, you can apply a smear effect using a *Move2D* node. This is done by turning on the useReference parameter (at the bottom of the Parameters tab) and setting referenceFrame (in the useReference subtree) to "time" (the default).

The following example uses a previously rendered a clip of a swinging pendulum.

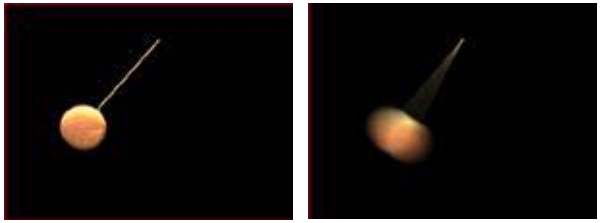
To add blur to a non-animated object with the useReference parameter:

1. Locate the center of rotation for the pendulum, then type the center values in the *Move2D* center field.
2. Approximate the rotation, then animate the angle to match the rotation.

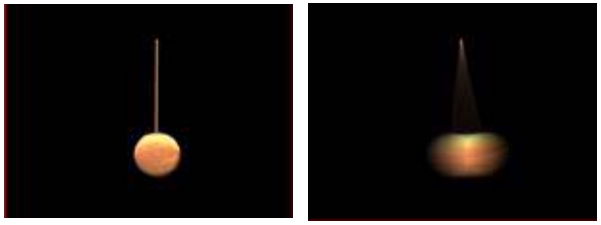
   In this example, the angle is -40 at frame 5 and 40 at frame 24.
3. Type the expression "time" into the referenceFrame value field (in the useReference subtree).
4. Set motionBlur to 1.

   When the animation is played back, the entire element swings out of frame due to the new animation.
5. Enable useReference (set it to 1).

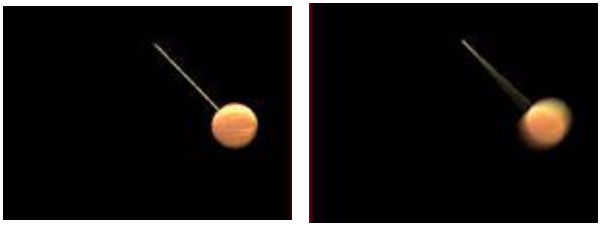
The element remains static, but the blur is still applied as if it were moving.



Frame 5



Frame 12



Frame 24

For a lesson on this subject, see Tutorial 4, "Working With Expressions," in the *Shake 4 Tutorials* book.

The AutoAlign Node

The *AutoAlign* node is unique among the various transform nodes in that it can take multiple image inputs and combine them into a single output, similar to a layering node. This node is designed to take two or three images or image sequences that overlap, and align them in different ways to create a single, seamless output image.

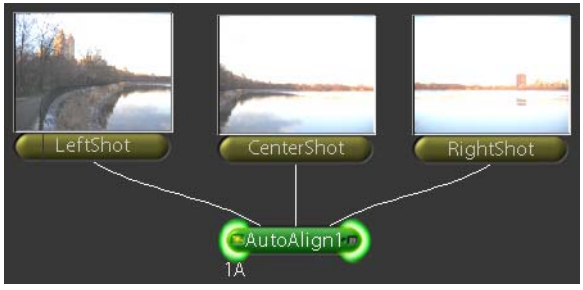
Unlike similar photographic tools, the *AutoAlign* node works with both stills and image sequences. As a result, you could film three side-by-side shots of an expanse of action, and later turn these into a single, extremely wide-angle background plate. Similarly, you can set a single still image to align with the same image features in a second shot with camera motion.

Stitching Images Together

When you connect two or three overlapping images or image sequences to the *AutoAlign* node, they're assembled into a single composite panoramic image. The input images can be either vertically or horizontally oriented—the *AutoAlign* node automatically determines the areas of overlap and aligns them appropriately.

Note: The order in which you connect images to the *AutoAlign* node doesn't matter; their position is automatically determined.

For example, if you connect the following three side-by-side shots of a skyline to the inputs of an *AutoAlign* node:



The overlapping areas are analyzed, matched, and the images are repositioned and manipulated to fit together.

As you can see in the above example, the resulting image may have an irregular border, depending on the amount and position of overlap, and the warping required to achieve alignment. If necessary, the border can be straightened with a *Crop* node.

Aligning Overlapping Images

You can also use this node to align two images that almost completely overlap. An obvious use of this is to align an image with something in the frame that you want to remove with a second "clean plate" image. After you've aligned the two images, you can use the aligned output image as a background plate for use in a paint or roto operation to remove the unwanted subject.

The primary advantage of the *AutoAlign* node in this scenario is that it allows you to precisely align a clean plate image to an image sequence in two difficult types of situations:

- If the clean plate image is from a different source, or has different framing or other characteristics that make it difficult to align using regular transformations
- If the target image is in motion, and alignment is difficult using regular trackers

For example, if you have the following shot of a man jumping across a bottomless chasm, with the camera moving dramatically to follow the action, it may be difficult to line up a still image clean plate.

Moving camera with unwanted safety rigging

Still image clean plate

Using the *AutoAlign* node to align the second image with the first, you can quickly match the clean plate still to the moving image sequence, and then output the newly aligned and animated clean plate for use by a paint or rotoscoping operation to paint out the safety line.

AutoAlign Limitations

Successful use of the *AutoAlign* node is highly dependent on the content of the input images, and results will vary widely from shot to shot. In general, moving images with a great deal of perspective and parallax can be difficult for the *AutoAlign* node to analyze properly, and may cause unexpected results.

Blending at the Seams

Two parameters help to automatically manage the seams between images that have been aligned with the *AutoAlign* node:

- *blendMode* automatically adjusts the overlapping border between two aligned images, intelligently selecting which portions of each image to retain for a seamless result.

blendMode set to none

blendMode set to smartBlend

- *matchIllum* automatically adjusts the brightness of each aligned image so that the exposure matches.

AutoAlign Image Requirements

Although the *AutoAlign* node is a very flexible tool, it produces the best results with material that is produced with this node in mind.

- There should be at least a 15-to-20-percent overlap between any two images for the *AutoAlign* node to work properly (the amount that's necessary may vary depending on the image).
- You may still be able to successfully line up images even when limited regions within each image don't line up, although this may affect the overall quality of the resulting alignment. Examples include clouds, waves, pedestrians, and birds. If you can arrange for it in advance, exact matches from simultaneously captured images are usually preferable.
- The warping performed by the *AutoAlign* node is designed to align multiple panoramic images that have been shot by *dollying* the camera laterally. Multiple images created by *panning* the camera from a single position may result in undesirable curvature in the final result.
- Interlaced images *must* be deinterlaced prior to analysis.
- When aligning image sequences, the *AutoAlign* node aligns the input frames in time according to each image sequence's timeShift parameter, as represented in the Time View. Offsetting an image sequence in the Time View offsets which frames will be aligned with one another.
- When aligning overlapping images, an object or person in one image that is not in the other may prevent successful analysis if it occupies too large an area.
- Images with an animated width or height cannot be aligned.

Image Stabilization Within the AutoAlign Node

You must select one of the input images to be used as the positional reference for *AutoAlign's* analysis, using the lockedPlate pop-up menu. If the shot you want to designate as the lockedPlate is not already steady, the other input images will be transformed to match the motion in this shot.

Example 1: A Procedure for Simple Image Alignment

This example covers how to set up a panoramic stitching operation using the *AutoAlign* node.

To automatically align two or three images:

1. In the Time View, adjust the offsets of the image sequences that you intend to use.
2. Create an *AutoAlign* node, and attach up to three images to it.

   The order in which you attach them doesn't matter.
3. Load the *AutoAlign* node's parameters into the Parameters tab.
4. If necessary, set the analysisRange to the frame range you want to analyze.
5. If you're analyzing images for the first time, set the mode parameter to precise.

If you're not satisfied with the result later in the operation, change the mode to robust and re-analyze the images.

6 Click the "analyze" button.

Shake steps through each frame in the image sequences and analyzes each set of aligned images at every frame in the designated analysisRange.

Note: The analysis can be interrupted at any time by pressing Esc.

7 Once the analysis has concluded, set clipLayer to All in order to expand the Region of Interest (ROI) to the overall width of the newly generated panorama.

8 Choose an input layer from the lockedPlate pop-up menu.

For purposes of creating a panorama, you should use an image that's either already stable, or one that has been locked down using the *SmoothCam* or *Stabilize* node.

Note: If the combined image is very wide, it may also be a good idea to set the centermost image as the lockedPlate.

9 To output the entire panorama as a single image, set ouputFrame to All.

10 If the seams between each aligned image are visible, set the blendMode parameter to smartBlend to make them invisible.

11 If the aligned images have different exposures, turn on matchIllum.

If you can still see differences in color or exposure, you can attach additional color-correction nodes between the upstream image and the *AutoAlign* node to make the necessary adjustments.

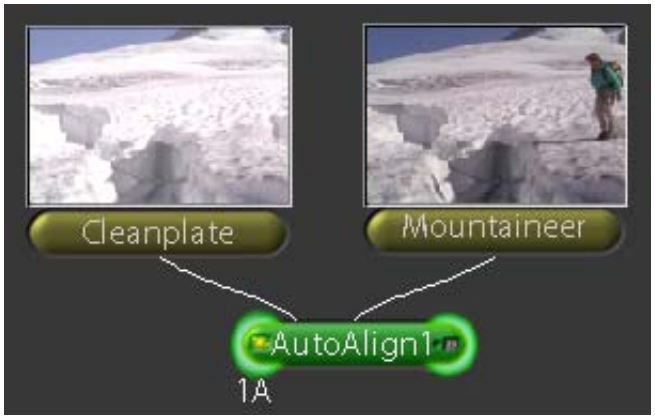
Example 2: Aligning a Clean Plate Image With a Moving Shot

This example shows how to align a clean plate still image with an image sequence in which the camera is moving.

To align a clean plate image with an overlapping image sequence:

1 Connect the clean plate image and the moving shot (in this example, the mountaineer image) to an *AutoAlign* node.

The order in which they are connected is not important.



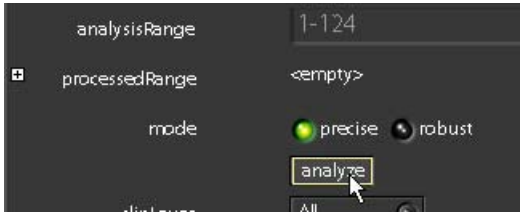
2 Use the clipLayer and lockedPlate pop-up menus to choose the input to which the clean plate image is connected.

In this example, you'll be choosing Input1. Leave the mode at the default setting of Precise.

3 Set the analysisRange to the number of frames you want to align.

This parameter defaults to the maximum number of frames within the longest image sequence that's connected to the *AutoAlign* node.

4 Click "analyze."



Shake begins the image analysis. The first frame may take longer than the other frames in the analysis. The analysis should speed up after the first frame. As the analysis is performed, the playhead steps through the Time Bar frame by frame, creating a keyframe at each analyzed frame.

Note: The analysis can be interrupted at any time by pressing Esc.

5 Once the analysis has concluded, changing blendMode to mix and scrubbing through the Time Bar shows you how well the resulting alignment works.

The Mix setting, in the case of two images that almost completely overlap, results in an a 50 percent blend of both images. In the above image, the ripples in the snow appear to align perfectly.

6 In order to use this result in a paint operation, set the following parameters:

a Set clipLayer to Input2, so that the moving shot defines the resolution of the output image.

b Set lockedPlate to Input2 as well, so that the clean plate image moves along with the background.

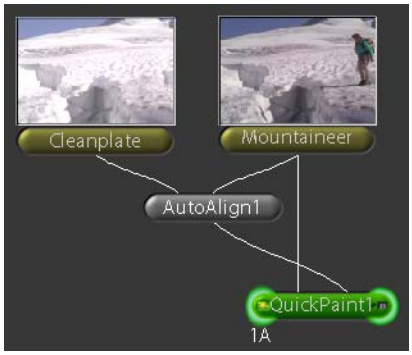
c Set outputFrame to Input1, and set blendMode to none, so that the only image that is output is the newly animated and aligned clean plate image.

The resulting output image is a transformed version of the clean plate image that matches the position of matching overlapping features in the mountaineer image.

Original clean plate image

Autoaligned clean plate image

7 The auto-aligned clean plate can now be used in paint or rotoscoping operations to remove unwanted rigging from the actor in the mountaineer shot. For example, you can connect the original *Mountaineer* image to the first input of a *QuickPaint* node, and the output from the *AutoAlign* node to the second input.



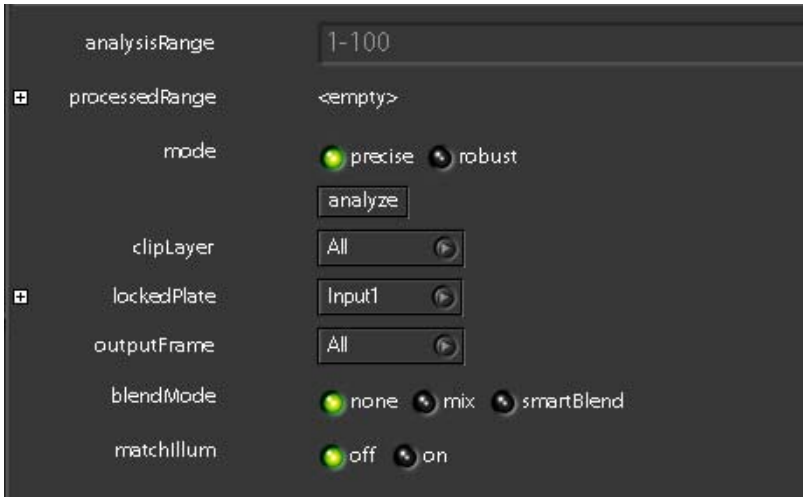
8 With this setup, you can use the Reveal Brush to paint out the rigging against the backdrop of the clean plate.

Original Image

Reveal Brush exposing auto-aligned image

AutoAlign Parameters

The *AutoAlign* node displays the following controls in the Parameters tab:



analyze

Click this button to perform the analysis that is the first step in aligning two or three images. This analysis creates a preprocessed data set that is used to perform the actual alignment. Images only have to be analyzed once, and the resulting transform data is stored inside the Shake script. The other parameters in the node act upon this transformation data—changing the parameters has no effect on the analysis data.

If, at any time, you're not satisfied with the results of the analysis, you can change the mode and re-analyze the analysisRange.

analysisRange

The range of frames you want to process. By default this parameter inherits the maximum number of frames in the *FileIn* node at the top of the tree.

processedRange

A non-editable display field that shows the number of frames that have already been analyzed. When you first create an *AutoAlign* node, this parameter reads "<empty>."

Opening the processedRange subtree reveals the following subparameters:

- *confidence:* The confidence value indicates the level of matching accuracy that resulted from the analysis. Each frame's confidence is restricted to one of three possible values. You can load the confidence parameter into the Curve Editor to quickly find problem areas in the analysis. The values are:
  - *1:* Indicates an analysis in which Shake has high confidence. A keyframe has been generated at these frames.

- *0.5:* Indicates an uncertain analysis. Despite the uncertainty, Shake has generated a keyframe at these frames.
- *0:* Indicates that Shake has no confidence in the analysis. No keyframe has been generated at these frames.

- *mMatrix, nMatrix:* These parameters contain the data accumulated by clicking the "analyze" button.

mode
Two options—precise and robust—let you change the method used to perform the analysis. In general, set this mode to precise the first time you analyze the input images. If the results are not satisfactory, change the mode to robust, then re-analyze.

clipLayer
Lets you to choose which image defines the resolution of the output image—Input1, Input2, Input3, or All, which sets the resolution to the final size of the aligned and merged images.

lockedPlate
This parameter lets you choose which input image to use as a positional reference. The other two input images will be stabilized and aligned to match the position of the lockedPlate. The lockedPlate also affects how the other two images are warped to match the overall perspective of the final image. If the final image is going to be a wide-angle shot, it may be a good idea to set the center image as the lockedPlate.

- *lockedPlateOffsetX, lockedPlateOffsetY:* These subparameters of lockedPlate contain the transform keyframes used to align the plates. Changing the lockedPlate also changes the keyframe set displayed by these parameters.

outputFrame
Lets you choose which image is output by the *AutoAlign* node. The options are identical to those of the clipLayer parameter.

blendMode
This parameter provides options for automatically blending the seams where the input images meet. There are three choices:

- *none:* The seams are not blended.
- *mix:* Both sides of each seam are simply feathered together.
- *smartBlend:* The highest quality mode. The best pixels from either side of each seam are used to patch together a seamless image.

matchIllum
Turning this parameter on automatically adjusts the brightness of the input images so that the lighting is consistent across the entire output image.

Note: If necessary, you can preprocess images connected to the *AutoAlign* node with other color-correction nodes to even out differences in gamma and contrast that aren't addressed by the matchIllum parameter.

The Transform Nodes

In addition to the *AutoAlign* node, Shake features numerous other transform nodes. The following section includes information on the transform nodes, which are located in the Transform Tool tab.

For information on the transform nodes that are used for tracking and stabilization (*MatchMove*, *Stabilize*, and *Tracker*), see Chapter 25, "Image Tracking, Stabilization, and SmoothCam," on page 717.

For information on the transform functions that affect image resolution (*Fit*, *Resize*, and *Zoom*), see "Nodes That Affect Image Resolution" on page 183.

For information on cropping functions (*AddBorders*, *Crop*, *Viewport*, and *Window*), see "Cropping Functions" on page 186.

CameraShake

The *CameraShake* node is a macro that applies noise functions to the pan values of *Move2D*. Since it is a good example of how to use noise, look at the actual macro parameters. Usually, a *Scale* is appended following a *CameraShake* node to make up for the black edges that appear. Because of the concatenation of transformations, this does not double-filter your image, so speed and quality are maintained.

Parameters

This node displays the following controls in the Parameters tab:

xFrequency, yFrequency
The x and y frequency of the shake. Higher numbers create faster jitter.

xAmplitude, yAmplitude
The maximum amount of pixels the element is moved by a single camera shake.

seed
When Shake generates a random pattern of values, you need to make sure for purposes of compositing that you can recreate the same random pattern a second time. In other words, you want to be able to create different random patterns, evaluating each one until you find the one that works best, but then you don't want that particular random pattern to change again.

Shake uses the seed value as the basis for generating a random value. Using the same seed value results in the same random value being generated, so that your image doesn't change every time you re-render. Use a single value for a static result, or use the keyword "time" to create a pattern of random values that changes over time.

For more information on using random numbers in expressions, see "Reference Tables for Functions, Variables, and Expressions" on page 941.

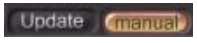
motionBlur
A Motion Blur quality level of 0. 0 produces no blur, whereas 1 represents standard filtering. For more speed, use a value less than 1. This value is multiplied by the global motionBlur parameter.

- *shutterTiming:* A subparameter of motionBlur used to specify shutter length. 0 is no blur, whereas 1 represents a whole frame of blur. Note that standard camera blur is 180 degrees, or a value of .5. This value is multiplied by the global shutterTiming parameter.
- *shutterOffset*: A subparameter of motionBlur. This is the offset from the current frame at which the blur is calculated. Default is 0; previous frames are less than 0.

CornerPin

The *CornerPin* node can be used to push the four corners of an image into four different positions, or to extract four positions and place them into the corners. The first use is ideal for positioning an image in an onscreen television, for example. The second mode is handy for extracting texture maps, among other things. The four coordinate pairs start at the lower-left corner of the image (x0, y0), and work around in a counterclockwise direction to arrive at the upper-left corner of the image (x3, y3).

To perform an "unpin" (push four points on the image to the four corners of the image), switch the Update mode to "manual," position the four points, enable inverseTransform, and click Update.



If the result is too blurred, lower the anti-aliasing value. For information on four-point tracking, see Chapter 25, "Image Tracking, Stabilization, and SmoothCam," on page 717.

Note: You can also use the *Move3D* node for perspective shift.

Parameters

This node displays the following controls in the Parameters tab:

x0, y0, x1, y1, x2, y2, x3, y3
Eight parameters controlling the position of each corner in the corner-pin operation.

xFilter, yFilter
A pop-up menu that lets you pick which method Shake uses to transform the image. For more information, see "Filters Within Transform Nodes" on page 862.

inverseTransform
A button that inverts the transform. In this case, it puts the four corners into the four coordinates (0, or pinning), or pulls the four coordinates to the corners (1, or unpinning).

antialiasing
Individual antialiasing. A value of 0 brings out more clarity.

motionBlur
Motion Blur quality level. 0 is no blur, whereas 1 represents standard filtering. For more speed, use less than 1. This value is multiplied by the global motionBlur parameter.

- *shutterTiming:* A subparameter of motionBlur used to specify shutter length. 0 is no blur, whereas 1 represents a whole frame of blur. Note that standard camera blur is 180 degrees, or a value of .5. This value is multiplied by the global shutterTiming parameter.
- *shutterOffset:* A subparameter of motionBlur representing the offset from the current frame at which the blur is calculated. Default is 0; previous frames are less than 0.

useReference
This is to be used to apply motion blur to previously animated elements. See Tutorial 4, "Working With Expressions," in the *Shake 4 Tutorials*.

Crop

For information on the *Crop* node, see "*Crop*" on page 186.

Fit

For information on the *Fit* node, see "*Fit*" on page 183.

Flip

The *Flip* node flips an image upside down. You can also use the *Scale* or *Move2D* node to invert the image by setting the yScale value to -1. There are no parameters for the *Flip* node.

Note: *Flip* (or a yScale of -1) forces the buffering of the image into memory.

Flop

The *Flop* node flops the image left and right. Unlike the *Flip* node, this does not buffer the image into memory. You can also use the *Scale* or *Move2D* node to invert the image by setting the yScale value to -1. There are no parameters for the *Flop* node.

MatchMove

For information on the *MatchMove* node, see "*MatchMove*" on page 740.

Move2D

The *Move2D* node combines many of the other transform nodes, including *Pan*, *Scale*, *Shear*, and *Rotate*. The xCenter and yCenter parameters apply to both scaling and rotation centers. If you need different centers, you can append a second node (*Move2D*, *Rotate*, or *Scale*) and switch scaling or rotation to that node. This does not cost you any process time, because Shake concatenates neighboring transforms into one big transform. Therefore, you lose neither quality nor calculation time.

Also, Shake's Infinite Workspace comes into play when you have two transforms together. (For example, one transform rotates the image, and the second transform rotates the image back, without clipping the corners.) However, in terms of workflow, when you pan around small elements that are later composited on larger resolution backgrounds, you do not have to crop your small elements out to create a larger space.

Parameters

This node displays the following controls in the Parameters tab:

xPan, yPan
These values let you move the image horizontally and vertically. 0, 0 represents the center of the frame, and the xPan and yPan sliders allow adjustments up to plus or minus the total width and height of the image frame.

angle
Lets you rotate the image around its center point. The slider allows adjustments plus or minus 360 degrees, but you can enter any value you want into the numeric field.

aspectRatio
This parameter inherits the current value of the defaultAspect global parameter. If you're working on a nonsquare pixel or anamorphic image, it is important that this value is set correctly to avoid unwanted distortion in the image.

xScale, yScale
These parameters let you change the horizontal and vertical scale of the image. By default, the yScale parameter is locked to the xScale parameter. The sliders let you adjust the scale of the image from 0 to 4 times the current size, but you can enter any value into the numeric field.

Note: Entering a negative value into the xScale or yScale numeric field reverses the image along that axis.

xShear, yShear
These parameters let you shear the image horizontally and vertically. The sliders let you adjust the shearing of the image anywhere between 0 and 1 (1 creates 45 degrees of shearing), but you can enter any value you want into the numeric field.

xCenter, yCenter
These parameters let you move the horizontal and vertical position of the center point around which all transformations occur. For example, moving the center point to the right changes the point about which the image rotates when you adjust the angle parameter.

xFilter, yFilter
Lets you pick which method Shake uses to transform the image. For more information, see "Filters Within Transform Nodes" on page 862.

transformationOrder
The order the transform is executed, with:
- t = translate
- r = rotate
- s = scale
- x = shear

By default, this is set to "trsx."

inverseTransform
Inverts the transform. This can be used to quickly convert tracking data into stabilization data.

motionBlur
Motion Blur quality level. 0 is no blur, whereas 1 represents standard filtering. For more speed, use less than 1. This value is multiplied by the global motionBlur parameter.
- *shutterTiming:* A subparameter of motionBlur used to specify shutter length. 0 is no blur, whereas 1 represents a whole frame of blur. Note that standard camera blur is 180 degrees, or a value of .5. This value is multiplied by the global shutterTiming parameter.
- *shutterOffset:* A subparameter of motionBlur representing the offset from the current frame at which the blur is calculated. Default is 0; previous frames are less than 0.

useReference
Applies the transform to the image or doesn't. If it doesn't, and you have animated values, useReference applies a motion blur to the image, but does not actually move it. This is good for adding blur to plates. See below for an example.

- 0 = Move image
- 1 = Smear-mode; image is not moved
- *referenceFrame:* A subparameter of useReference used for image stabilization. By default, this parameter is set to "time," that is, in reference to itself. However, if inverseTransform is set to 1, you are stabilizing, with the assumption that any animation you have applied matches up to the source animation. Setting the reference frame locks the movement in to a specific frame.

Note: For an example of using the useReference parameter, see "Applying Motion Blur to Non-Keyframed Motion" on page 781.

Move3D

The *Move3D* node allows you to create perspective changes by rotating and visually moving the image in depth. It is similar in behavior to the *Move2D* node.

Some *Move3D* parameters, such as zPan, have no effect unless the fieldOfView parameter is set to a value greater than 0. The angle of the Z axis is controlled, as in *Move2D*, by the angle parameter. There is no shearing, but you can rotate the image in X or Y, and keep the fieldOfView set to 0 to get orthogonal shearing effects.

Parameters

This node displays the following controls in the Parameters tab:

xPan, yPan, zPan
These values let you move the image horizontally, vertically, and in and out along the Z axis. 0, 0 represents the center of the frame, and the xPan and yPan sliders allow adjustments up to plus or minus the total width and height of the image frame.

zPan has no effect unless the fieldOfView value is set to greater than 0.

xAngle, yAngle, zAngle
Lets you rotate the image around its center point, along any axis in three-dimensional space. The slider allows adjustments plus or minus 360 degrees, but you can enter any value you want into the numeric field.

aspectRatio
This parameter inherits the current value of the defaultAspect global parameter. If you're working on a nonsquare pixel or anamorphic image, it is important that this value is set correctly to avoid unwanted distortion in the image.

xScale, yScale, zScale
These parameters let you change the scale of the image along any axis. By default, the yScale parameter is locked to the xScale parameter. The sliders let you adjust the scale of the image from 0 to 4 times the current size, but you can enter any value into the numeric field.

Note: Entering a negative value into the xScale, yScale, or zScale numeric field reverses the image along that axis.

xCenter, yCenter, zCenter
These parameters let you change the position of the center point around which all transformations occur. For example, moving the center point to the right changes the point about which the image rotates when you adjust the angle parameter.

fieldOfView
Designates the vertical field of view, and affects the appearance of perspective shift. When this is 0, the view is considered to be orthogonal, with increasing perspective changes when you combine zPan, angleX, and angleY with a higher fieldOfView.

xFilter, yFilter
Lets you pick which method Shake uses to transform the image. For more information, see "Filters Within Transform Nodes" on page 862.

transformationOrder
The order the transform is executed, with
- t = translate
- r = rotate
- s = scale

By default, this is set to "trs."

Note: If you are trying to transfer camera setups from 3D applications, this is the order transforms are pushed onto the matrix. The r transform can be replaced with yzx (representing the three dimensions used by the 3D camera) and the resulting transformationOrder would read tyzxs.

inverseTransform
Inverts the transform. This can be used to quickly convert tracking data into stabilization data.

motionBlur
Motion Blur quality level. 0 is no blur, whereas 1 represents standard filtering. For more speed, use less than 1. This value is multiplied by the global motionBlur parameter.

- *shutterTiming:* A subparameter of motionBlur used to specify shutter length. 0 is no blur, whereas 1 represents a whole frame of blur. Note that standard camera blur is 180 degrees, or a value of .5. This value is multiplied by the global shutterTiming parameter.
- *shutterOffset:* A subparameter of motionBlur representing the offset from the current frame at which the blur is calculated. Default is 0; previous frames are less than 0.

useReference

Applies the transform to the image or doesn't. If it doesn't, and you have animated values, useReference applies a motion blur to the image, but does not actually move it. This is good for adding blur to plates. 0 = Move image. 1 = Smear-mode (image is not moved).

- referenceFrame : A subparameter of useReference used for image stabilization. By default, this parameter is set to "time," that is, in reference to itself. However, if inverseTransform is set to 1, you are stabilizing, with the assumption that any animation you have applied matches up to the source animation. Setting the reference frame locks the movement in to a specific frame.

Orient

The *Orient* node rotates the image by 90-degree increments, resizing the image frame if necessary. (For example, for one turn on a 300 x 600 frame, it rotates it 90 degrees, and makes a 600 x 300 frame.) You can also apply a *Flip* node and *Flop* node to the image. To rotate in degrees, use a *Rotate* node or a *Move2D* node.

Parameters

This node displays the following controls in the Parameters tab:

nTurn

Number of 90-degree increments the image is rotated, for example:

- 0 = no rotation
- 1 = 90 degrees turn counterclockwise
- 2 = 180 degrees turn counterclockwise
- 3 = 270 degrees turn counterclockwise
- 4 = no rotation

flop

Flops the image left and right around the Y axis.

- 0 = no flop
- 1 = flop

flip

Flips the image up and down around the X axis.

- 0 = no flip
- 1 = flip

Pan

The *Pan* node pans the image with subpixel precision. To wrap an image around the frame (for example, anything that moves off the right edge of the frame reappears on the left), use the *Scroll* node.

Parameters

This node displays the following controls in the Parameters tab:

xPan, yPan

These values let you move the image horizontally and vertically. 0, 0 represents the center of the frame, and the xPan and yPan sliders allow adjustments up to plus or minus the total width and height of the image frame.

motionBlur

Motion Blur quality level. 0 is no blur, whereas 1 represents standard filtering. For more speed, use less than 1. This value is multiplied by the global motionBlur parameter.

- *shutterTiming:* A subparameter of motionBlur used to specify shutter length. 0 is no blur, whereas 1 represents a whole frame of blur. Note that standard camera blur is 180 degrees, or a value of .5. This value is multiplied by the global shutterTiming parameter.
- *shutterOffset:* A subparameter of motionBlur representing the offset from the current frame at which the blur is calculated. Default is 0; previous frames are less than 0.

Resize

For information on the *Resize* node, see "*Resize*" on page 184.

Rotate

The *Rotate* node rotates the image with subpixel precision.

Parameters

This node displays the following controls in the Parameters tab:

angle

Lets you rotate the image around its center point. The slider allows adjustments plus or minus 360 degrees, but you can enter any value you want into the numeric field.

aspectRatio

This parameter inherits the current value of the defaultAspect global parameter. If you're working on a nonsquare pixel or anamorphic image, it is important that this value is set correctly to avoid unwanted distortion in the image.

xCenter, yCenter

These parameters let you move the horizontal and vertical position of the center point around which all transformations occur. For example, moving the center point to the right changes the point about which the image rotates when you adjust the angle parameter.

motionBlur
Motion Blur quality level. 0 is no blur, whereas 1 represents standard filtering. For more speed, use less than 1. This value is multiplied by the global motionBlur parameter.

- *shutterTiming:* A subparameter of motionBlur used to specify shutter length. 0 is no blur, whereas 1 represents a whole frame of blur. Note that standard camera blur is 180 degrees, or a value of .5. This value is multiplied by the global shutterTiming parameter.
- *shutterOffset:* A subparameter of motionBlur representing the offset from the current frame at which the blur is calculated. Default is 0; previous frames are less than 0.

Scale

The *Scale* node scales the image with subpixel precision. To rotate and scale an image, you probably want to use the two independent nodes, *Rotate* and *Scale*, rather than a *Move2D* node, because you can control the centers independently. Unlike the *Resize* node, *Scale* does not change the image resolution.

If you scale by a negative number on the Y axis, you buffer the image. You can also use the *Flip* and *Flop* nodes to invert your image.

Parameters

This node displays the following controls in the Parameters tab:

xScale, yScale
These parameters let you change the horizontal and vertical scale of the image. By default, the yScale parameter is locked to the xScale parameter. The sliders let you adjust the scale of the image from 0 to 4 times the current size, but you can enter any value into the numeric field.

Note: Entering a negative value into the xScale or yScale numeric field reverses the image along that axis.

aspectRatio
This parameter inherits the current value of the defaultAspect global parameter. If you're working on a nonsquare pixel or anamorphic image, it is important that this value is set correctly to avoid unwanted distortion in the image.

xCenter, yCenter
These parameters let you move the horizontal and vertical position of the center point around which all transformations occur. For example, moving the center point to the right changes the point about which the image rotates when you adjust the angle parameter.

motionBlur
Motion Blur quality level. 0 is no blur, whereas 1 represents standard filtering. For more speed, use less than 1. This value is multiplied by the global motionBlur parameter.

- *shutterTiming:* A subparameter of motionBlur used to specify shutter length. 0 is no blur, whereas 1 represents a whole frame of blur. Note that standard camera blur is 180 degrees, or a value of .5. This value is multiplied by the global shutterTiming parameter.
- *shutterOffset:* A subparameter of motionBlur representing the offset from the current frame at which the blur is calculated. Default is 0; previous frames are less than 0.

Scroll

The *Scroll* node is similar to the *Pan* node, except that the image wraps around the frame, and reappears on the opposite side. Because there is no way to track a pixel, there is no motion blur available in the *Scroll* node.

Parameters

This node displays the following controls in the Parameters tab:

xScroll, yScroll
The X and Y panning values.

motionBlur
Motion Blur quality level. 0 is no blur, whereas 1 represents standard filtering. For more speed, use less than 1. This value is multiplied by the global motionBlur parameter.

- *shutterTiming:* A subparameter of motionBlur used to specify shutter length. 0 is no blur, whereas 1 represents a whole frame of blur. Note that standard camera blur is 180 degrees, or a value of .5. This value is multiplied by the global shutterTiming parameter.
- *shutterOffset:* A subparameter of motionBlur representing the offset from the current frame at which the blur is calculated. Default is 0; previous frames are less than 0.

SetDOD

The *SetDOD* node limits the active area, or Domain of Definition (DOD), to a limited window. If you have a large element with only a small portion of non-black pixels, you can apply a *SetDOD* node to isolate the area and to speed all image processes.

Shake automatically assigns a DOD to an .iff image. For example, if you scale down an image, Shake limits the DOD to the scaled area. This remains in effect for later composites, but does not affect the current script.

You can procedurally access the DOD of an image with the following variables:

- *dod[0]* Left edge of DOD
- *dod[1]* Bottom edge of DOD
- *dod[2]* Right edge of DOD
- *dod[3]* Top edge of DOD

For example, calling *FileIn1.dod[0]* returns the minimum X value of the *FileIn1* node's DOD.

Parameters

This node displays the following controls in the Parameters tab:

left, right, bottom, top

As their names imply, these parameters let you set the outer boundaries of the DOD.

For more information, see "The Domain of Definition (DOD)" on page 82.

Shear

The *Shear* node skews the image left and right, or up and down. Motion blur can also be applied.

Parameters

This node displays the following controls in the Parameters tab:

xShear, yShear

These parameters let you shear the image horizontally and vertically. The sliders let you adjust the shearing of the image anywhere between 0 and 1 (1 creates 45 degrees of shearing), but you can enter any value you want into the numeric field.

xCenter, yCenter

These parameters let you move the horizontal and vertical position of the center point around which all transformations occur. For example, moving the center point to the right changes the point about which the image rotates when you adjust the angle parameter.

motionBlur

Motion Blur quality level. 0 is no blur, whereas 1 represents standard filtering. For more speed, use less than 1. This value is multiplied by the global motionBlur parameter.

- *shutterTiming:* A subparameter of motionBlur used to specify shutter length. 0 is no blur, whereas 1 represents a whole frame of blur. Note that standard camera blur is 180 degrees, or a value of .5. This value is multiplied by the global shutterTiming parameter.
- *shutterOffset:* A subparameter of motionBlur representing the offset from the current frame at which the blur is calculated. Default is 0; previous frames are less than 0.

Stabilize, Tracker

For information on the *Stabilize* and *Tracker* nodes, see "Tracking Nodes" on page 740.

Viewport, Window

For information on the *Viewport* and *Window* nodes, see "Cropping Functions" on page 186.

Zoom

For information on the *Zoom* node, see "*Zoom*" on page 185.

Warping and Morphing Images

27

Shake provides powerful warping and morphing tools that are flexible enough to use for a wide variety of compositing tasks, from creating or correcting lens distortions, to morphing one subject into another.

About Warps

Shake's various linear transformation nodes, such as *Move2D, Move3D, Rotate,* and *Scale*, operate on entire images so that each pixel is moved, rotated, or scaled by the same amount. Even the *CornerPin* node applies the same transformation to every pixel—an amount defined by the location of each of the four corners. Shake's warp nodes, including *iDisplace, Turbulate*, and *Twirl*, and the *Warper* and *Morpher* nodes, differ from linear transformations in that each pixel's transformation is calculated individually. Depending on the type and settings of the warp you're creating, each pixel of an image can be moved independently of its neighbor. For this reason, warps can be referred to as "nonlinear transformations."

The Basic Warp Nodes

Shake's basic warp nodes are good for deforming one image using another as a guide, making rippling patterns, randomizing the texture of an image, and other similar effects. These nodes are quite powerful, but their main strength is in making wholesale image deformations while requiring a minimum of manual control, using either mathematical expressions or secondary images to define the warping being done. This section discusses the basic warp nodes, located in the Warp Tool tab.

DisplaceX

The *DisplaceX* node is a general purpose warping tool that is similar to *WarpX*, except you can access a second warping image to control the distribution of a warp. Any formula can be entered in the x and y fields for custom warps.

You can also create multiline expressions with this node.

Parameters

This node displays the following controls in the Parameters tab:

overSampling

The actual number of samples per pixel equals this number squared. For better antialiasing, increase the number in the value field.

xExpr, yExpr

The expression for where the pixel information is pulled. Expressions of x and y return the same image. Expressions of x+5, y+5 pull the color from the pixel five units up and to the right of the current pixel. You can access the values of the warping image (the second one) with r, g, b, a, and z.

Here are some example expressions:

```
"x*r"
"x+((r-.5)*30*r)"
"x+cos(y/140)*70*g"
"x+r*r*cos(x*y/100)*100"
```

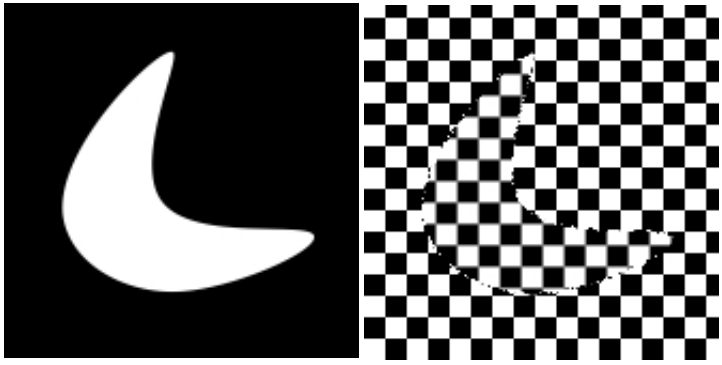
xDelta, yDelta

Sets the maximum distance that any pixel is expected to move, but doesn't actually move it. Any given pixel in a displaced image may be affected by any pixel within the Delta distance. This means it must consider a much greater amount of pixels that may possibly affect the currently rendered pixel. This is bad. However, if you set a Delta value that is too small, you get errors if your expression tells the pixel to move beyond that limit. Therefore, you'll want to do some testing to balance between speed with errors, or accuracy with drastically slower renders. Our advice: Start small and increase the size until the errors disappear.

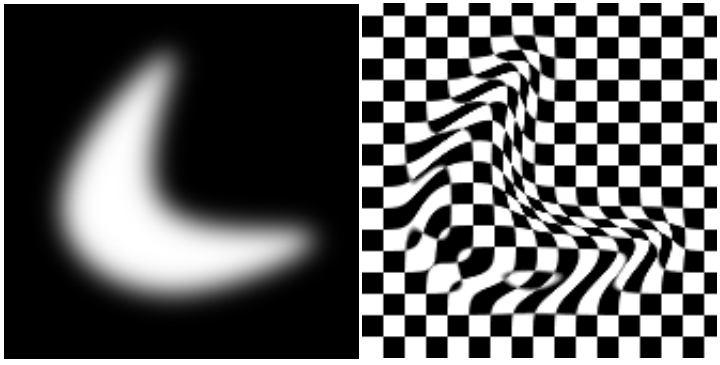
IDisplace

The *IDisplace* node is a hard-wired version of *DisplaceX* intended to warp an image based on the intensity of a second image. The formula used is *x-(a*xScale)* and *y-(a*yScale)*.

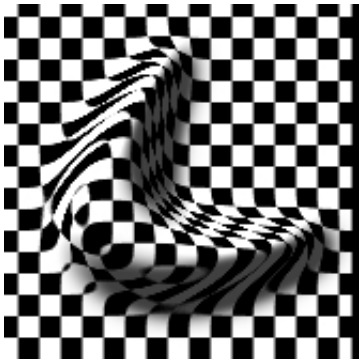
The following image is a checkerboard warped with a *QuickShape* node. Because the shape is black and white, with little gray, it is difficult to make out the distortion in the checkerboard.



As the following image demonstrates, it is often a good idea to insert a blur between a high-contrast distortion image and the *IDisplace* node.



This technique combines well with the Relief macro in the "Cookbook" chapter of this manual.



Parameters

This node displays the following controls in the Parameters tab:

xScale, yScale
The number of pixels that the foreground image is offset by the background image.

xDOffset, yDOffset
A panning factor applied to the image. Intensity is usually 0 to 1; 1 is 100 percent of the x/yScale factor.

xChannel, yChannel
The channel from the background image that is used to distort the foreground image.

xDelta, yDelta
The anticipated distance the pixels will move. If this value is too high, calculations slow down. If it is too low, black holes will appear in the image.

LensWarp

This node lets you make subtle or large adjustments to an image to either correct for, or simulate, different types of film and video lenses. As a corrective measure, you can use this node to remove barrel distortion in an image. You can also use this node to simulate the lens used in one image, in order to warp another image you're compositing over it so that they appear to have been shot using the same lens. This node affords more appropriate control over the result than the *PinCushion* node.

Before

After *LensWarp* undistort

There are two ways you can use the *LensWarp* node.

Manual Adjustments

You can manually adjust the factor and Kappa values to create the desired effect. If you're making a simple correction, this may be the fastest way to proceed.

Automatic Calibration

You can also draw shapes along the edge of any distorted features within the image that are supposed to be straight, and use the calibrate button to automatically correct the image. This provides the most accurate result, especially for moving images.

To automatically calibrate the *LensWarp* node:

1. Attach the *LensWarp* node to an image with wide-angle distortion.

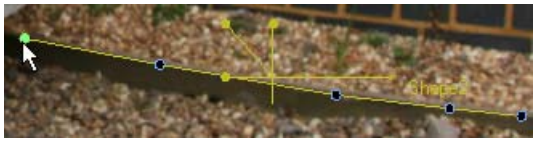
   By default, the *LensWarp* node is in Add Shapes mode.

2. In the Viewer, identify one or more curved features that should be straight, then trace each curve with an open shape.

   Important: Make sure the *LensWarp* node is turned off in the Parameters tab before adding calibration shapes.

   Each feature should be traced with at least four points. The more curved a feature is, the more points you should use to trace it with. On the other hand, you should rarely need to use more than ten points per feature.

To finish drawing an open shape, double-click to draw the last point and end the shape.



Note: The *LensWarp* node does not use Bezier curves.

3 If you're correcting an image sequence, scrub through the frame range to find more curved features, then trace these as well.

The more features you identify in different areas of the frame, the more accurate the final result will be.

4 When you're finished, click the "analyze" button in the *LensWarp* parameters to calculate the result.

The analyze state changes to "analysis up to date."

5 Do one of the following:

- If you're correcting distortion in an image, set the *LensWarp* node to Undistort.
- If you're applying the distortion from this image to another one, detach the *LensWarp* node, reattach it to the other image, and set the *LensWarp* node to Distort.

LensWarp Viewer Shelf Controls

This node has a subset of Shake's standard shape-drawing tools that you can use to identify distorted features for automatic calibration.

Button		Description
	Add Shapes	Click this button to add shapes with which to trace distorted features that you want to correct. You can only draw polygonal shapes with the *LensWarp* node.
	Edit Shapes	Click this button to edit shapes that you've made.
	Import/Export Shape Data	Lets you import or export shapes.

Button		Description
	Delete Control Point	Select a point and click this button to remove it from the shape.
	Enable/Disable Shape Transform Control	Lets you show or hide the transform control at the center of each shape. Hiding these controls will prevent you from accidentally transforming shapes while making adjustments to control points.

Parameters

This node displays the following controls in the Parameters tab:

analyze

After you've created one or more shapes to trace distorted features within the image in the Viewer, clicking calibrate automatically sets the factor and Kappa parameters to the values necessary to either correct or match the lens distortion.

analyze state

This parameter indicates whether or not the analysis has been made.

Off, Distort, Undistort

Use this control to toggle the *LensWarp* node between three modes:

- *Off:* Disables the effect of the *LensWarp* node. Make sure the *LensWarp* node is turned off before you add calibration shapes.
- *Distort:* Warps the image to introduce lens curvature to the image. This mode is useful for matching the lens curvature in another image.
- *Undistort:* Warps the image to remove lens curvature.

factor

A two-dimensional parameter that lets you adjust the amount of vertical and horizontal curvature. Opening the subtree lets you edit the X and Y dimensions independently.

kappa

The amount of warping applied to the image. Typical adjustments fall between 0 and 1.5, but higher adjustments can be made if necessary.

Negative values produce opposite effects—if the *LensWarp* node is set to Distort, negative kappa values undistort the image. If it is set to Undistort, negative values distort the image. This enables you to animate changes from one state to another.

center

The center of the warp performed to the image. By default, xCenter is set to `width/2`, and yCenter is set to `height/2`.

overSampling
An integer value that represents the numbers of samples per pixel that are taken into account when performing a warp. This parameter is set to 1 by default, which results in faster rendering times. However, extreme warping effects may introduce aliasing artifacts that can be reduced or eliminated by increasing this value, up to a maximum value of 4. Increasing this parameter may cause render times to increase dramatically.

Note: Although the slider is limited to a range of 1 to 4, you can enter larger values into this parameter's value field.

PinCushion

The *PinCushion* node distorts the corners of the image in and out to mimic a particular type of edge lens distortion. You can push the values below 0 as well.

Parameters

This node displays the following controls in the Parameters tab:

overSampling
The actual number of samples per pixel equals this number squared. For better antialiasing, increase the number.

xFactor, yFactor
The amount of distortion. Adjusting the xFactor bends the sides of the image, while adjusting the yFactor bends the top and bottom. Positive values bow the image outside the frame, and negative values bow the image into the frame.

Randomize

The *Randomize* node randomizes the position of each pixel within a certain distance. To randomize the color, create a *Rand* node, and layer it with your image using an *IMult* node.

Parameters

This node displays the following controls in the Parameters tab:

overSampling
The actual number of samples per pixel equals this number squared. For better antialiasing, increase the number.

seed
When Shake generates a random pattern of values, you need to make sure for purposes of compositing that you can recreate the same random pattern a second time. In other words, you want to be able to create different random patterns, evaluating each one until you find the one that works best, but then you don't want that particular random pattern to change again.

Shake uses the seed value as the basis for generating a random value. Using the same seed value results in the same random value being generated, so that your image doesn't change every time you re-render. Use a single value for a static result, or use the expression "time" to create a pattern of random values that changes over time.

For more information on using random numbers in expressions, see "Reference Tables for Functions, Variables, and Expressions" on page 941.

xAmplitude, yAmplitude
The amount of randomization in pixel distance.

xOffset, yOffset
An offset to the random pattern.

Turbulate

The *Turbulate* node is similar to the *Randomize* node, except that it passes a continuous field of noise over the image, rather than just randomly stirring the pixels around. This is a processor-intensive node.

Parameters

This node displays the following controls in the Parameters tab:

overSampling
The actual number of samples per pixel equals this number squared. For better antialiasing, increase the number.

detail
The amount of fractal detail. The higher the number, the more iterations of fractal noise. This can be very processor-intensive.

xNoiseScale, yNoiseScale
The scale of the waves.

xAmplitude, yAmplitude
The amount of randomization in pixel distance.

xOffset, yOffset
Lets you offset the effect on the image.

seed
When Shake generates a random pattern of values, you need to make sure for purposes of compositing that you can recreate the same random pattern a second time. In other words, you want to be able to create different random patterns, evaluating each one until you find the one that works best, but then you don't want that particular random pattern to change again.

Shake uses the seed value as the basis for generating a random value. Using the same seed value results in the same random value being generated, so that your image doesn't change every time you re-render. Use a single value for a static result, or use the expression "time" to create a pattern of random values that changes over time.

For more information on using random numbers in expressions, see "Reference Tables for Functions, Variables, and Expressions" on page 941.

Twirl

The *Twirl* node creates a whirlpool-like effect.

Parameters

This node displays the following controls in the Parameters tab:

startAngle
The amount of twirl near the Center of the rotation.

endAngle
The amount of twirl away from the Center.

aspectRatio
This parameter inherits the current value of the defaultAspect global parameter. If you're working on a nonsquare pixel or anamorphic image, it is important that this value is set correctly to avoid unwanted distortion in the image.

xCenter, yCenter
Center of the twirl.

xRadius, yRadius
Radius of the twirl circle.

bias
Controls how much of the twist occurs between the center and the Radius. 0 means the outer Radius is not rotated; 1 means the center is not rotated.

antialiasing
Anti-aliasing on the effect.

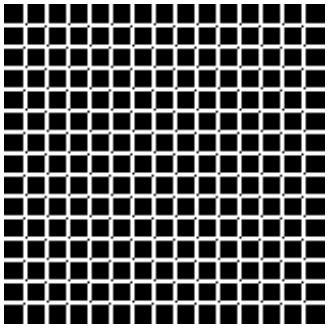
WarpX

The *WarpX* node is a general-purpose warping tool, similar to *ColorX*, except that instead of changing a pixel's color, you change its position. Any formula can be entered in the xExpr and yExpr fields for custom warps. You can also create multiline expressions in this node.

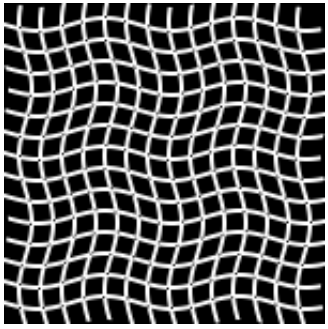
One note of caution: *WarpX* does not properly set the DOD, so you may need to manually attach a Transform–*SetDOD* node following the *WarpX* node.

The following examples are on a grid. By modifying x and y, you specify from what pixel the information is pulled. For example, *x+5, y+5* shifts the image left and down by 5 pixels.

Expr	Value
xExpr	`x`
yExpr	`y`

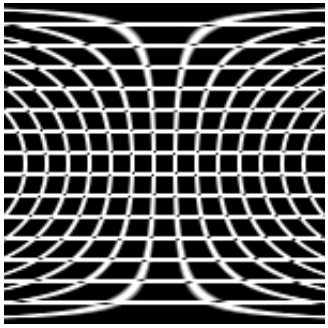


Expr	Value
xExpr	`x+3*sin(y/10)`
yExpr	`y+3*sin(x/10)`



Expr	Value
xExpr	`float xc=(x-width/2); float yc=(y-height/2); float r=sqrt(xc*xc+yc*yc); float newr=r*sin(r/100); width/2+ newr*xc/r`
yExpr	`float xc=(x-width/2); float yc=(y-height/2); float r=sqrt(xc*xc+yc*yc); float newr=r*sin(r/100); height/2+ newr*yc/r`

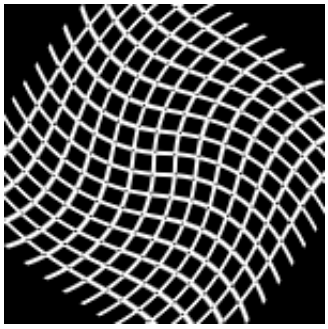
Expr	Value
xExpr	`((x/width-0.5)*sin(3.141592654*y/height)+0.5)*width`
yExpr	`y`



Expr	Value

xExpr	`float xc=(x-width/2); float yc=(y-height/2);`
	`float r=sqrt(xc*xc+yc*yc); float a=atan2(yc,xc);`
	`float newA= a+3.141592654/2*r/200; width/2+r*cos(newA)`

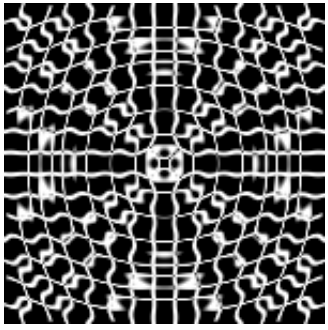
yExpr	`float xc=(x-width/2); float yc=(y-height/2);`
	`float r=sqrt(xc*xc+yc*yc); float a=atan2(yc,xc);`
	`float newA= a+3.141592654/2*r/200; height/2+r*sin(newA)`



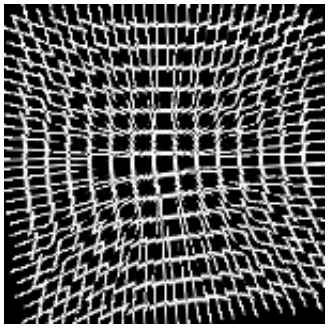
Expr	Value

xExpr	`float xc=(x-width/2); float yc=(y-height/2);`
	`float r=sqrt(xc*xc+yc*yc); float newr=r+3*sin(r/2);`
	`width/2+ newr*xc/r`

yExpr	`float xc=(x-width/2); float yc=(y-height/2);`
	`float r=sqrt(xc*xc+yc*yc); float newr=r+3*sin(r/2);`
	`height/2+ newr*yc/r`



Expr	Value
xExpr	`float xc=(x-width/2); float yc=(y-height/2); float r=sqrt(xc*xc+yc*yc); float a=atan2d(yc,xc); float newA= a+((int)a)%8-4; width/2+r*cosd(newA)`
yExpr	`float xc=(x-width/2); float yc=(y-height/2); float r=sqrt(xc*xc+yc*yc); float a=atan2d(yc,xc); float newA= a+((int)a)%8-4; width/2+r*sind(newA)`



Expr	Value
xExpr	`float xc=(x-width/2); float yc=(y-height/2); float r=sqrt(xc*xc+yc*yc); float newr=r*r/200; width/2+ newr*xc/r`
yExpr	`float xc=(x-width/2); float yc=(y-height/2); float r=sqrt(xc*xc+yc*yc); float newr=r*r/200; height/2+ newr*yc/r`

Parameters

This node displays the following controls in the Parameters tab:

overSampling
The actual number of samples per pixel equals this number squared. For better antialiasing, increase the number.

xExpr, yExpr
The expression to be placed. See above for examples.

xDelta, yDelta
Sets the maximum distance that any pixel is expected to move, but doesn't actually move it. A given pixel in an image may be affected by any pixel with the Delta distance. This means that Shake must consider a much greater amount of pixels that may possibly affect the currently rendered pixel. This is bad. However, if you set a Delta value that is too small, you get errors if your expression tells the pixel to move beyond that limit. Therefore, you'll want to do some testing to balance between speed with errors, or accuracy with drastically slower renders. It is recommended that you start small and increase the size until the errors disappear.

The Warper and Morpher Nodes

Shake's shape-based warping nodes, the *Warper* and *Morpher*, let you easily create specific warping effects using shape tools that are very similar to those used by the *RotoShape* node. Using these tools, you can deform parts of an image to conform to shapes you create in the Viewer.

Warper and Morpher Memory Usage

The *Warper* and *Morpher* nodes use a lot of memory when processing high-resolution images—using four image channels of the full image buffer in float space for each processing thread. As a result, memory usage may become an issue when warping and morphing large images with multi-threaded processing enabled. In this situation, virtual memory usage may noticeably slow processing speed when the maximum available RAM is used.

For example, if you have 2 GB of RAM in your computer, and Shake plus assorted OS operations use 300 MB, this leaves 1.7 GB of total memory for image processing by the *Warper* or *Morpher* node for any given frame. You can calculate the RAM used for a frame at a given image size using the following formula:

```
4 * (image width * image height * 4) * (number of threads)
```

Using this formula yields the following memory usage table:

Number of Threads	2K Image Calculation	4K Image Calculation	8K Image Calculation
1	49MB	195 MB	778 MB
2	97 MB	389 MB	1.6 GB

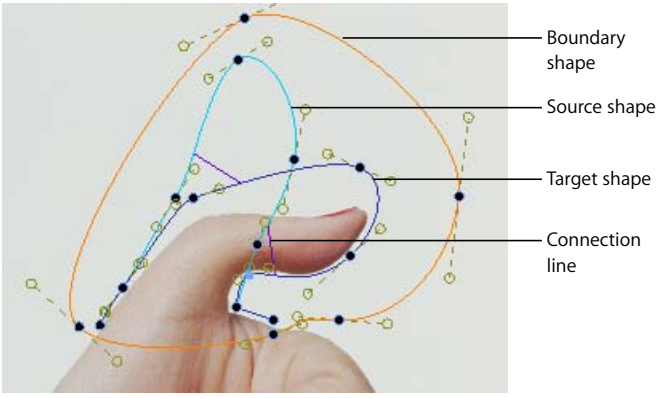
If you don't have enough RAM to handle the resolution you're working at, switch the maxThread parameter (in the renderControls subtree of the Globals tab) to 1. This reduces the memory requirements for this operation.

Using Shapes to Warp and Morph

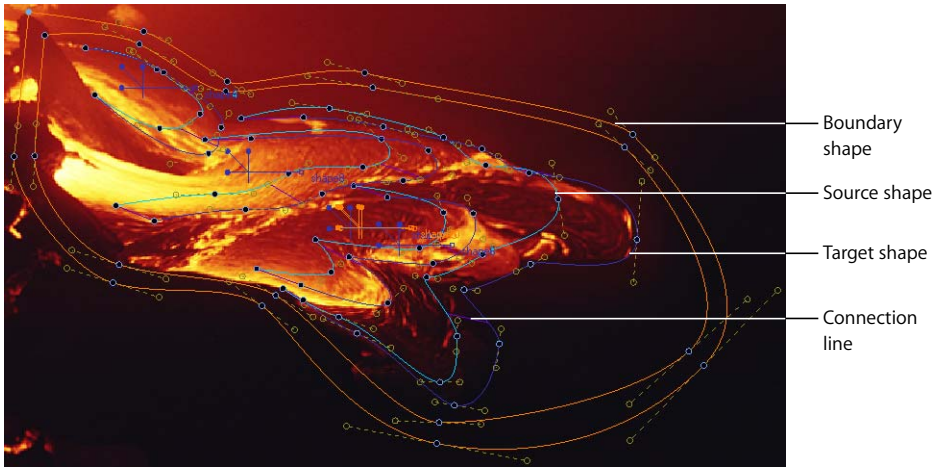
Both the *Warper* and *Morpher* nodes allow you to use animated shapes to control and animate the deformation of an image. Both nodes work using four types of shapes that you draw. These shapes work together to define which parts of an image will be deformed to fit into shapes that you define.

Types of Control Shapes

The following simple example clearly shows the different shapes used to create a warp effect used to bend the thumb down.



The following example shows multiple instances of these same basic shapes employed to create a more complex effect—that of lava within a still image flowing forward.



- *Source Shapes:* These are shapes you draw that conform to the subject of the source image you want to deform. They generally follow well-defined contour lines of the subject—examples might include the edge of a person's face, the contours of the eyes, eyebrows, nose, and mouth, or the outline of an arm or leg. Source shapes are light blue by default.
- *Target Shapes:* These are shapes you draw that define the shape you want the deformed image to conform to. For example, if you want to warp the eyes of a cat to make them appear to bulge on cue, you would create animated target shapes defining the new shape of the eyes. Target shapes are dark blue by default.
- *Connection Lines:* These are automatically created when you connect a source shape to a target shape, and indicate the correspondence of each point in a source shape to its destination on the target shape. It is by pairing source shapes with target shapes that Shake is able to create controlled deformations of an image. Although four connection lines are automatically created for each source/target shape pair, you can create additional ones to give you added control over the deformation. Connection lines are purple by default.
- *Boundary Shapes:* The *Warper* and *Morpher* nodes can sometimes create unwanted deformations in areas surrounding the parts of the image you intend to manipulate. Boundary shapes are essentially shapes that are both source and target shapes, which is how they keep affected pixels from moving. You can use boundary shapes to minimize the effect of a warp on surrounding parts of an image, either by excluding whole regions of the source, or by "pinning down" specific areas that you don't want to be deformed. You can create as many boundary shapes as necessary, since it may take more than one to pin down an image completely. Boundary shapes are orange by default.

- *Displaced Target Shapes:* These are not shapes you either create or modify directly. Instead, they're indicators that show the amount of displacement in that region of the image, based on either the overallDisplacement parameter, or that shape's Displacement parameter if it is being animated independently. These shapes are designed to help you see what the deformation will be without having to render the entire image. Displaced target shapes are pink by default.

Note: The colors of each control shape type can be modified in the shapeColors subtree of the colors subtree of the Globals tab.

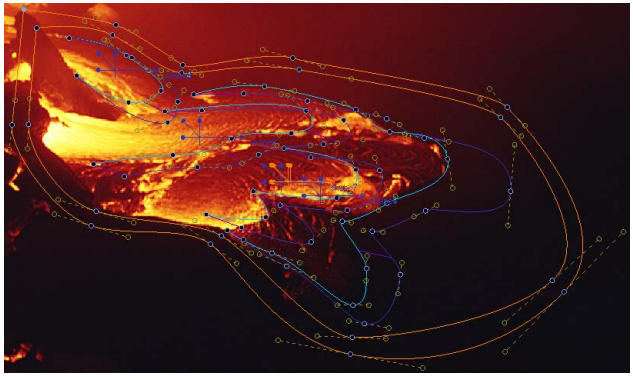
For information on drawing shapes in the *Warper* and *Morpher* nodes, see "Creating and Modifying Shapes" on page 830.

Source shapes and target shapes may be drawn separately, or you can duplicate the source shape you create and modify it to quickly create a target shape. It's not necessary for the source and target shapes to have the same number of points, since the actual path that an animated deformation will follow runs along the connection lines that appear once you connect a source shape to a target shape.

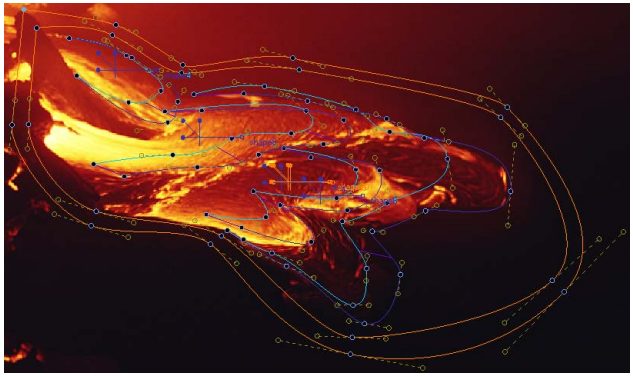
Controlling a Warp Effect Using Several Shapes

In both the *Warper* and *Morpher* nodes, you may create as many source/target shape pairs as necessary to deform various parts of the subject. Unlike the *RotoShape* node, which only allows for the creation of closed shapes, the *Warper* and *Morpher* nodes allow you to create closed shapes, open-ended shapes, and single-point shapes. This flexibility allows you to create any kind of deformation you need.

In some instances, you can create a more convincing effect using multiple source/target shape pairs. In the following example, four source/target shape pairs are used to create the effect of the lava flowing forward.



Multi-shape warp with OverallDisplacement at 0



Multi-shape warp with OverallDisplacement at 1

In this example, a single large shape is used to pull the entire outside shape of the lava forward. An unbroken shape is usually best for warping the outer boundries of an image, but this can sometimes create an unnatural stretching within the subject itself.

By adding individual source/target pairs within the lava image, the individual ripples and eddies of the lava can be individually animated to achieve a more natural-looking effect.

For more information about drawing and manipulating shapes, see "Creating and Modifying Shapes" on page 830.

Animating Control Shapes

Unless you're deforming a still image, it will probably be necessary to animate the source and target shapes you use to fit the motion of the subject you're deforming. For example, if you're creating a warp for an actor who's moving, you'll need to animate the source shape to conform to the outlines of the actor so that they follow his or her motion. You'll then need to animate the target outlines to follow the same motion.

Here's a shortcut that may save you some effort when you create a warp effect using an animated shape. First, animate the source shape that defines the area of the image you want to warp. Afterwards, you can duplicate and modify it as necessary to use as the target shape, without having to reanimate the entire shape.

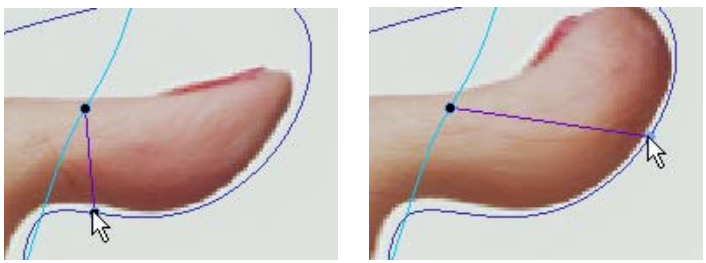
For more information about keyframing shapes, see "Animating Shapes" on page 557.

Using Motion Tracking to Animate Control Shapes

In addition to manually keyframing source and target shapes, you can attach *Stabilize* or *Tracker* nodes to either source or target shapes to aid you when rotoscoping moving features. This works identically to the way you attach *Stabilize* or *Tracker* nodes to shapes in the *RotoShape* node. For more information, see "Attaching Trackers to Shapes" on page 562.

Controlling Warp and Morph Deformation Using Connection Lines

When you first connect a source shape to a target shape in the Viewer, four connection lines appear that run from the source to the target shapes. These lines serve two purposes. First, they show you which segments of a source shape correspond to which segments of its connected target shape. Second, their angles define the path the pixels of the image will follow when warping from their original position to the target position you've defined.

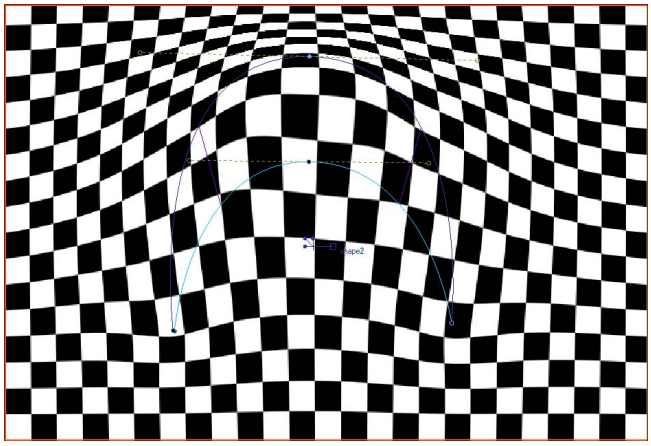


The start and end points of connection lines that are connected to the source and target shapes can be moved by turning on the Edit Connection button, and then dragging them back and forth along the shapes themselves. Changing the angle of the lines by moving the in or out point of a connection line independently allows you to redefine the angle of deformation for all pixels in that area of the warp.

Connection lines can be moved, and even animated, to control the speed and direction of deformation. Additional connection lines may also be created to give you more precise control over the deformation itself.

Using Boundary Shapes to Limit Deformation in an Image

The *Warper* and *Morpher* nodes both work by pushing and pulling pixels from the region of an image defined by the source shapes to the region defined by the target shapes. When part of an image is warped, the surrounding area stretches to accommodate the change, as if the image is on a sheet of rubber being pushed and pulled to distort it.



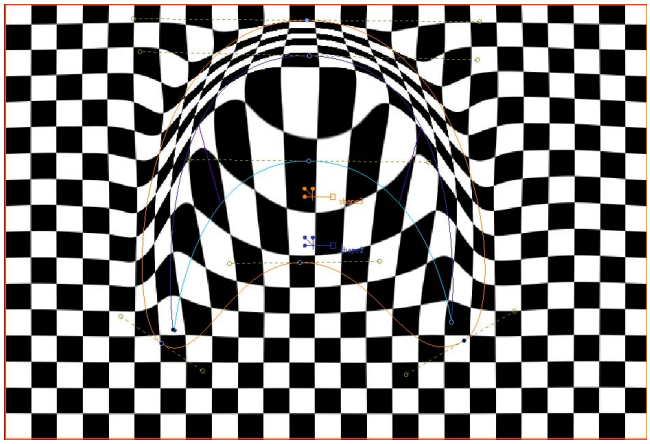
Warp without boundary shape

The region affected by the resulting deformation is not limited to the area defined by the source/target shape pairs. In fact, you'll notice that a significant area of the image surrounding each source/target shape pair is also deformed. While there is a 100 percent displacement at the actual position of the source and target shapes, the total area of deformation lessens gradually with the distance from the shape pair. This may result in a warp or morph not only affecting the intended subject, but also the surrounding background.

This aspect of the Shake warper is useful in that it helps to smooth the transition between the warped and unwarped parts of your image, resulting in a more realistic effect. It also means that sometimes it's not necessary to create as many source/target shape pairs as you might think—a single shape pair's area of influence may be enough to create the effect you want.

On the other hand, there are usually parts of an image that you don't want warped. For example, if you're warping someone's eyebrows, chances are you don't want his or her hair to be distorted as well. You exclude parts of an image from being affected by the *Warper* or *Morpher* node using boundary shapes.

It's important to understand that boundary shapes don't eliminate distortion from the surrounding image; they minimize it.



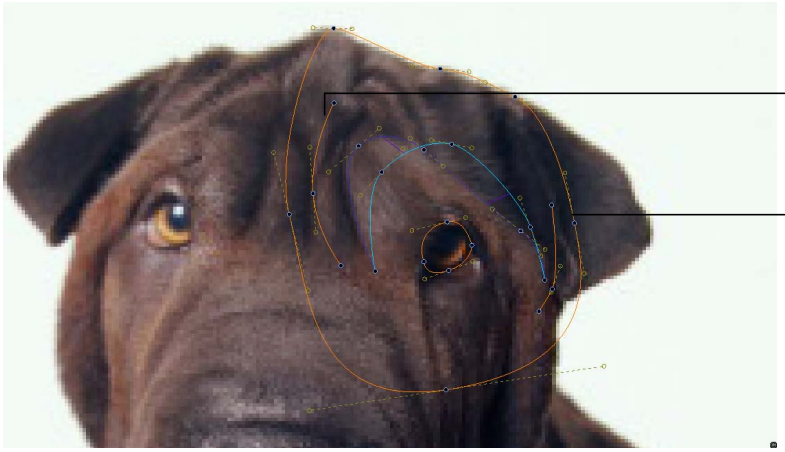
Warp with boundary shape

It may take more than one boundary shape to completely lock down an image. Fortunately, you can create as many boundary shapes as necessary to eliminate unwanted distortion in an image.

Important: Target shapes should never cross boundary shapes. Doing so may create unwanted distortion, possibly tearing in the resulting image.

There are many ways you can use boundary shapes to isolate parts of an image from deformation. One technique is to use a closed shape to surround a pair of source/target shapes, thus minimizing their effect on the surrounding area of the image.

For example, if you want to isolate a warping operation to a particular region, you can create a closed boundary shape to lock off just that area. Sometimes, you may have to use several concentric rings of boundary shapes to completely lock down an area of the image. You can also use open boundary shapes to "pin down" specific areas of an image that you don't want to be affected by a warping effect. For example, if you were creating a warp effect to manipulate an animal's face, you could use a combination of open and closed shapes and single-point shapes to prevent the eyes and nose from being affected by the warp you're applying to the eyebrow area.



Note: By default, the outer edge of the frame is also used as a boundary shape, by default. This behavior can be disabled by turning off the addBorderShape parameter in the Parameters tab for the *Warper* or *Morpher* node you're adjusting, but this may produce unexpected results.

Isolating the Subject of Deformation Prior to Warping or Morphing

Even when you use one or more boundary shapes to pin down areas surrounding a warp effect, you may find that some of the surrounding image is still affected, however slightly. For this reason, it may be useful to isolate the subject of the image prior to using either the *Warper* or *Morpher* node. Ideally, the subject of the warp effect was shot against bluescreen or greenscreen, and can be keyed. If not, you can always rotoscope the image using a *RotoShape* node.

In either case, the *Warper* and *Morpher* nodes affect the alpha channel of the image along with the RGB portion, so you can always add either node to the tree after you've isolated your subject by keying or rotoscoping. This way, you can add a clean background no matter how extreme the warping effect is.

Creating and Modifying Shapes

Many of the shape controls of the *Warper* and *Morpher* nodes are identical to those of the *RotoShape* node, and all share the same methods for creating tangents, closing shapes, inserting and deleting points, and so on. If necessary, you can refer to the *RotoShape* documentation for more information on creating and modifying shapes.

Warper and Morpher Viewer Shelf Controls

When a *Warper* or *Morpher* node is selected, the following buttons appear in the Viewer shelf.

Button		Description
	Add Shapes	Creates new shapes. Closed shapes are created by clicking the first shape point you created. Open shapes and single-point shapes are created by double-clicking when creating the last point, or by right-clicking in the Viewer, then choosing Finish Shape from the shortcut menu.
	Edit Shapes	Allows you to edit shapes.
	Connect Shapes	Clicking this button allows you to create a source/target shape pair by first clicking the shape you want to be the source, and then clicking a second time the target shape you want to link it to. To define a boundary shape, click this button, then click twice a shape you want to turn into a boundary shape. This effectively makes a single shape into both a source and target shape.
	Edit Connections	Once two shapes have been joined with the Connect Shapes button, the location and angle of each connection line that links source to target shapes may be edited by clicking this button. With this control turned on, select the source or destination point of a connection line.
	Show/Hide Tangents	Toggles the Viewer among showing All shape tangents (the handles that allow you to manipulate Bezier curves), None, or Pick, which only shows the shape tangents of individually selected points.
	Lock/Unlock Tangents	Locks or unlocks all shape tangents in the Viewer. If locked, shape points may still be moved, but the tangents defining the angle of curvature remain locked.

Button		Description
	Spline/Linear Mode	Toggles selected to act as either corner points or Bezier points.
	Delete Control Point	Deletes selected points on a shape.
	Key Current Shape/ Key All Shapes	Toggles between two shape keyframing modes. In All Shapes, all shapes are keyframed whenever any one shape is modified with Autokey on. In Current Shape, only the selected shape is keyframed when Autokey is on.
	Enable/Disable Shape Transform Control	When turned on, this control makes the shape transform controls for each shape visible in the Viewer. Each shape can be manipulated as a whole using this control. When turned off, all shape transform controls are hidden, and cannot be used.
	Visibility Toggles	These buttons toggle the visibility of specific types of shapes in the Viewer. From left to right, they control: • Source Shape Visibility • Target Shape Visibility • Connection Line Visibility • Boundary (or lockdown) Shape Visibility • Unconnected Shape Visibility • Displaced Target Shape Visibility Each control affects the visibility of all shapes of that type in the Viewer. Individual shapes may be made invisible using controls in the Parameters tab. However, the Visibility toggles in the Viewer shelf supersede the Visibility settings in the Parameters tab. Each setting in the Select Display Image pop-up menu of the *Warper* and *Morpher* Viewer shelf allows these controls to be toggled independently. For example, in the *Warper*, the visibility settings of the source image can differ from those used by the warped (target) image.

Button		Description
	Lock Shapes	These three buttons lock all source, target, and boundary shapes in the Viewer, preventing them from being edited. Each control locks all shapes of that type in the Viewer. Individual shapes may be locked using controls in the Parameters tab. However, the Shape Lock buttons in the Viewer shelf supersede the Lock buttons in the Parameters tab.
	Select Display Image	The Source/Warped Image pop-up menu allows you to toggle the Viewer's display between the unmodified and modified images. You may quickly jump between views by pressing: • F9 to view the original source image • F10 to view the original target image (*Morpher* only) • F11 to view the warped image

Drawing and Editing Shapes

The biggest difference between drawing shapes with the *RotoShape* node and with the *Warper* and *Morpher* nodes is that while *RotoShape* allows you to draw only closed shapes, the *Warper* and *Morpher* nodes allow you to create not only closed shapes, but also open shapes and single-point shapes. Open shapes make it very simple to define deformations for visual features like eyebrows, muscle outlines, and other contours that don't require a closed outline. Single-point shapes allow you to define deformations for small image details, and are also very effective as boundary shapes you can use to pin down parts of the image you don't want to be affected by nearby source/target shape pairs.

The *Warper* and *Morpher* nodes both warp the image using the same shape controls, and the methods used to create and edit shapes in each node are identical.

Drawing New Shapes

Drawing new shapes works the same whether you're creating a source, target, or boundary shape. In each case, you create a new, unassigned shape first, and then assign its type in a subsequent step. Unassigned shapes appear yellow, by default.

To create a new unassigned shape:

1 Click the right side of a *Warper* or *Morpher* node to load its parameters into the Parameters tab, and its controls into the Viewer shelf.

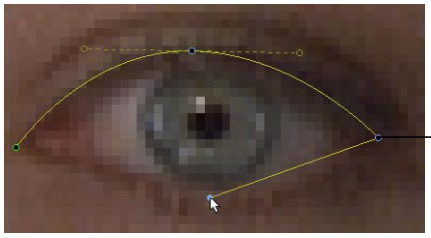
2 In the Viewer shelf, click the Add Shape button.

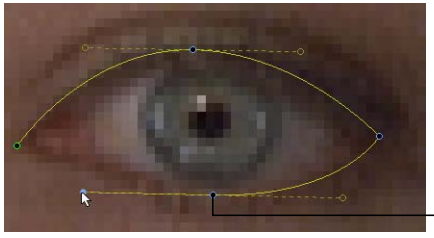
3 In the Viewer, begin drawing a shape by clicking anywhere on the image to place a point.

If necessary, zoom into the image in the Viewer to better trace the necessary features of the subject you want to warp or morph.

4 Continue clicking to add more points to the shape.

- Click once to create a sharply angled point.
- To create a point with tangent controls to make a Bezier curve, drag to one side of the point until the angled point becomes a curve.

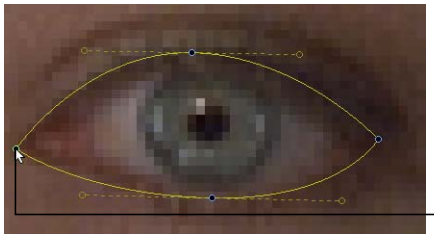




Note: The distance you have to drag before the angled point becomes a curve is customizable via the rotoTangentCreationRadius parameter in the shapeControls subtree of the guiControls subtree in the Globals tab. For more information on customizing Shake's shape creation tools, see "Customizing Shape Controls" on page 843.

5 There are three ways you can end shape drawing to create different kinds of shapes:

- To create a single point shape, right-click in the Viewer immediately after creating the first point, then choose Finish Shape from the shortcut menu.
- To create an open shape, either double-click when creating the last point of the shape, or right-click and choose Finish Shape from the shortcut menu.
- To create a closed shape, click the first point of the shape you created.



Important: You can only create single-point shapes and open shapes in the *Warper* and *Morpher* nodes. You cannot create these kinds of shapes in the *RotoShape* node.

Every time you create a new shape, an additional shape parameter appears in the Parameters tab of the corresponding *Warper* or *Morpher* node. By default, each new shape parameter that's created is named "shape1Name," and the middle number is incremented with each new shape you draw. These names can be changed to more easily identify the specific parts of the subject you've isolated for individual manipulation later.

Editing Shapes

Once you've created a shape, there are several ways you can modify it. These techniques also work for keyframing shapes used for animated warping or morphing effects. For more information about keyframing shapes, see "Animating Single or Multiple Shapes" on page 558.

When editing shapes that are close to other shapes, it may be helpful to turn off Enable/Disable Shape Transform Control in the Viewer shelf, to hide transform controls from other shapes that may overlap the shape you're editing. After your source/target shape pairs have been defined, it may also be helpful to turn off the visibility of shape types that you don't need to see. For example, turning off the visibility of all source shapes while you're editing their corresponding target shapes will prevent accidental adjustment of the wrong overlapping points. You can turn different groups of visibility controls on and off for each setting of the Select Display Image pop-up menu in the Viewer shelf.

Important: In order to edit *Warper* or *Morpher* shapes, it's important to make sure the Edit Shapes button is turned on.

To edit a shape:

1. Click the right side of a *Warper* or *Morpher* node to load its parameters into the Parameters tab, and its controls into the Viewer shelf.
2. In the Viewer shelf, click the Edit Shapes button.

3. Select one or more points you want to edit by doing one of the following:
   - Click a single point to select it.
   - Shift-click additional points to add them to the selection.
   - Click in the Viewer and drag a selection box over all the points you want to select.

- Hold the Shift key down and drag to add points to a selection.
- Hold the Command or Control key down, then drag to remove points from the selection.
- Move the pointer over the edge, or the transform control, of a shape, and press Control-A or Command-A to select every point on that shape.

4 When the selected points are highlighted, rearrange them as necessary by doing one of the following:

- To move one or more selected points, drag them where you want them to go.
- To move one or more selected points using that shape's transform control, press the Shift key while you drag over the transform control.

  Note: Using the transform control without the Shift key pressed modifies the entire shape, regardless of how many points are selected. For more information on using the transform control, see page 837.

To add a point to a shape:

1 Click the Edit Shapes button.

2 Shift-click the part of the shape where you want to add a control point.

A new control point appears on the shape where you clicked.

To remove one or more points from a shape:

1 Select the point or points you want to remove.

2 Do one of the following:

- Click the Delete Control Point button in the Viewer shelf.

- Press the Delete key (near the Home and End keys).

Those points disappear, and the shape changes to conform to the remaining points.

To convert linear points to Bezier points, and vice versa:

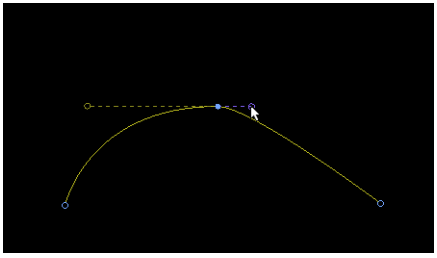
1 Select the point or points you want to convert.

2 Click the Spline/Linear Mode button to convert linear points to Bezier points, or Bezier points to linear points.

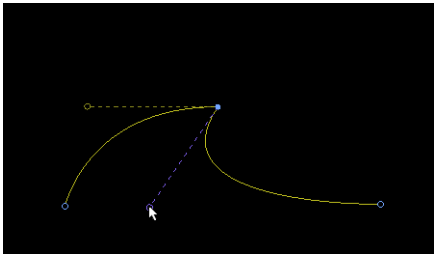
An optional step is to set the Show/Hide Tangents button to All or Pick in order to view tangents as they're created.

To change a curve by editing a point's tangent handles:

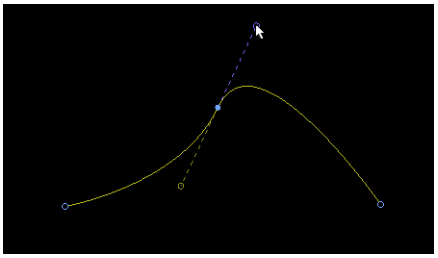
1. Make sure the Show/Hide Tangents button is set to All (to view all tangents) or Pick (to view only the tangents of points that you select).
2. Make sure the Lock/Unlock Tangents button is set to Unlock.
3. Do one of the following:
   - To change the length of one of the tangent handles independently from the other, while keeping the angle of both handles locked relative to each other, drag a handle to lengthen or shorten it. You can also rotate both handles around the axis of the selected point.



   - To change the angle of one of the tangent handles relative to the other, along with its length, press the Command or Control key while dragging a handle around the axis of the selected point. The selected tangent handle moves, but the opposing tangent handle remains stationary.



To keep the angle of both tangent handles at 180 degrees relative to one another, keeping the lengths of each side of the tangent identical, press the Shift key while dragging either of the tangent handles around the axis of the selected point. If you Shift-drag tangent handles that were previously angled, they are reset.



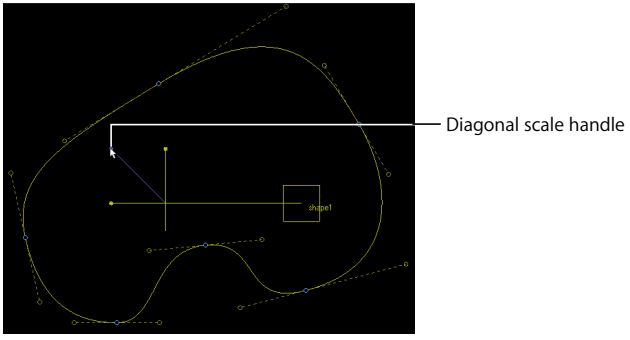
To edit a shape using its transform control:

1 Make sure that Enable/Disable Shape Transform Control is turned on.

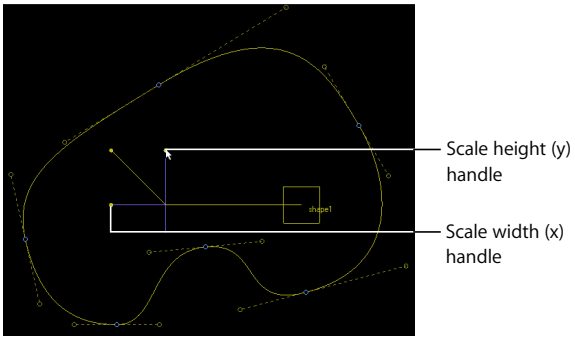
When you move, scale, or rotate a shape using its transform control, each transformation occurs relative to the position of the transform control. To move a shape's transform control in order to change the center point about which that shape's transformation occurs, press the Command or Control key while dragging the transform control to a new position.

2 To manipulate the shape, drag one of the transform control's handles:

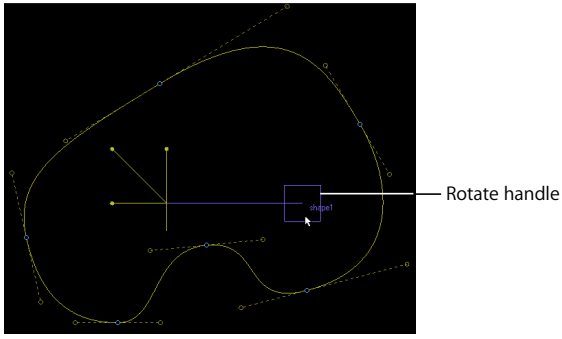
- Drag the center of the transform control to move the entire shape in the Viewer. Both the X and Y handles will highlight to show you're adjusting the X and Y coordinates of the shape.
- Drag the diagonal scale handle to resize the shape, maintaining its current aspect ratio.



- Drag the X handle to resize the shape horizontally, or drag the Y handle to resize the shape vertically.



- Drag the rotate handle to rotate the shape about the axis of the transform control.



Showing and Hiding Shapes

Individual shapes may be hidden, if necessary, to help you isolate one or more shapes when making adjustments. Hiding shapes simply makes them invisible. Hiding a shape has no effect on the resulting warp effect—all source/target shape pairs continue to warp the image as before.

When a *Warper* or *Morpher* node is selected in the Node View, each shape in that node is labeled in the Viewer. By default, each shape is numbered in the order that it was created. These names can be customized in that shape's corresponding parameter in the Parameters tab. These names help you identify which shapes are which when you're changing their individual visibility.

To show or hide an individual shape directly in the Viewer, do one of the following:

- Right-click anywhere in the Viewer to display the Viewer shortcut menu, then choose the Shape Visibility submenu, and select a label that corresponds to the shape you want to show or hide. Shapes that are checked are shown, while shapes that are unchecked are hidden.
- In the Parameters tab, click the Visibility button of the shape parameter that corresponds to the shape you want to show or hide. These controls are linked to the settings in the Shape Visibility submenu of the Viewer shortcut menu. Changes made to one automatically apply to the other.

You can also show or hide all shapes of a particular type using the Visibility toggles in the Viewer shelf. Each control affects the visibility of all shapes of that type in the Viewer. The Visibility toggles supersede the Visibility settings in the Parameters tab.

Each setting in the Select Display Image pop-up menu in the Viewer shelf of the *Warper* and *Morpher* nodes allows these controls to be set independently. For example, in the *Warper*, the visibility settings set when displaying the source image can differ from those set when displaying the warped (target) image.

To show or hide all shapes of a particular type:

- Click the Visibility toggle in the Viewer shelf that corresponds to the shape type you want to hide.

Duplicating Shapes

A fast and easy way to create corresponding target shapes once you've drawn a source shape is to duplicate it, then modify the duplicate. This is especially useful for instances where the general shape of the target you want to create is similar to the source.

To duplicate a shape:

1. Click the Edit Shapes button to allow you to select shapes in the Viewer.
2. Move the pointer over the edge, or the transform control, of the shape you want to duplicate, then right-click and choose one of the following commands from the shortcut menu:
   - Choose Duplicate Shape to simply duplicate the shape.
   - Choose Duplicate and Connect Shape (or press Control-D or Command-D) to duplicate the shape and automatically connect the duplicate to the source shape you clicked.

Note: After using the Duplicate or Connect Shape command, locking or hiding the source shape immediately insures that you won't accidentally modify it when making changes to the new duplicate.

Copying Shapes From a RotoShape Node

You can copy shapes from a *RotoShape* node and paste them into a *Warper* or *Morpher* node for use as a source, target, or boundary shape. This is especially useful in cases where you've already isolated the subject using a *RotoShape* node.

Important: If you copy a shape with a soft edge from a *RotoShape* node, only the main center shape is pasted into a *Warper* or *Morpher* node. The soft edges are not used.

To copy a shape from a *RotoShape* node:

1. With the pointer over the transform control of the shape you want to copy in the Viewer, do one of the following:
   - Right-click, then choose Copy Shape from the shortcut menu.
   - Press Command-C or Control-C.

   Note: To copy all currently visible shapes in a node, Control-click or right-click in the Viewer, then choose Copy Visible Shapes from the shortcut menu.

   Important: When copying a shape, the pointer must be directly over the shape you intend to copy. Otherwise, you may not copy the correct shape.
2. Select the *Warper* or *Morpher* node into which you want to paste the rotoshape.
3. Do one of the following:
   - Right-click in the Viewer, then choose Paste Shapes from the shortcut menu.
   - Press Command-V or Control-V.

   The pasted shape appears just like any other newly created closed shape, and you can modify or duplicate it as necessary.

Connecting Source and Target Shapes

To create the actual warp or morph effect, you need to connect each source shape you've created to a corresponding target shape. You can do so by either drawing two shapes separately and connecting them afterwards, or by drawing the source shape and duplicating it to use it as a starting point for the target. (Two shortcut menu commands—Duplicate Shape, and Duplicate and Connect Shape—make this easy.) Regardless of your intended use for the shapes you've created, until they're connected to one another, they remain unassigned.

To connect a separately drawn source shape to a target shape:

1. Click the Connect Shapes button.

2. Click a source shape.
3. Immediately click the target shape you want to connect to the source shape.

After the shapes are connected, the source shape appears with a light blue path, and the target shape appears with a dark blue path, indicating that the connection has been made. Purple connection lines appear between the source and target shapes to show which parts of each shape are connected.

In the Parameters tab of the corresponding *Warper* or *Morpher* node, an additional connection parameter appears for the connection you established. By default, each new connection parameter that's created is named "connection1Name," with the middle number incremented as each new connection is created. These names can be changed to more easily identify each connection for individual manipulation later on.

To disconnect a source shape from a target shape:

- In the Parameters tab of the corresponding *Warper* or *Morpher* node, click the Delete button of the connection parameter you want to break.

After disconnecting a source/target shape pair, both shapes become unassigned, and turn yellow by default.

For more information about the parameters of the *Warper* node, see "Parameters in the Warper Node" on page 846. For more information about the parameters of the *Morpher* node, see "Additional Controls and Parameters in the Morpher Node" on page 855.

Modifying Connection Lines

Once you've connected a pair of source/target shapes, connection lines appear to show the deformation path that pixels in the source shape will follow to conform to the target shape. These connection lines can be moved to change the path and alter the look of the warp or morph effect. You can also add more connection lines to increase the amount of control you have over the warp or morph effect.

To move the start or end point of a connection line independently:

1. Click the Edit Connections button.
2. Drag a connection point to another location on the shape. The connection point's movement is restricted to the contour of the shape.

To move the entire connection line at once:

1. Click the Edit Connections button.
2. Drag a bounding box or Shift-click each point to select both the start and end points of the connection line you want to move.

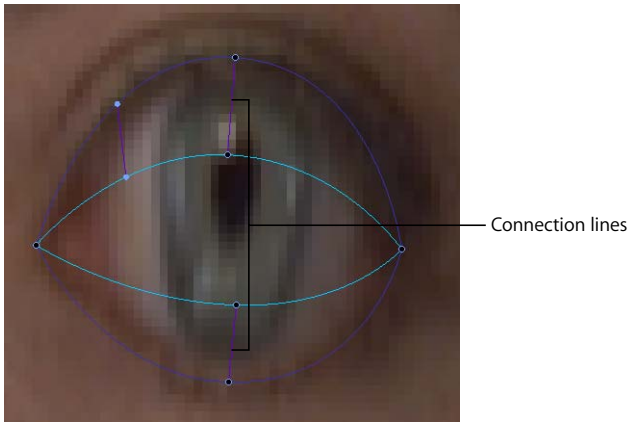
3 With both points selected, dragging one of them will move both at the same time. Both ends of the connection line are restricted to moving along the contours of the source and target shapes, and you can't move a connection point past another connection point.

To add more connection lines to a source/target shape pair:

1 Click the Edit Connections button.

2 Shift-click either a source or target shape at the location where you want a new connection line to be created.

A new connection line is immediately created where you clicked. The other end of the new connection line is placed at the closest point of the corresponding shape in the pair.



Locking Source and Target Shapes

Once you've created one or more source or target shapes, you can lock them individually in a *Warper* or *Morpher* node's Parameters tab, or all together using the Lock Shapes buttons in the Viewer shelf. This is useful if you're modifying source and target shapes that are very close together, and you want to make changes to one without accidentally moving the other.

To lock all source and/or target shapes in the currently selected node, do one of the following:

- Click the Lock Source Shapes button to lock all source shapes.
- Click the Lock Target Shapes button to lock all target shapes.
- Click the Lock Boundary Shapes button to lock all boundary shapes.

You can also lock individual source and target shapes using the lock button to the left of each shape parameter in that node's Parameters tab. However, the Lock Shapes buttons in the Viewer shelf always supersede these individual shape-locking parameter controls. See "Parameters in the Warper Node" on page 846 for more information.

Defining Boundary Shapes

You can use any open or closed shape or single-point shape as a boundary shape to pin down areas of the image you don't want to be warped, or to exclude whole areas of the image from being affected by the source/target shape pairs you've created. You can create as many boundary shapes as you need to lock areas of the image you don't want to be warped.

Since boundary shapes are essentially shapes that are both source and target shapes simultaneously, you also define them using the Connect Shapes button.

To make an unassigned shape into a boundary shape:

1. Select the *Warper* or *Morpher* node you're working on, then create a new shape outlining the region of the image you want to lock down.
2. To turn this shape into a boundary shape, do one of the following:
   - Click the Connect Shapes button, then click the shape you've created twice.
   - Right-click the shape, then choose Set as Boundary Shape from the shortcut menu.

The shape turns orange by default to indicate that it's now a boundary shape, and a new connection parameter appears in the Parameters tab of the *Warper* or *Morpher* node. By default, each connection parameter that defines a boundary shape is named after the shape it corresponds to. For example, if the original shape was named "shape3," the connection parameter that defines it as a boundary shape is named "shape3_boundary."

Turning Boundary Shapes Into Unassigned Shapes

Once you've turned a shape into a boundary shape, the only way to turn it back into an unassigned shape is to delete the connection parameter that corresponds to it in the Parameters tab of the *Warper* or *Morpher* node, using that parameter's Delete button.

Customizing Shape Controls

Several parameters in the shapeColors subtree of the colors subtree, and in the shapeControls subtree of the guiControls subtree of the Globals tab, allow you to customize the color of shapes and behavior shape controls in the Viewer.

Shape Colors

By default, the paths of source shapes are light blue; paths of target shapes are dark blue; paths of connection lines are purple; paths of boundary shapes are orange; and paths of unassigned shapes are yellow. These colors can all be changed using the following parameters in the shapeColors section of the colors subtree in the Globals tab.

Parameter	Shape Type	Default Color
ShapeColor	Unassigned shapes	Yellow
sourceColor	Source shapes	Light Blue
targetColor	Target shapes	Dark Blue
connectionColor	Connection lines	Purple
boundaryColor	Boundary shapes	Orange
lockedColor	Locked shapes	Gray
displacedColor	Displaced Target shapes	Pink

To change the default color of a shape color parameter:

1. Click the color swatch of the shape color parameter you want to change.
2. Use the Color Picker to select a new color to use for that shape type.

All shapes of that type are now displayed with the new default color you selected.

Shape Editing Controls

Various behaviors for selecting points, creating Bezier curves, and adjusting each shape's transform control can be customized in the shapeControls subtree of the guiControls subtree of the Globals tab. You can modify how each of these controls works to better suit your working style or input method—for example, whether you use a graphics tablet or mouse.

Each parameter has a slider that adjusts the control's behavior.

rotoAutoControlScale
An option which, when enabled, increases the size of the transform controls of shapes, based on the vertical resolution of the image to which the shape is assigned. This makes it easier to manipulate a shape's transform control even when the image is scaled down by a large ratio.

rotoControlScale
A slider that allows you to change the default size of all transform controls in the Viewer.

You can also resize every transform control appearing in the Viewer by holding down the Command or Control key while dragging the handles of any transform control in the Viewer.

rotoTransformIncrement
This parameter allows you to adjust the sensitivity of shape transform controls. When this parameter is set to lower values, transform handles move more slowly when dragged, allowing more detailed control. At higher values, transform handles move more quickly when dragged. A slider lets you choose from a range of 1-6. The default value is 5, which matches the transform control sensitivity of previous versions of Shake.

rotoPickRadius
This parameter lets you select individual points on a shape that fall within a user-definable region around the pointer. This allows you to easily select points that are near the pointer that may be hard to select by clicking directly. A slider allows you to define how far the pointer may be from a point to select it, in pixels.

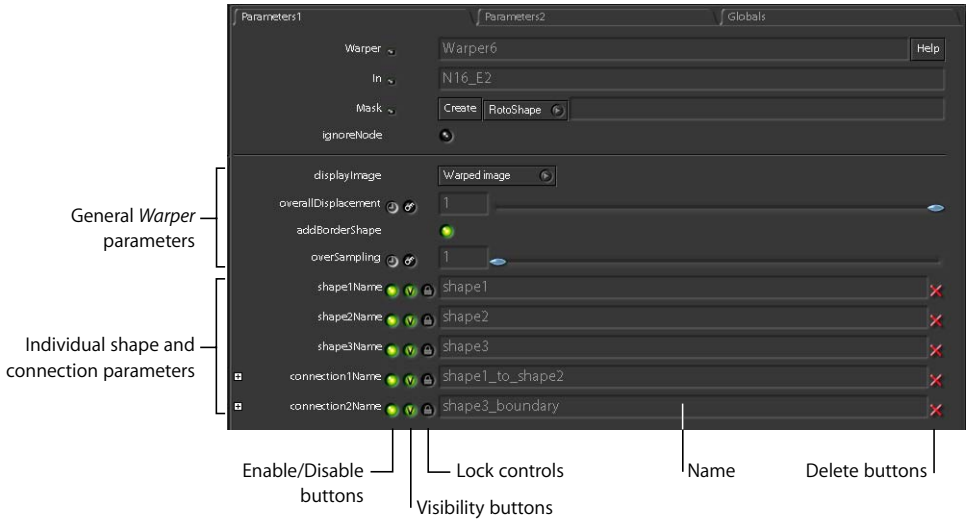
rotoTangentCreationRadius
This parameter lets you define the distance you must drag the pointer when drawing a shape point to turn it into a Bezier curve. Using this control, you can make it easier to create curves when drawing shapes of different sizes. For example, you could increase the distance you must drag to avoid accidentally creating Bezier curves, or you can decrease the distance you must drag to make it easier to create Bezier curves when drawing short shape segments.

Using the Warper Node

The *Warper* node is useful for creating targeted deformations to alter the shape of a subject in an image. Examples might include making someone's nose grow, making an animal's eyes widen in surprise, or causing a bump to grow on someone's forehead. The *Warper* can be used to make a static change to a subject, or it can be animated to cause dynamic changes in the subject's shape.

Parameters in the Warper Node

A simple example of a *Warper* node used to warp an image with a single pair of source/target shapes would appear with the following parameters. (For *Warper* nodes with more source/target shape pairs defined, there will be more shapeName and connectionName parameters listed.)



displayImage

A pop-up menu that allows you to choose whether the Viewer displays the source image or the warped image. The warped image will be neither displayed nor rendered if this pop-up menu isn't set to Warped Image.

overallDisplacement

Defines the amount of displacement that is applied to all source/target shape pairs simultaneously. A value of 0 applies no displacement, 0.5 applies displacement halfway between the source and target shapes, and 1 applies the maximum displacement to match the target shape. It is also possible to set this parameter to a value greater than 1, although this results in an overlapping displacement which may not be desirable.

Note: When creating a warp effect, you may achieve a more realistic or organic effect if you adjust the displacement of each individual source/target shape pair separately, rather than relying on this single control to animate the displacement of every shape pair in the node.

addBorderShape
A button that allows you to use the border of the image as a control shape to limit the warping effect. By default this parameter is turned on, and is the recommended setting for most cases. Turning this control off results in each source/target pair having a considerably more exaggerated effect on the image, and may necessitate the use of additional boundary shapes to control the effect.

overSampling
An integer value that represents the number of samples per pixel that are taken into account when performing a warp. This parameter is set to 1 by default, which results in faster rendering times. However, extreme warping effects may introduce aliasing artifacts that can be reduced or eliminated by increasing this value, up to a maximum value of 4. Increasing this parameter may cause render times to increase dramatically.

Note: Although the slider is limited to a range of 1 to 4, you can enter larger values into this parameter's value field.

- *dodPad:* A subparameter of overSampling. This slider lets you pad the DOD around the image affected by the *Warper* node by 0 to 100 pixels. The *Warper* node tries to automatically calculate a new DOD for the affected image, but in certain instances the resulting DOD may be too small. In these instances, the dodPad parameter lets you expand an incorrectly calculated DOD to avoid clipping.

shape1Name
In this example, the shape1Name parameter represents the source shape. Additional controls in all shape parameters allow you to turn the shape on or off, make the shape itself visible or invisible in the Viewer, lock the shape to prevent any further changes to it, or delete the shape.

shape2Name
In this example, the shape2Name parameter represents the target shape. Each target shape has a corresponding shapeName parameter.

connection1Name
Connection parameters represent both connection lines that connect source shapes to target shapes, and boundary shapes that you've defined. Deleting a connection parameter deletes either the corresponding connection line, or turns a boundary shape back into an unassigned shape.

- *connection1Displacement:* This subparameter of the connection1Name parameter defines the amount of displacement that is applied to the source/target shape pair defined by the connection1Name parameter. Each source/target shape pair has its own corresponding connectionDisplacement parameter, allowing you to animate each warp independently for a more organic, natural look. By default, each connectionDisplacement parameter is linked to the overallDisplacement parameter, so that they all animate together.

A Warper Node Example

The *Warper* node is extremely flexible, and can be used for a wide variety of image distortion or manipulation tasks. In this example, we'll use the *Warper* to change a dog's facial features.

To warp an image:

1 Attach the *Warper* node to an image.

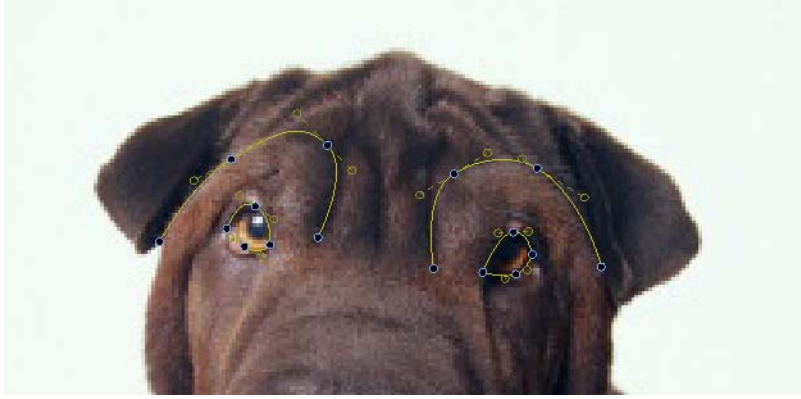
2 First, draw and, if necessary, animate your source shapes (see "Drawing New Shapes" on page 832).

These shapes define the parts of the subject you want to warp.

3 When you're ready to finish your shape, do one of the following:

- Click the first point if you want to create a closed shape.
- Double-click when creating the last point to create an open shape.
- To create a single-point shape, immediately after creating your first point, right-click and choose Finish Shape from the shortcut menu.

To add additional source shapes to define additional warp areas, click the Add Shape button. Each shape you create using the *Morpher* node is yellow, indicating that it's unassigned and does not yet have any effect on the image.



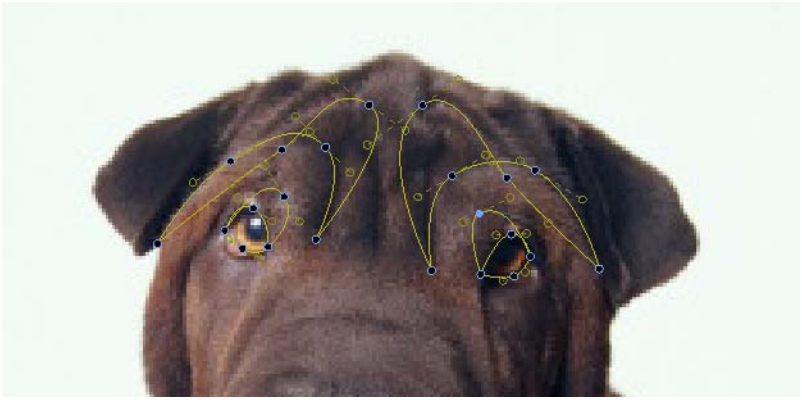
Unassigned shapes defining the area to warp

Next, you need to create a corresponding target shape for each source shape you created. Target shapes define the contour of deformation to which pixels identified by each source shape are moved.

4 Create target shapes using the same shape-drawing techniques used in step 2.

Note: Another technique you can use to create a target shape quickly is to duplicate the source shape by right-clicking it and choosing Duplicate Shape from the shortcut menu. You can also choose Duplicate and Connect Shape (or press Control-D or Command-D), in which case you can skip step 5. If you're using either of these options, you may want to animate the original source shape first, as the copied shape inherits the animation.

As you create target shapes for each source shape, they remain yellow to indicate that they're still unassigned, and have no effect on the image.



Unassigned shapes are yellow.

To create the actual warping effect, you have to connect the shapes you created a pair at a time.

5 To do this, use the following steps:

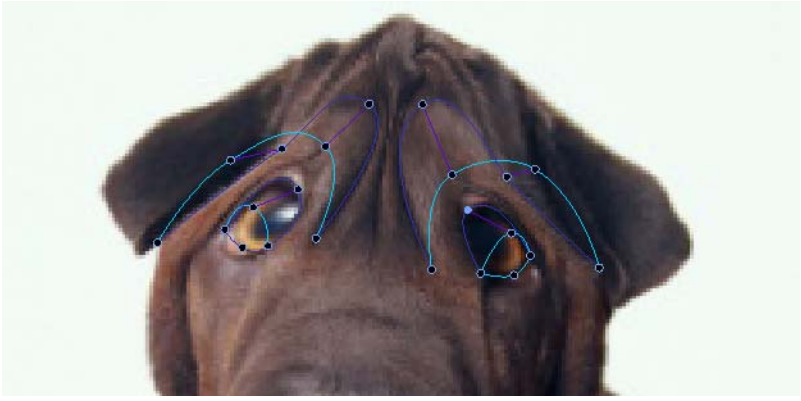
a Click the Connect Shapes button.

b Click a source shape that conforms to the actual position of the first feature you identified.

c Immediately click the corresponding target shape you created.

After this second click, the source/target shape pair is defined, the shape colors change, and a connectionName parameter appears in the Parameters tab. Because the overallDisplacement parameter defaults to 1, the effect is immediately seen (see "Connecting Source and Target Shapes" on page 840).

Once connected, source shapes become light blue, target shapes become dark blue, and the connection lines between them become purple. These colors can be customized, if necessary. For more information on customizing shape colors in the Viewer, see "Customizing Shape Controls" on page 843.

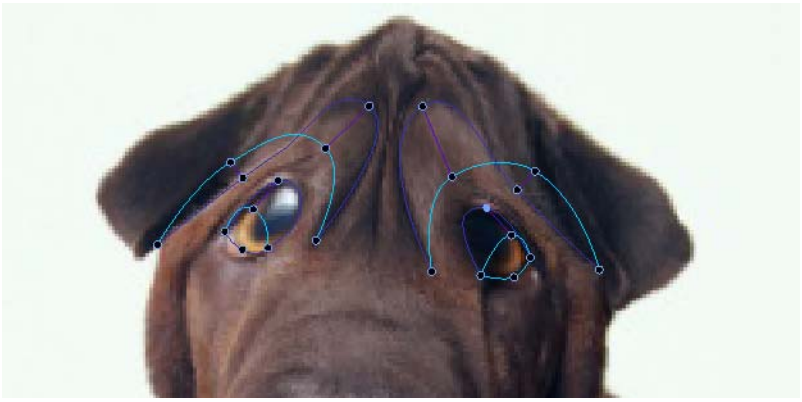


Once connected, source shapes are light blue and target shapes are dark blue.

6 If necessary, click the Edit Connections button, then drag in the Viewer to adjust the position of the connection lines running between the source and target shapes.

7 Drag the source and target connection points and slide them along the shape to change the angle of deformation in order to create the effect you need.

In this example, the connection lines are straightened in the eyes (see "Modifying Connection Lines" on page 841).



Final adjusted source/target shape pairs with modified connection points

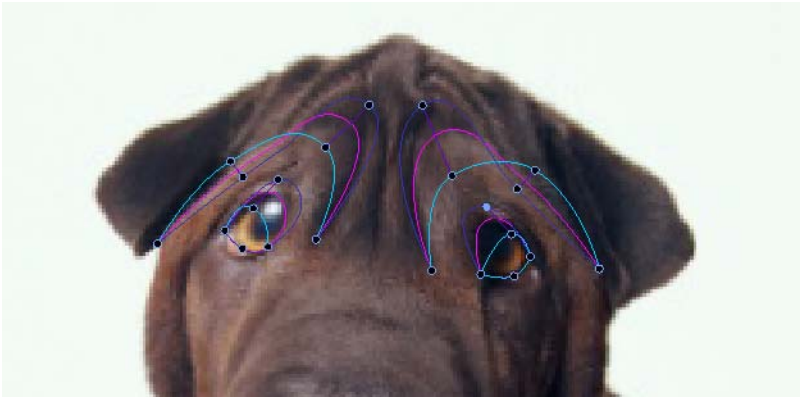
8 If necessary, adjust the amount of warp by modifying the overallDisplacement parameter in the Parameters tab.

You can also adjust the displacement of each source/target shape pair individually using the connectionDisplacement parameter in that pair's connectionName parameter.

A value of 0 in the overallDisplacement parameter results in an unwarped image. A value of 0.5 produces a warp that's halfway between the source and target shapes, and a value of 1 results in a warp that completely conforms to the target shape.

To see the actual warp effect, choose Warped Image from the Select Display Image pop-up menu in the Viewer shelf (or press the F11 key).

Note: In addition to viewing the actual warp effect, you can view the position of the displacement targets, as defined by the overallDisplacement and connectionDisplacement parameters, by turning on the Displaced Target Shape Visibility button in the Viewer shelf. These indicators are designed to help you see what the deformation will be without having to render the entire image. Displaced target shapes are pink by default.



Pink displaced target shapes indicate the value of the displacement parameters.

This example displays a characteristic of the warper—it works as if the image is made of a sheet of rubber and you're actually pushing the pixels of the image around, stretching the surrounding image. In the above image, the pixels of the eyebrow are moved up because they lie directly on the path of the source shape. You'll also notice that the right edge of the eyebrow appears to stretch back to the original position of the eyebrow. This is because the pixels surrounding the eyebrow are stretching to fill in the areas of the image where the eyebrow used to be. If the effect is not what you want, modify the source shape to redefine the area of the image being manipulated.

You'll notice that, in addition to the eye and eyebrow being warped, a significant area of the face surrounding the source/target shape pair is also affected, and the top of the head is pushed upward. To limit the warping effect to the region immediately surrounding the eyes and eyebrows, create one or more boundary shapes.

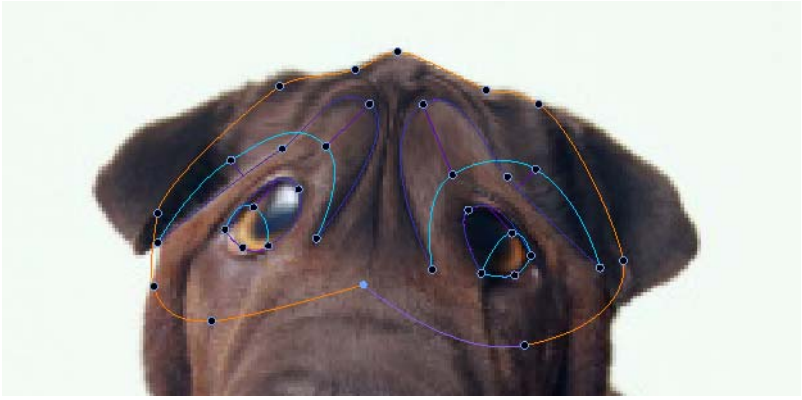
9 First, choose Source Image from the Select Display Image pop-up menu in the Viewer shelf. This allows you to draw your boundary shapes to match features in the original image.

10 Next, draw more shapes identifying the areas of the source image you want to lock down.

Boundary shapes can be either open, closed, or single-point shapes, depending on how much of the image you want to lock down. In this instance, you want to exclude the entire image from the warp effect except for the eye, eyebrow, and surrounding region, so a closed shape is drawn surrounding this area.

11 Right-click the shape you just created, then choose Set as Boundary Shape from the shortcut menu.

This effectively sets this shape to be both a source and target shape, which pins that area of the image down.



Boundary shapes are orange.

Now, you'll probably need to make adjustments to refine the effect you're trying to achieve. It will probably be helpful to use the Visibility and Lock buttons to assist you when manipulating the shapes, so that you don't accidentally adjust the wrong points when two shapes overlap. Use the controls in the Viewer shelf to change the visibility and locking of all shapes of a given type simultaneously, or set the visibility and locking of each shape individually in the Parameters tab.

Using the Parameters tab controls for each individual shape, you may rename the shape, enable it, toggle its visibility, lock the curve so it is visible but can't be modified, or delete it. Additional parameters also appear for each connection line and boundary shape definition you've set up.

12 To create an animated effect, keyframe the overallDisplacement parameter to animate every source/target shape pair simultaneously.

You can also individually animate the displacement caused by each source/target shape pair you've defined. To do so, open the connectionName subtree to reveal the connectionDisplacement parameter. Animating specific parameters can create a more organic-looking effect.

Before

After

Using the Morpher Node

The *Morpher* node blends two images together to create the effect of one subject changing shape to turn into another. The *Morpher* node does this by performing two warping operations, one on the source image to warp it into the shape of the target image, and another warping operation to warp the target image from the shape of the source back to its own shape. Once both warping operations match the shapes of the source and target images to one another, a built-in cross-fade dissolves from the first warp to the second, providing the illusion that the first image is changing into the second.

Tips For Successful Morphing

Successful morphs benefit from planning ahead during the shoot. Ideally, the positions of the source and target images match relatively well. If they need adjustment, resizing, or repositioning to help them match better, you can make these adjustments in your node tree prior to adding the *Morpher* node.

If the source and target subjects are moving, their movements should match one another so that the warping targets you set for both can line up properly. If the motions line up but the timing is off, you can select the offending clip's *FileIn* node and use the Timing tab parameters to remap its speed so that the motion lines up. For more information, see "Retiming" on page 117.

Because morphing warps images the same way the *Warper* node does, it is essential to isolate the subjects you're morphing prior to adding the *Morpher* node. This way, the background won't change as the source image morphs into the target, nor will the warp being applied to the subject of each image affect the background incorrectly.

Additional Controls and Parameters in the Morpher Node

Most of the *Morpher* node's controls are identical to those of the *Warper*. For instructions on how to use specific functions, consult the *Warper* node, above. The *Morpher* node does have some additional parameters and controls.

Additional Viewing Modes in the Viewer Shelf

In addition to the Source Image and Target Image options in the Set Display Image pop-up menu of the Viewer shelf, the *Morpher* provides five additional Viewer modes:

- *Morphed Image:* Shows the actual morph effect being created. This image is a combination of the source warped and target warped images being dissolved together. This is the end result of the *Morpher* node.
- *Source Warped Image:* Displays the warp effect being applied to the source image.
- *Target Warped Image:* Displays the warp effect being applied to the target image.
- *Dissolve Mask:* A grayscale image generated by the dissolve settings for each of the shapes. Because the dissolve settings for each individual connectionName parameter are linked to the overallDissolve parameter, this option displays a solid screen where:
  - Black represents an overallDissolve value of 0, showing only the source image.
  - White represents an overallDissolve value of 1, showing only the target image.

If you've animated the individual connection_Dissolve parameters (in the Connection_Name subtree):

- *Dissolved Image:* Displays the dissolve between the source and target images, without any warping being applied. Appears as a simple cross-dissolve.

Additional Parameters in the Morpher Node

This node displays the folowing additional controls in the Parameters tab.

overallDissolve

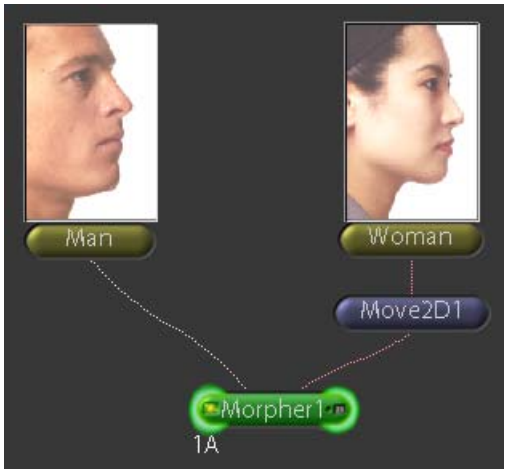
Controls whether the color of the morphed image is taken from the source image or the target image. 0 represents 100 percent source image, .5 results in a blend between both, and 1 represents 100 percent target image.

connectionDissolve

Each source/target shape pair has a corresponding connection_Name parameter. Nested within each connectionName parameter is a pair of connection_Displacement and connection_Dissolve parameters that allow you to independently adjust the displacement and dissolve of each part of the morph. By default, each connection_Dissolve parameter is linked to the overallDissolve parameter, so that they all animate together.

How to Morph Two Images

1 In the Node View, attach a *Morpher* node to two images.



This example creates the effect of the man's face turning into that of the woman. The image of the man is the source, connected to the morpher1.Source input. The transformed image of the woman is connected to morpher1.Target input.

Source image

Target image

If the images need to be manipulated to make them line up, do this first.

2 In this example, the image of the woman is transformed with a *Move2D* node, to position and rotate it to more approximately fit the image of the man.

This is essential to creating a smooth morphing effect.

If it is necessary to isolate the subject of the source and target images, you may want to insert *RotoShape* or keying nodes prior to the *Morpher* node.

3 Move the Time Bar playhead to the first frame of the clip where you want the morph effect to take place, then choose Source Image from the Select Display Image pop-up menu.

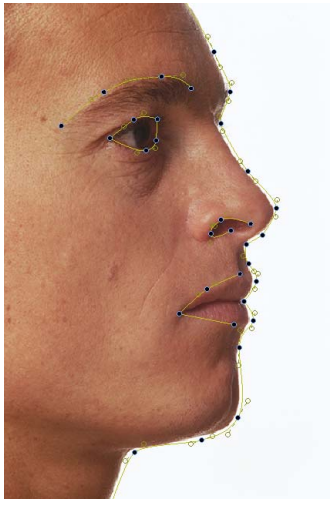
4 Click the Add Shapes button in the Viewer shelf, then draw shapes as necessary to match the features of the subject.

If necessary, animate your shapes to follow the animation.

Note: You can quickly jump to the source image by pressing F9.

The more features you identify with source shapes, the more detailed the morphing effect will be. You should remember that warping affects the entire region of the image surrounding each source/target shape pair. While it's important to create shapes for each significant feature of the image, you don't have to go overboard creating shapes for every tiny feature of the subject image, unless it will enhance the effect you're trying to achieve.

When picking features to manipulate, keep in mind that the source shapes you define are pushing and pulling the corresponding image features into the shape of the target. Pick features that have a direct path to similar features in the target image, if at all possible, to avoid unwanted artifacts in the image.



New unassigned shapes defining the source features

5 Once you've created all the source shapes you think you need, set the Viewer to display the target image (using the Select Display Image pop-up menu).

Note: You can quickly jump to the target image by pressing F10.

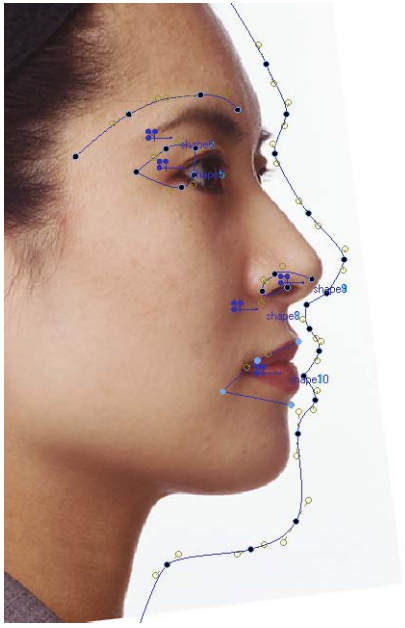
6 To create a set of target shapes to connect to the source shapes you created in step 4, do one of the following:

- The easiest method is to right-click each source shape, then choose Duplicate and Connect from the shortcut menu (or press Control-D or Command-D). Afterward, you can hide all your source shapes by turning off the Source Shape Visibility toggle in the Viewer shelf to avoid accidentally moving them while you adjust the target shapes to line up with the appropriate features of the target image.

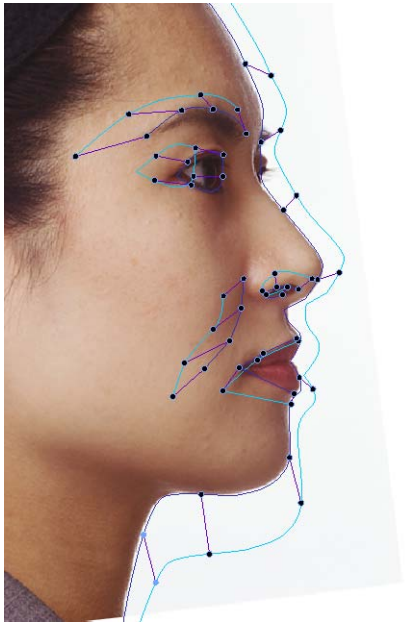
  Note: It may also help to turn off Enable/Disable Shape Transform Control in the Viewer shelf, to avoid accidentally dragging transform controls that overlap the shape you're trying to adjust.

- Manually draw more shapes over features in the target image that correspond to the features you identified in the source image. When you're done, you should make sure that you've drawn a target shape to correspond to every source shape.

7 Next, connect each source/target shape pair together using the Connect Shapes button in the Viewer shelf (see "Connecting Source and Target Shapes" on page 840).



Shapes immediately after using the Duplicate and Connect Shapes command



Target shapes have been adjusted to fit the target image. Connection lines have been added for additional adjustments.

When readjusting the target shapes you've created, the sheer number of shapes needed to create your morphing effect may make the Viewer a little crowded, making it difficult to adjust individual shapes. You may find it's easier if you hide every shape except the one you want to work on. You can hide all of the target shapes by right-clicking in the Viewer, then choosing Shape Visibility > Hide All Shapes from the shortcut menu. Afterwards, right-click again, then choose the name of the first target shape you want to edit from the shortcut menu. The shape reappears in the Viewer, ready for editing. Continue hiding and showing individual shapes as necessary until you've adjusted them all.

8 If necessary, animate the target shapes you've just created to match any motion in the target image.

9 Adjust the overallDisplacement parameter to control how much warp is applied to push the pixels from the source shapes to the target shapes you've defined.

Note: The overallDisplacement parameter operates on all source/target shape pairs in the node simultaneously.

The morphed image with overallDisplacement and overallDissolve of 0.5

10 To animate the morphing effect, keyframe the overallDisplacement parameter. To see the morphing effect in the Viewer as you adjust the overallDisplacement slider, you must choose Morphed Image in the Select Display Image pop-up menu in the Viewer shelf. You can also set the Viewer to display the morphed image by pressing F11.

To add a new keyframe, move the playhead to a frame where you want to make an adjustment, click the Autokey button for the overallDisplacement parameter in the Parameters tab, then adjust the overallDisplacement slider.

A value of 0 in both the overallDisplacement and overallDissolve parameters results in an unmorphed source image. A value of .5 produces a morph that's halfway between the source and target images, and a value of 1 results in the end of the morph—the final target image.

While you adjust these parameters, enable the Show Displaced Target Shapes button in the Viewer shelf to see the actual position of the displacement targets as defined by the overallDisplacement and connection_Displacement parameters. These indicators are designed to help you see what the deformation will look like, without having to render the entire image. Displaced target shapes are pink by default.

Note: As with the *Warper* node, you can adjust the individual displacement of each source/target shape pair using the connection_Displacement and connection_Dissolve parameters nested within each connection_Name parameter in the Parameters tab. Keyframing these parameters with separate timings creates a more sophisticated-looking effect than if you simply animated the overallDisplacement parameter.

11 If you want, you can keyframe the overallDissolve parameter separately from the overallDisplacement parameter to create different effects. Adjustments to the overallDissolve parameter control how the source image fades into the target image—this works exactly like a *Mix* node.

Note: By default, the overallDissolve parameter is linked to the overallDisplacement parameter, so keyframing one will automatically keyframe the other to the same value. Keyframing the overallDissolve parameter will break this link.

12 Test the resulting effect to see how well it works. If you see problems, toggle the Viewer between the source warped image and target warped image to see how successfully the source and target images are matching. Viewing each image independently makes it easier to spot unwanted artifacts stemming from poorly placed or inadequate numbers of source/target shape pairs.

If you see problems, either adjust the position and shape of the source and target shapes as necessary, or identify additional features to create source/target shape pairs in order to increase the amount of control you have over the effect.

Filters

28

The filter nodes in Shake not only enable simple image manipulation—they also provide numerous ways to modify alpha channel data, allowing you to create useful images for masking functions.

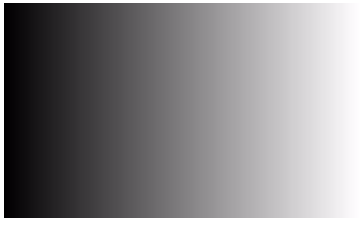
About Filters

While color corrections change the value of an individual pixel according to a mathematical equation (for example, *2, -.5, and so on), filters calculate the new value of a pixel by examining its neighbors, and passing it through what is called a *spatial filter*. Classic filter functions are blur, image sharpen, emboss, and median. You can also create your own unique filters, of any resolution, with the *Convolve* node. Spatial filters are also applied when a geometrically transformed image is resampled—for example, following a *Scale* or *Rotate* node.

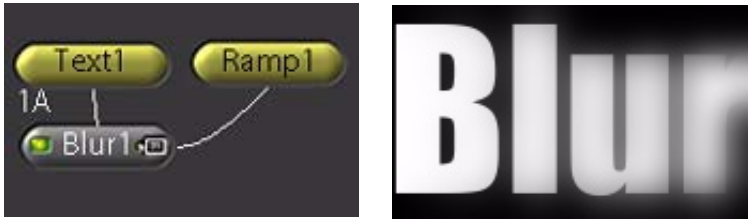
Masking Filters

When you use an image to control the amount of filtering, you should use the multi-input image-based filters such as *IBlur*, *ISharpen*, *IRBlur*, rather than simply masking the effect. Using image-based filter nodes instead of mask inputs will slow down processing, but the resulting effect will be ofs higher quality.

In the following example, a *Text* node is attached to a *Blur* node. A *Ramp* node attached to the *Blur* mask input acts as a control image to modify the blur effect.



The result—is merely a blend between sharp and blurred elements—is not very compelling. (Note that the *Ramp* node has a default alpha value of 1 for both ends; you should change the alpha1 value to 0.)



To get a better result, use the dedicated *IBlur* node instead, with the *Ramp* node as the second input image, rather than a mask input.

Filters Within Transform Nodes

Filter operations aren't limited simply to blurs, emboss effects, and other convolution operations assigned to filter nodes. Filter parameters are also available in many transform nodes.

Most filter and transform nodes allow you to use one of many different filtering processes to perform transforms, blurs, and convolve effects. Which filter you choose can make a big difference in the quality of the resulting effect, especially when animated.

To maximize the quality of scaling in Shake, the "default" filter setting in transform nodes actually switches between two different filters—the mitchell filter for nodes that increase image scale, and the sinc filter for nodes that decrease image scale. A panel of film professionals watched several versions of a shot that had been processed with different filters projected onto a cinema screen. They decided that the mitchell and sinc filters provided the best quality. Other filters, such as the box filter (the closest operation to what is more commonly referred to as "bilinear"), may give subjectively superior results in some cases (particularly with static images) but tend to handle high-frequency image data poorly.

To further maximize the quality of transforms, some nodes in Shake (such as *CornerPin*) let you use separate filter operations for horizontal and vertical transforms. Keep in mind that the default filter option uses mitchell for scaling up, and sinc for scaling down.

Applying Separate Filters to X and Y Dimensions

If a node does not already have separate filtering options for X and Y transforms (such as the *Resize* node), you can set up independent filtering for each dimension in two steps. For example, apply one *Resize* node, set the filter parameter to the desired filter method, then adjust the xSize parameter while leaving the ySize parameter untouched. Next, apply a second *Resize* node immediately after the first. Set a different filter method, then adjust its ySize parameter. Because both nodes concatenate, the result is computationally and qualitatively efficient.

Note: The subpixel parameter in the *Resize* node affects the way fractional resize is performed. If your resize value is not an integer (for example, 512 zooming to 1024 is an integer by a factor of 2; zooming 512 to 1023 is not an integer, because it is a factor of 1.998), you have subpixel sampling. For *Resize*, if the new width or height is not an integer (either because it was set that way, or because of a proxy scale), you have a choice to snap to the closest integer (0 = subpixel off) or generate transparent pixels for the last row and column (1 = subpixel on).

Filter	Description
box	Computationally fast, and gives a "boxy" look. Default size is 1 x 1.
default	By default, mitchell is used to resize up, and sinc to resize down.
dirac	Dirac and impulse are the same. Default size is 0 x 0.
gauss	Gaussian lacks in sharpness, but is good with ringing and aliasing. Default size is 5 x 5.
impulse	Fast but lower quality. Default size is 0 x 0.
lanczos	Similar to the sinc filter, but with less sharpness and ringing. Default size is 6 x 6.
mitchell	This is the default filter when scaling up. A good balance between sharpness and ringing, and so a good choice for scaling up. Default size is 4 x 4. This is also known as a high-quality Catmull-Rom filter.
quad	Like triangle, but more blur with fewer artifacts. Default size is 3 x 3.
sinc	This is the default filter when scaling down. Keeps small details when scaling down with good aliasing. Ringing problems make it a questionable choice for scaling up. Default size is 4 x 4. It can also deliver negative values, which can be interesting when working in float/channel bit depth.
triangle	Not highest quality, but fine for displaying a scaled image on your screen. Default size is 2 x 2.

The Filter Nodes

The following sections describe each filter node, and include parameters, defaults, and examples.

ApplyFilter

The *ApplyFilter* node applies a blur effect like the *Blur* node, but additionally allows you to choose separate filters for the X and Y dimensions. You can then scale the default base range (in pixels) of the predefined filters. For instance, if the default number of pixels sampled on either side of the base pixel is 3 pixels, an xScale of 2 increases that range to 6 pixels.

You can change the filter type in the much faster *Blur* node. The *ApplyFilter* node exists only to allow absolute compatibility with images generated by other software packages.

Note that dirac and impulse filters have no effect with *ApplyFilter*.

Parameters

This node displays the following controls in the Parameters tab:

xFilter, yFilter

See "Filters Within Transform Nodes" on page 862.

xScale, yScale

The amount of filtering in pixels.

spread

Tells Shake whether or not to apply the blur to areas outside of the frame. A button to the right of the parameter name lets you set the mode.

- 0 = Compute "In Frame Only."
- 1 = Compute "Outside Frame."

Because of the Infinite Workspace, it is sometimes handy to compute outside of the frame as well, for example, if the *Blur* is placed after a *Scale* command. Note that if nothing is outside of the frame (black), you see a black edge.

Blur

The *Blur* node blurs the image. This is a Gaussian blur (by default), but you can change the filter for both X and Y. Use this node instead of the similar, but slower, *ApplyFilter* node.

Shake's *Blur* is one of the few nodes that can deactivate the Infinite Workspace —its "spread" parameter gives you the choice of blurring pixels *inside* or *outside* of the image boundaries. If your final image appears clipped and you aren't sure why (for example, you haven't applied any *Crop* commands), go back and check the *Blur* node spread parameter. Toggle the spread parameters to Outside Frame (1), and the clipping should disappear.

For more information on the Infinite Workspace, see "Taking Advantage of the Infinite Workspace" on page 405.

Parameters

This node displays the following controls in the Parameters tab:

xPixels, yPixels
The amount of blur as described in pixels, for example, entering a value 200 blurs 200 pixels to either side of the current pixel. By default, yPixels is linked to xPixels.

spread
Tells Shake whether or not to consider outside of the frame. A button to the right of the parameter name lets you set the mode.

- 0 = Compute "In Frame Only."
- 1 = Compute "Outside Frame."

Because of the Infinite Workspace, it is sometimes handy to compute outside of the frame as well, for example, if the *Blur* is placed after a *Scale* command. Note that if nothing is outside of the frame (black), you see a black edge.

xFilter, yFilter
Lets you pick which method Shake uses to transform the image. For more information, see "Filters Within Transform Nodes" on page 862.

channels
Lets you set which channels Shake should blur. You can choose one or all of the red, green, blue, or alpha channels. The default is "rgba."

Convolve

The *Convolve* node allows you to define your own custom filter using a convolution kernel. Standard filters are available in your *include/nreal.h* file, and appear in the kernel pop-up menu in the Parameters tab. The included kernels define the following filters:

- *blur3x3:* 3 x 3-pixel blur
- *blur5x5:* 5 x 5-pixel blur
- *sharpen*
- *edge3x3:* 3 x 3-pixel edge detection
- *edge5x5:* 5 x 5-pixel edge detection
- *laplace:* edge detection
- *smoothedge:* another type of edge detection

- *sobelH:* horizontal embossing
- *sobelV:* veritcal embossing
- *BabuV:* another vertical edge detection
- *BabuH:* another horizontal edge detection

You can use these convolution matrixes as is, or as models to create your own matrixes.

Creating Custom Convolution Kernels

Convolution kernels consist of properly formatted data, which is used by the *Convolve* node to produce the desired image processing effect. This data is included by default in the *include/nreal.h* file, but may also be placed in other files in the same way you would add a macro. For example, you could add convolution kernels to *include/startup/my_file.h*.

A convolution kernel must have the following information. Each parameter should be followed by a comma:

- The kernel begins with the following declaration:

  ```
  DefKernel (
  ```

- The first parameter is the kernel's name, enclosed in quotes.
- The second parameter is the size of the kernel matrix—in this case, 5 x 5.
- The third parameter is the gain factor. Each number is multiplied by this number, so 1 results in no change.
- The fourth parameter is the offset, which is added to the result before it is clamped/quantized. 0 results in no offset.
- Next, the kernel matrix is entered as a comma-delimited list. In the case of a 5 x 5 matrix, this list would take the form of five lines of five comma-delimited values.
- The final line should be the end parenthesis and a semicolon, to end the declaration.

Here's an example of a properly formatted kernel:

```
DefKernel(
  “edge5x5”,
  5, 5,
  1,
  0,
  -1, -1, -1, -1, -1,
  -1, -1, -1, -1, -1,
  -1, -1, 24, -1, -1,
  -1, -1, -1, -1, -1,
  -1, -1, -1, -1, -1
);
```

Parameters

This node displays the following controls in the Parameters tab:

channels

Lets you set which channels Shake should blur. You can choose one or all of the red, green, blue, or alpha channels. The default is "rgba."

kernel

A pop-up menu that allows you to select any of the kernels included in the *include/nreal.h* file, or wherever else you may have added convolution kernels of your own.

percent

A slider that lets you mix the modified image and the original image together to produce a blend of both. By default, this parameter is set to 100 percent, resulting in 100 percent of the modified image being output.

absolute

Some filters return negative values. When absolute is enabled, these values are inverted to produce positive values.

Defocus

The *Defocus* blur node is a more accurate model of the blurring that occurs through an out-of-focus real-world camera lens. It flares out the high points, resulting in a circular, hexagonal, or octagonal shape around the highlights.

Original image

Real camera defocus

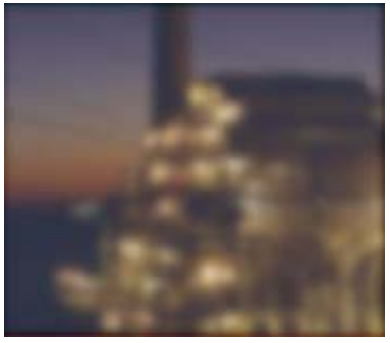


Image blurred with normal Gaussian blur in Shake

Image blurred with *Defocus* node in Shake

Parameters

This node displays the following controls in the Parameters tab:

xPixels, yPixels

Defines the kernel size for the defocus, in pixels, which produces the general size of the defocused flaring in the image. Values lower than 3 have no effect. Progressively higher values are more processor-intensive to render.

channels

Lets you set which channels Shake should defocus. You can choose one or all of the red, green, blue, or alpha channels. The default is "rgba."

percent
A slider that lets you mix the modified image and the original image together to produce a blend of both. By default, this parameter is set to 100 percent, resulting in 100 percent of the modified image being output.

shape
A pop-up menu that lets you choose the shape of the flaring in the resulting image. The fast modes give you low quality but process quickly. The circle, square, hexagon, and octagon give you a superior look, but are significantly more processor-intensive to render. The options are:

- fast gaussian
- fast box
- circle
- square
- hexagon
- octagon

boostPoint
The image value where the superwhite boosting starts. If boostPoint is set to .75, RGB values above .75 are boosted to increase flaring effects. A high value generally decreases flare areas, since fewer candidate pixels are flared. By default this parameter is set to .95.

superWhite
Max value to boost image to. A value of 1 in the original will be boosted to this value. By boosting this parameter, you increase the brightness of the flare area. Values around 50 yield a very strong boost.

DilateErode

The *DilateErode* node isolates each channel and cuts or adds pixels to the edge of that channel. For example, to chew into your mask, set your channels to "a," then set the xPixels and yPixels value to -1.

By default, this node only affects whole pixels. Subpixel values are ignored, even when they are set within the pixels parameters.To dilate or erode at the subpixel level, turn on the "soften" button. Note that the soften parameter *really* slows the function. If you use the soften feature, use low values for xPixels and yPixels.

To avoid affecting an image when using *DilateErode* to modify alpha channel data, enter "a" as the channel, then apply a Color–*MMult* node to multiply the RGB by the modified alpha.

Parameters

This node displays the following controls in the Parameters tab:

channels
Lets you set which channels Shake should blur. You can choose one or all of the red, green, blue, or alpha channels. The default is "rgba."

xPixels, yPixels
The number of pixels added (dilate) or taken from (erode) an edge. Positive values add to the edge; negative values eat away at the edge.

borders
This parameter determines whether Shake considers or ignores the border pixels at the edge of the image.

soften
This parameter lets you turn softening on or off. When this parameter is disabled, *DilateErode* affects only whole pixels. When enabled, *DilateErode* can dilate or erode at a subpixel level.

sharpness
The sharpness factor for the softening. A value of 0 gives a smooth gradation, whereas 2 gives you a sharp cutoff.

EdgeDetect

The *EdgeDetect* node is great for pulling out and massaging edges. You can control what is detected, the strength of the edge, and the ability to expand or soften the edge.

Parameters

This node displays the following controls in the Parameters tab:

strength
Applies a gain to the filter. Lower strength values eliminate detail in favor of the strongest outlines, whereas higher strength values reveal more details. The slider lets you choose a value from 0 to 10. Practically speaking, 10 is the highest useful value, but you can enter higher values into the number field if you need to.

threshold
This parameter lets you further reduce the amount of detail in the resulting image. Pixels below the threshold turn black. The range is 0 to 1.

binaryOutput
When binaryOutput is turned on, all pixels falling under the threshold parameter are made black, and all pixels falling above the threshold parameter are made white. The resulting image is only black and white, with no grayscale values.

directionFilter
Enables an effect similar to *Emboss*.

directionFilterangle
This parameter changes the lighting angle when the directionFilter parameter is turned on.

despeckle
Similar to a median filter, this parameter removes isolated pixels up to the despeckle radius (in pixels), and can be useful for eliminating noise from the resulting image.

xBlur, yBlur
Blurs the resulting image after the edge detection has been performed. By default, the yBlur parameter is linked to the xBlur.

xDilateErode, yDilateErode
Applies the *DilateErode* effect to the first filter pass. By default, the yDilateErode parameter is linked to the xDilateErode.

rgbEdges
Inherits color from the input image instead of just black and white, and applies it to the final output image.

addEdgesToImage
Mixes the resulting image from the edge detection node to the original input image, to create a blend of both.

method
A pop-up menu that lets you select which edge detection method to use. Your choices are:

- Sobel
- Babu
- Fast

Note: Babu is an algorithm that is extremely slow and cannot be used on large (2K or larger) plates. It is maintained for compatibility purposes.

babuSteps
The steps performed when using the Babu filter. More steps result in a higher quality image, but are more processor-intensive to render.

bytes
Three buttons let you choose the output bit depth. You can choose among 8-bit, 16-bit, and float.

Emboss

With the *Emboss* node, you control the gain and light direction to simulate a raised texture over an image.

Note: The *Emboss* node converts your image to a BWA image (since there is no color information).

If you use extreme gain, you may start to see terracing on your image. To correct this, insert a *Bytes* node before the *Emboss* node, and boost your image to 2 bytes per channel. You can get interesting patterns with a *Bytes* node set to 2 bytes, followed by a *Blur* node, and then the *Emboss* node.

The elevation is set to 30 because it makes the median gray value 0.5.

Parameters

This node displays the following controls in the Parameters tab:

gain
The amount of emboss. Higher values result in a more pronounced effect.

azimuth
The apparent direction from which light is shining. 0 and 360 simulate light shining from the right side of the image, 90 is from the top, and so on.

elevation
This is the "height" of the embossed image. 0 is parallel to the image; 90 is the same axis as a line from your nose to the image.

FilmGrain

Use the *FilmGrain* node to apply grain that corresponds to real film grain to an element. Grain is typically added to still or CG images so the images more closely match the inherent noisiness of film plates.

You can choose to apply a preset film stock, sample grain from an existing image, or create your own grain by adjusting the sliders.

> *Warning:* You cannot clone a *FilmGrain* node in the Node View using the Paste Linked command.

To sample grain from an image:

1. Attach a Filter–*FilmGrain* node to the element to which you want to add grain.
2. Ensure that you are not in Proxy mode.
3. Load the image to be sampled into the Viewer.

   Note: This should not be the image created by the *FilmGrain* node itself but, rather, one generated prior to it in the node tree.

4 Ensure that the *FilmGrain* parameters are still active.

5 In the Viewer, drag to create boxes in the areas you want to sample.

Note: The sampled areas should be very flat without detail that may disrupt the grain analysis. Small elements are perceived as grain detail, so the best sample areas are featureless walls, exposure cards, and so on.

You can sample as many areas as you want.

6 To undo a sample, do one of the following:

- To undo a box drawing, click the Undo Last Region button in the Viewer shelf.

- To remove all boxes and start over, click the Reset the Regions button in the Viewer shelf.

7 Once the boxes are set, click the Analyze Grain button in the Viewer shelf.

The parameters in the *FilmGrain* node are set to match the plate.

Parameters

This node displays the following controls in the Parameters tab:

intensity
The intensity of the grain. Values are between 0 and 2.

grainSize
Size of the grain, between 0 and 2.

aspectRatio
This parameter inherits the current value of the defaultAspect global parameter. If you're working on a nonsquare pixel or anamorphic image, it is important that this value is set correctly to avoid unwanted distortion in the image.

seed
The random generation seed. Set this to a constant value to freeze the grain. By default, this is set to the *time* expression. When Shake generates a random pattern of values, you need to make sure for purposes of compositing that you can recreate the same random pattern a second time. In other words, you want to be able to create different random patterns, evaluating each one until you find the one that works best, but then you don't want that particular random pattern to change again.

Shake uses the seed value as the basis for generating a random value. Using the same seed value results in the same random value being generated, so that your image doesn't change every time you re-render. Use a single value for a static result, or use the expression "time" to create a pattern of random values that changes over time.

For more information on using random numbers in expressions, see "Reference Tables for Functions, Variables, and Expressions" on page 941.

filmStock
A pop-up menu that allows you to select from one of many preset film stocks. You can also apply custom values by selecting Custom. Accepted inputs are:

- *Custom*
- *Eastman 5245*
- *Eastman 5247*
- *Eastman 5248ac*
- *Eastman 5248nc*
- *Eastman 5287*
- *Eastman 5293ac*
- *Eastman 5293nc*
- *Eastman 5296*
- *Eastman 5298*
- *Kodak 5246ac*
- *Kodak 5246nc*
- *Kodak 5274ac*
- *Kodak 5274nc*
- *Kodak 5277*
- *Kodak 5279*

Note: "ac" indicates a stock with aperture correction. "nc" indicates no aperture correction.

stdDev
This value is multiplied by the amount parameter. A higher value indicates more variation in the grain, making it more apparent. This parameter defaults to 0.5.

The rStdDev, gStdDev, and bStdDev subparameters let you control the variation within each individual color channel. By default, these three parameters are locked together.

softness
Blurs the edges of the grain that's introduced. This parameter defaults to 1.2. The rSoftness, gSoftness, and bSoftness subparameters let you control the grain softness of each individual color channel.

filmResponse
Determines the extent to which the grain inherits its color from the input image instead of simply black and white. Progressively higher positive values result in the grain matching the color of the input image more closely. Progressively higher negative values result in the color of the grain becoming somewhat muted. This parameter defaults to 0. The rFilmResponse, gFilmResponse, and bFilmResponse subparameters let you customize the filmResponse of each individual color channel.

colorCorr
This parameter specifies the apparent saturation of the grain in units that represent the color-correlation value measured by statistical analysis of a particular film sample.

The value represents how closely the grain patterns in each channel overlap. This means that negative color-correlation values decrease the amount of overlap, which increases the apparent saturation of the grain, while positive values decrease the apparent saturation.

Grain

The *Grain* node adds grain to an image. It is used to simulate film grain for 3D-rendered elements. The *Grain* node is not as accurate as the newer *FilmGrain* node.

This node gives you complete per-channel control over grain size, density, softness, and so on. The controls are explained below, with visual examples after the parameter list.

Note: If you have an RGB channel image, there is no grain if obeyAlpha is enabled, as there is an alpha value of 0.

In general, when there is a parameter followed by the same parameter on a per-channel basis, the first one acts as a multiplier on the channel parameters. For example, a density of .5 multiplies rDensity, gDensity, and bDensity by 0.5.

Parameters

This node displays the following controls in the Parameters tab:

size
The overall size of the grain. "size" is a multiplier on the rSize, gSize, and bSize subparameters. You can have sizes less than 1.

density
The density of the grain. 1 is maximum density. "density" is a multiplier on rDensity, gDensity, and bDensity.

seed
The random seed for the grain. When Shake generates a random pattern of values, you need to make sure for purposes of compositing that you can recreate the same random pattern a second time. In other words, you want to be able to create different random patterns, evaluating each one until you find the one that works best, but then you don't want that particular random pattern to change again.

Shake uses the seed value as the basis for generating a random value. Using the same seed value results in the same random value being generated, so that your image doesn't change every time you re-render. Use a single value for a static result, or use the expression "time" to create a pattern of random values that changes over time.

For more information on using random numbers in expressions, see "Reference Tables for Functions, Variables, and Expressions" on page 941.

filmResponse
The valid range is theoretically infinite, but practically is -1 and 1. -1 is typical of standard film, with grain applied to the entire range except the brightest whites, and black is the most affected. 1 is the inverse of that, withholding grain from the darks, with most grain on the whites. Default is -1.

lGain, rGain, gGain, bGain
The overall intensity of the grain. lGain (luminance) applies to all three channels equally, while rGain, gGain, and bGain apply only to each particular color channel.

Each of these parameters has subparameters for adjusting its bias, LowOffset, and HighOffset values.

- bias: Shifts the grain level higher or lower in intensity.
- LowOffset: From the midpoint of the Gain, squeezes the darker grain areas. This is a useful adjustment when the highlights are good and you want to modify only the level of the darker grain.
- HighOffset: From the midpoint of the Gain, squeezes the brighter grain areas.

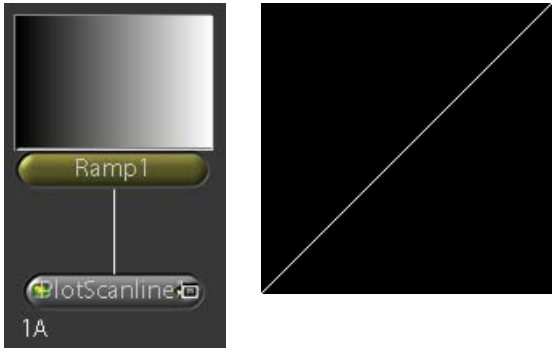
aspect
The grain aspect ratio. An aspect of 2 stretches it twice as wide.

softness
A value above 0 softens the grain. A value below 0 sharpens the grain. The global softness is an additive factor, so it is added to the values of the per-channel parameters.

obeyAlpha
When enabled, grain is applied to the image through its alpha channel. When disabled, grain is applied to the entire image, and the alpha is ignored. Useful for applying grain to premultiplied CG images without contaminating the background black.

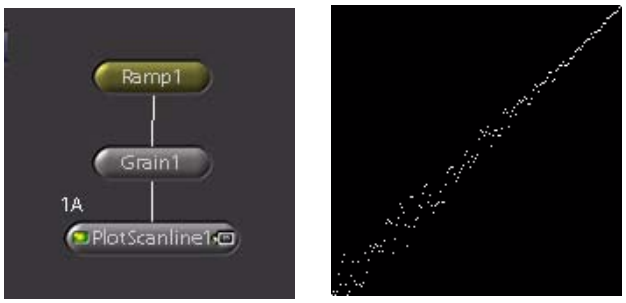
Grain Example

In the following example, the first node tree consists of a *Ramp* node and a *PlotScanline* node. The *PlotScanline* node is added to analyze the image.



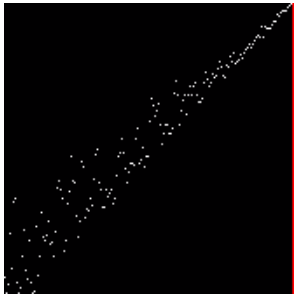
Because the ramp is black to white, a linear line appears in the plot scanline from left to right when no grain is applied.

When a *Grain* node is inserted between the *Ramp* node and the *PlotScanline* node, noise is introduced that disturbs the line. There is more noise (grain) near the lower, dark area of the plot scanline. This is due to the filmResponse of -1, which concentrates grain on the lower areas.

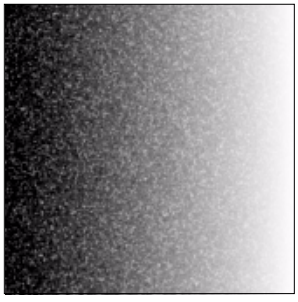


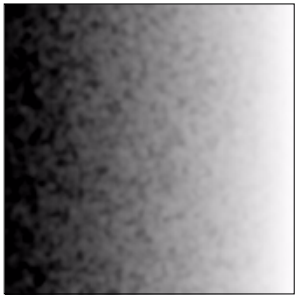
The next image is the result of increasing the lGain (or rGain, gGain, and bGain on a per-channel basis), and increasing the range of the grain. This results in making the diagonal line both lighter and darker simultaneously.

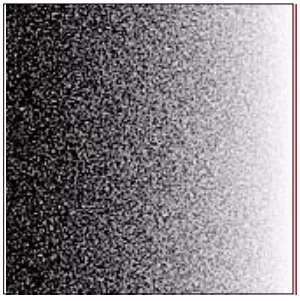
The following two images show the result of modifying the Bias and Gain. In the first image, the rBias is lowered (looking only at the red channel) to -.5. The grain shifts downward from the diagonal line, making it darker. This affects both the lighter grain (the dots above the original diagonal line) and the darker grain (the dots below the original diagonal line). To adjust only the lighter or darker grains, use the offset sliders. The second image shows a rLowOffset of .5, squeezing only the darker grains, leaving only lighter grains.



In the following images, the left image shows the standard softness. The middle image has a softness of 10, and the right image has a softness of -10.



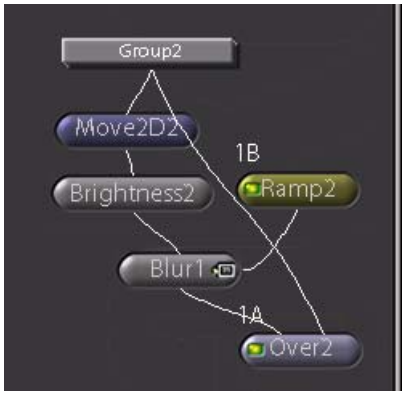




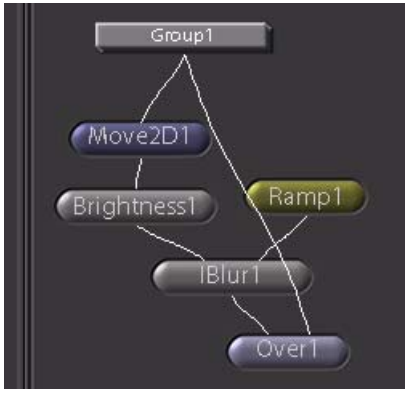
IBlur

The *IBlur* node blurs the image, with the amount of blur set by a second control image. Maximum blur occurs in the white areas of the second image, and no blur occurs in the black areas.

In the following example, the first node tree uses a *Ramp* that is approximately one-half of the height of the image as a mask for a blur. Some bad blending occurs in the lower portion of the image.



In the second node tree, a *Ramp* is used as a second input into *IBlur*, and the result is a nice blend of the blurred and non-blurred areas.



The above script is saved as *doc/html/cook/iblur_example.shk*.

Note: The *IBlur* node is much slower than the normal *Blur* node.

Parameters

This node displays the following controls in the Parameters tab:

xPixels, yPixels

The amount of blur as described in pixels, for example, 200 blurs 200 pixels to either side of the current pixel. By default, yPixels is linked to xPixels.

spread

Tells Shake whether or not to consider areas outside of the frame. A button to the right of the parameter name lets you set the mode.

- 0 = Compute "In Frame Only."
- 1 = Compute "Outside Frame."

Because of the Infinite Workspace, it is sometimes handy to compute outside of the frame as well, for example, if the *Blur* is placed after a *Scale* command. Note that if nothing is outside of the frame (black), you see a black edge.

xFilter, yFilter

Lets you pick which method Shake uses to transform the image. For more information, see "Filters Within Transform Nodes" on page 862.

steps

The amount of steps. The intensity of the control image is divided up X amount of zones, with X equal to steps.

stepBlend

Controls the blending between the amount of regions (see below). If you set this parameter to 0, each step has a constant blur value. If the setting is 1, there is a continuous blend between the different regions.

controlChannel

The channel of the controlling image to use to control the amount of blur.

channels

The channels of the input image to blur.

invert

Inverts the controlChannel.

IDefocus

The *IDefocus* node is the *Defocus* node with a second image input to control the size of the defocused flaring. For more information, see "*Defocus*" on page 868.

xPixels, yPixels

Defines the kernel size for the defocus effect, in pixels, which produces the general size of the defocused flaring in the image. Values lower than 3 have no effect. Progressively higher values are more processor-intensive to render.

channels
Lets you set which channels Shake should blur. You can choose one or all of the red, green, blue, or alpha channels. The default is "rgba."

percent
A slider that lets you mix the modified image and the original image together to produce a blend of both. By default, this parameter is set to 100 percent, resulting in 100 percent of the modified image being output.

shape
A pop-up menu that lets you choose the shape of the flaring in the resulting image. The fast modes give you low quality but process quickly. The circle, square, hexagon, and octagon give you a superior look, but are significantly more processor-intensive to render. The options are:

- fast gaussian
- fast box
- circle
- square
- hexagon
- octagon

boostPoint
The image value where the superwhite boosting starts. If boostPoint is set to .75, RGB values above .75 are boosted to increase flaring effects. A high value generally decreases flare areas, since fewer candidate pixels are flared. By default this is set to .95.

superWhite
Max value to boost image to. A value of 1 in the original will be boosted to this value. By boosting this value, you increase the brightness of the flare area. Values around 50 yield a very strong boost.

steps
The amount of steps. The intensity of the control image is divided up X amount of zones, with X equal to steps.

stepBlend
Controls the blending between the amount of regions (see below). If you put this at 0, each step has a constant blur value. If this is 1, there is a continuous blend between the different regions.

controlChannel
The channel of the controlling image to use to control the amount of blur.

channels
The channels of the input image to defocus.

invert
Inverts the controlChannel.

IDilateErode

The *IDilateErode* node isolates each channel and cuts or adds pixels to the edge of that channel. For example, to chew into your mask, set your channels to "a," and then set the xPixels and yPixels values to -1. By default, you work on whole pixels. To switch to subpixel chewing, enable "soften." Note that the soften parameter *really* slows the node. If you use the soften feature, use low values for xPixels and yPixels.

Parameters

This node displays the following controls in the Parameters tab:

channels
Lets you set which channels Shake should blur. You can choose one or all of the red, green, blue, or alpha channels. The default is "rgba."

xPixels, yPixels
The number of pixels added or taken from an edge. Positive values add to the edge; negative values eat away at the edge.

borders
This parameter determines whether Shake considers or ignores the border pixels at the edge of the image.

soften
This parameter lets you turn softening on or off, or affects the subpixel. If enabled, this node becomes considerably more processor-intensive with high xPixel and yPixel values.

sharpness
The sharpness factor for the softening. A value of 0 gives a smooth gradation, whereas 2 gives you a sharp cutoff.

steps
The amount of steps. The intensity of the control image is divided up X amount of zones, with X equal to steps.

stepBlend
Controls the blending between the amount of regions (see below). If you set this parameter to 0, each step has a constant value. If the setting is 1, there is a continuous blend between the different regions.

controlChannel
The channel of the controlling image to use to control the amount of the effect.

invert
Inverts the controlChannel.

IRBlur

The *IRBlur* node is an image-based version of the *RBlur* node, using the alpha mask of a second image (by default) to control the amount of radial blurring on an image. This is useful for faking motion blur effects.

In this example, the foreground objects (cubes) are rendered on a beach in a 3D software package with the depth information. The 3D image is composited over a background photo of the beach. The *DepthKey* node was used to extract a matte of the depth supplied by the 3D render. The matte is then fed into the second image of the *IRBlur* to give a zooming effect.

Parameters

This node displays the following controls in the Parameters tab:

xCenter, yCenter
The center point of the blur. By default, these parameters are set to width/2, height/2.

iRadius
The distance from the center that contains the blur sample area.

oRadius
The outer edge for the blur area.

aspectRatio
This parameter inherits the current value of the defaultAspect global parameter. If you're working on a nonsquare pixel or anamorphic image, it is important that this value is set correctly to avoid unwanted distortion in the image.

damp
A gamma value on the blur.

amplitude
The overall amount of blur. This number can also be negative, to blur away from the viewer, rather than toward the viewer.

blurQuality
The amount of samples. A quality of 1, the maximum, is 64 samples.

mirror
Considers points beyond the center area if your amplitude is high enough when enabled.

steps
The number of steps. The intensity of the control image is divided up X amount of zones, with X equal to steps.

stepBlend
Controls the blending between the number of regions (see below). If you put this at 0, each step has a constant blur value. If this is 1, there is a continuous blend between the different regions.

controlChannel
The channel of the controlling image to use to control the amount of blur.

channels
The channels of the input image to blur.

invert
Inverts the controlChannel.

ISharpen

The *ISharpen* node is identical to the *Sharpen* node, except it uses a second control image to set the amount of sharpening across the first image. Un-sharp masking is used for the sharpening filter. This process blurs the image slightly, takes the difference between the blurred result and the input image, and adds that back over the input image. High values of x and yPixels (for example, greater than 3 percent of the image size) return undesirable results. You can also use the sharpen filter in the *Convolve* node, but ringing may result.

For an example of this process, see "*IBlur*" on page 879.

Parameters

This node displays the following controls in the Parameters tab:

percent
The amount of blend between the non-sharpened image (0) and the sharpened image (100).

xPixels, yPixels
The amount of sharpening as described in pixels.

steps
The number of steps. The intensity of the control image is divided up X amount of zones, with X equal to steps.

stepBlend
Controls the blending between the amount of regions (see below). If you set this parameter to 0, each step has a constant blur value. If the setting is 1, there is a continuous blend between the different regions.

controlChannel
The channel of the controlling image to use to control the amount of blur.

channels
The channels of the input image to sharpen.

invert
Inverts the controlChannel.

Median

The *Median* node applies a 3 x 3 median filter. It is good for the removal of individual dots and noise.

Parameters

This node displays the following controls in the Parameters tab:

channels
Lets you set which channels the Median filter should affect. You can choose one or all of the red, green, blue, or alpha channels. The default is "rgba."

threshold
The change is allowed if the change is above or below this threshold value.

thresholdType
Defines whether the result is below (0) or above (1) the threshold to allow the change.

PercentBlur

The *PercentBlur* node blurs the image according to a percentage factor, rather than a pixel size. Therefore, 100 percent blurs with the entire image in the blur calculation.

Parameters

This node displays the following controls in the Parameters tab:

xPercent, yPercent
Percent of the image taken into consideration for the blur. 100 = the entire image, or a flat color. Default is 0.

spread
Tells Shake whether or not to consider outside of the frame. A button to the right of the parameter name lets you set the mode.

- 0 = Compute "In Frame Only."
- 1 = Compute "Outside Frame."

Because of the Infinite Workspace, it is sometimes handy to compute outside of the frame as well, for example, if the *Blur* is placed after a *Scale* command. Note that if nothing is outside of the frame (black), you see a black edge.

channels
Lets you set which channels Shake should blur. You can choose one or all of the red, green, blue, or alpha channels. The default is "rgba."

Pixelize

The *Pixelize* node filters the images into square blocks, giving a mosaic look by averaging all of the pixels within the block. It produces an effect similar to the *Blur* node with the xFilter and yFilter parameters both set to impulse.

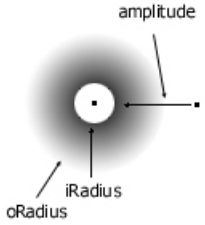
Parameters

This node displays the following control in the Parameters tab:

xPixels, yPixels
The block size, in pixels, to be filtered.

RBlur

The *RBlur* node blurs the image radially from a center point, creating flare effects. Although this is a processor-intensive process, it is extremely accurate. The entire image is blurred, based on the range from the inner radius to the outer radius. A pixel outside of the oRadius distance is blurred by the oRadius amount. A pixel inside of the iRadius amount is not blurred. A pixel between iRadius and oRadius is blurred by a percentage of its position between the two Radius parameters. You can also use the mirror parameter to blur beyond the center point, as well as use a negative amplitude to sample pixels away from the center rather than toward the center.



A point outside of the oRadius is blurred by the amount of oRadius. This is modified by amplitude. If amplitude is negative, the blur is calculated outwards. If mirror is turned on, the blur will extend past the center point.

Parameters

This node displays the following controls in the Parameters tab:

xCenter, yCenter
The center point of the blur. By default, these parameters are set to width/2, height/2.

iRadius
The distance from the center that contains the blur sample area.

oRadius
The outer edge for the blur area.

aspectRatio
This parameter inherits the current value of the defaultAspect global parameter. If you're working on a nonsquare pixel or anamorphic image, it is important that this value is set correctly to avoid unwanted distortion in the image.

damp
A gamma value on the blur.

amplitude
The overall amount of blur. This number can also be negative, to blur away from the viewer, rather than toward the viewer.

quality
The amount of samples. A quality of 1, the maximum, is 64 samples.

mirror
Considers points beyond the center area if your amplitude is high enough when enabled.

seed
When Shake generates a random pattern of values, you need to make sure for purposes of compositing that you can recreate the same random pattern a second time. In other words, you want to be able to create different random patterns, evaluating each one until you find the one that works best, but then you don't want that particular random pattern to change again.

Shake uses the seed value as the basis for generating a random value. Using the same seed value results in the same random value being generated, so that your image doesn't change every time you re-render. Use a single value for a static result, or use the expression "time" to create a pattern of random values that changes over time.

For more information on using random numbers in expressions, see "Reference Tables for Functions, Variables, and Expressions" on page 941.

Sharpen

Un-sharp masking is used for the sharpening filter. This process blurs the image slightly, takes the difference between the blurred result and the input image, and adds that back over the input image. High values of x and yPixels (for example, greater than 3 percent of the image size) return undesirable results. You can also use the sharpen filter in the *Convolve* node, but ringing may result.

Parameters

This node displays the following controls in the Parameters tab:

percent

The amount of mixing between the sharpened and the original image. With a value of 100, none of the original image is present.

xPixels, yPixels

The amount of sharpening as described in pixels.

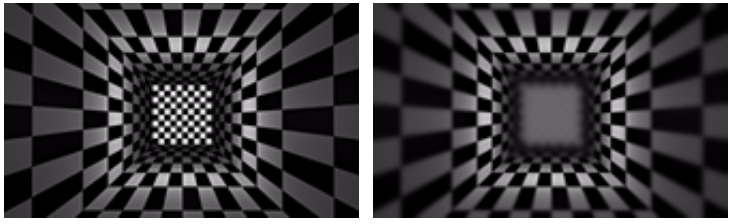
channels

The channels of the input image to sharpen.

ZBlur

The *ZBlur* node is a dedicated version of *IBlur* used to simulate depth-of-field blurring based on the Z channel. You can set a center point of focus (focusCenter) and the range of drop-off to the maximum amount of blur. If you do not have a Z channel, you can copy a channel from another image. Therefore, if you are working in 8 bit or 16 bit, your ranges are between 0 and 1. For example, your focusCenter could not be set to 10, but is somewhere between 0 and 1.

The *ZBlur* node works best with gradual changes of Z, for example, looking down a hallway, represented in the following images.



In cases where foreground objects overlap blurred background objects, ringing results on the background. It is recommended that you separate the foreground from the background into two separate elements. For example, do two renders if you are generating elements in 3D.

The following example is from a 3D render. Ringing appears around the background letters of the text when normally passed into a *ZBlur*. Instead, separate the circle into foreground and background elements with the 3D render. These are then blurred and composited in Shake.

Parameters

This node displays the following controls in the Parameters tab:

amount
The maximum amount of blur.

near
The value of distance toward the camera at which maximum blur occurs.

far
The value of distance away from the camera at which maximum blur occurs.

focusCenter
The distance from the camera at which there is no blur. By default this is set to (far-near)/2+near.

focusRange
The distance away from the focusCenter, both toward and away from the camera, that remains unblurred.

steps
The number of steps that the total range is divided between.

stepBlend
The mixing of the different steps. 0 is no mixing and good for getting a feel for your step ranges. 1 is complete, linear blending.

ZDefocus

The *ZDefocus* node is identical to the *Defocus* blurring node, except you can create a realistic depth of field by using the image's Z channel. The Z channel usually comes from a Z-depth 3D render, but of course can also be placed with Shake channel-swapping tools (*Reorder* and *Copy*).

Like *ZBlur*, this node works best when there is no abrupt overlapping of foreground objects over background. The foreground elements work fine, but ringing occurs on the background. If you are developing a shot of this nature, try to split your foreground into a separate element.

Parameters

This node displays the following controls in the Parameters tab:

xPixels, yPixels
Defines the kernel size for the defocus, in pixels, which produces the general size of the defocused flaring in the image. Values lower than 3 have no effect. Progressively higher values are more processor-intensive to render.

channels
Lets you set which channels Shake should defocus. You can choose one or all of the red, green, blue, or alpha channels. The default is "rgba."

percent
A slider that lets you mix the modified image and the original image together to produce a blend of both. By default, this parameter is set to 100 percent, resulting in 100 percent of the modified image being output.

shape
A pop-up menu that lets you choose the shape of the flaring in the resulting image. The fast modes give you low quality but process quickly. The circle, square, hexagon, and octagon give you a superior look, but are significantly more processor-intensive to render. The options are:

- fast gaussian
- fast box
- circle
- square
- hexagon
- octagon

boostPoint
The image value where the superwhite boosting starts. If boostPoint is set to .75, RGB values above .75 are boosted to increase flaring effects. A high value generally decreases flare areas, since fewer candidate pixels are flared. By default this is set to .95.

superWhite
Max value to boost image to. A value of 1 in the original will be boosted to this value. By boosting this, you increase the brightness of the flare area. Values around 50 yield a very strong boost.

near
The value of distance toward the camera at which maximum blur occurs.

far
The value of distance away from the camera at which maximum blur occurs.

focusCenter
The distance from the camera at which there is no blur. By default this is set to (far-near)/2+near.

focusRange
The distance away from the focusCenter, both toward and away from the camera, that remains unblurred.

steps
The amount of steps that the total range is divided between.

stepBlend
The mixing of the different steps. 0 is no mixing and good for getting a feel for your step ranges. 1 is complete, linear blending.

Part III: Optimizing, Macros, and Scripting

This section covers advanced techniques and tips that allow you to streamline your workflow in Shake.

Optimizing and Troubleshooting Your Scripts

29

This chapter provides tips and techniques for optimizing your Shake scripts, to maximize image quality and minimize render times. Additional information on troubleshooting frequently encountered issues is also provided.

Optimization

This section contains information about how to improve your scripts—maximizing image quality and processing efficiency.

Use Only the Color Channels You Need

In Shake, you can combine images that use different channels. For example, you can composite a two-channel image over a four-channel image. Shake is optimized to work on a per-channel basis—a one-channel image usually calculates about three times faster than a three-channel image.

For this reason, if you read in masks that have been generated by a different application, it's a good idea to turn them into one-channel images (using a *Monochrome* node) to save on disk space and processing time. If, later in the node tree, you apply an operation that changes channel information, Shake automatically adds back the necessary channels.

For example, if you place a *Monochrome* or an *Emboss* node after an RGB image, that image becomes a BW image at that point, speeding the processing of subsequent nodes. If you later composite the image over an RGB image, or change its color (for example, *Mult* with values of 1.01, 1, 1), it becomes an RGB image again.

Image Conversion Prior to Shake Import

Shake is "channel agnostic"—you can pipe any channel image into any other. When you generate or save mask images, you can save the images using a format that supports one-channel images (RLA or IFF, for example) to reduce disk space and network activity. You can quickly strip channels out using the command line:

To strip out the RGB channels, leaving the alpha:

- Enter the following command-line function:

```
shake mymask.#.tif -bri 0 -fo mynewmask.#.iff
```

To strip out the alpha channel, force the RGB as a monochrome 1 channel:

- Enter the following command-line function:

```
shake mymask.#.tif -mono -setAlpha 0 -fo mynewmask.#.iff
```

Note: Shake automatically optimizes itself to read only the channel it needs.

To save out a BW image as RGB for compatibility with other applications:

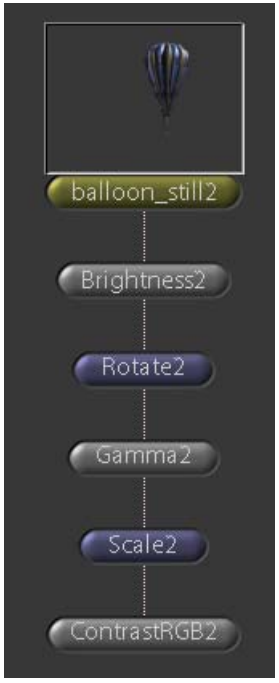
- Do one of the following:
  - Set the RGB output in the *FileOut*.
  - Use the command line function *-forcergb*:

```
shake myBWimage.#.iff -forcergb -fo myRGBimage.#.iff
```

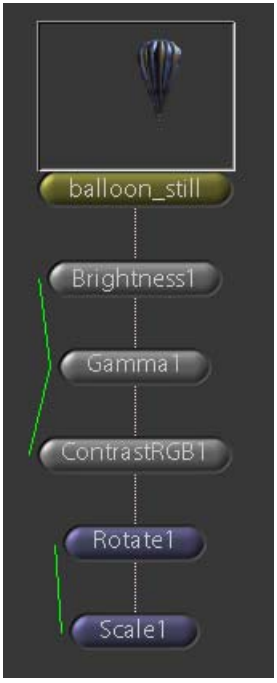
Concatenating Color-Correction and Transform Nodes

Many nodes in the Color and Transform tabs concatenate with nodes of the same type (that is, Color nodes with Color nodes, Transform nodes with Transform nodes). The nodes compile several connected nodes into one render pass, preserving quality and decreasing processing time. Nodes that concatenate are marked with a "C."

To take advantage of this feature, try not to mask or insert non-concatenating nodes between two or more concatenating nodes. In the following example, the second tree is more efficient because the color-correction and transform nodes have been grouped together, allowing them to concatenate. The effect of the second tree is identical to that of the first, but it's more computationally efficient.



Because of their arrangement, these nodes cannot concatenate.



Rearranging these nodes allows them to concatenate, but the effect is unchanged.

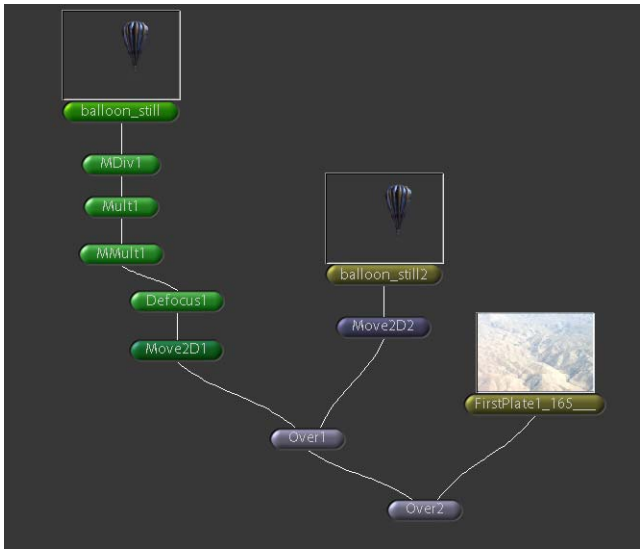
Pre-Rendering Segments of Your Node Tree

If you've finished making adjustments to a particular segment of your node tree and don't anticipate making any further changes, you can pre-render it to save time. This can be an especially important timesaver when you have extremely processor-intensive operations occurring at earlier points of expansive node trees.

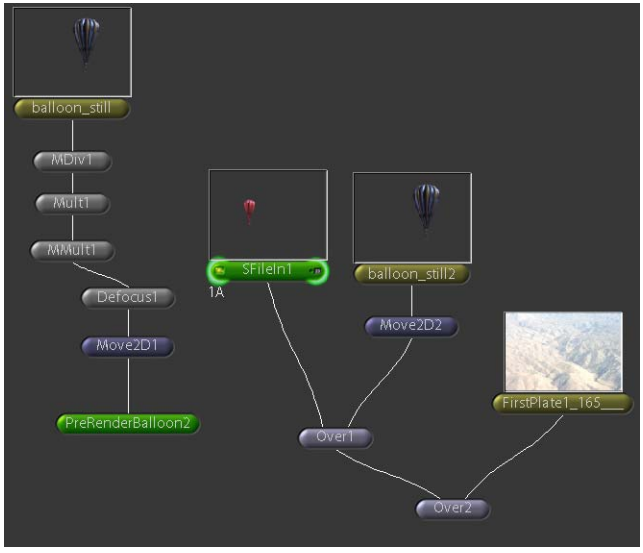
Prime candidates for pre-rendering include:

- Images that aren't animated, but have filtering and color correction applied
- Looping series of frames with processor-intensive adjustments being made to them
- Early branches of a node tree that no longer need to be adjusted

In the following example, one of the balloon images has a color correction, *Defocus* operation, and a *Move2D* node with a high motion-blur setting.

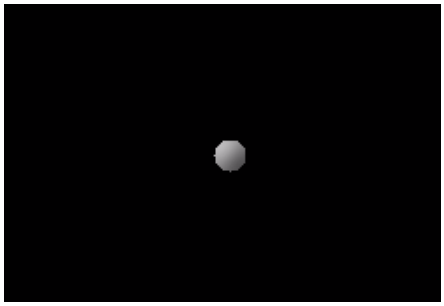


The *Defocus* and motion-blur settings are processor-intensive, so once the first balloon image's settings have been finalized, that portion of the node tree above the *Over1* node at the top can be rendered with a *FileOut* as a self-contained file. The rendered output can be read back into the script with a *FileIn*, and that image inserted into the *Over1* node in place of the original branch of the node tree. The original branch can be preserved, off to the side, in case you want to make any future adjustments.

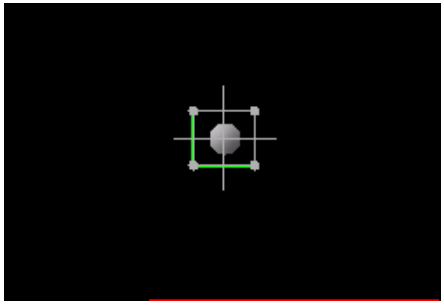


Use the SetDOD Node to Reduce Rendering Time

This node limits the portion of the image the renderer needs to concentrate on, as well as quickly masks off a large portion of the image. *SetDOD* optimizes memory, IO activity, and render times. In this example, even though the only interesting portion of the image is the ball in the middle, Shake inefficiently has to consider the entire image.



To limit the area Shake must consider, apply a Transform–*SetDOD* node to optimize the render.



Problems With Premultiplication

Most noticeable problems with premultiplication fall into two basic categories:

- Unwanted fringing appears around keyed or masked subjects.
- Color-correction or filtering nodes affect parts of an image they're not supposed to.

To resolve these issues, you must follow two specific rules about premultiplication.

The Unbreakable Rules of Premultiplication

If you don't read the full explanation of the mathematics of premultiplication in "About Premultiplication and Compositing" on page 421, here are the two rules you must *always* follow when creating a composition in Shake:

- *Rule Number 1:* Always color correct unpremultiplied images. To unpremultiply an image, use an *MDiv* node.
- *Rule Number 2:* Always filter and transform premultiplied images. To premultiply an image, use an *MMult* node.

Combine Image and Alpha Channels Prior to Filtering

If you need to mask, rotoscope, key, or otherwise add an alpha channel to an image, make sure you do it in such a way that the result is premultiplied *prior* to adding filtering nodes.

Unwanted Gamma Shifts During FileIn and FileOut

Shake and Final Cut Pro display and process the gamma of QuickTime movies and RGB image files differently.

Shake makes no automatic changes to the gamma of QuickTime or RGB image files and sequences. Users must make sure that their monitor is properly calibrated for their production environment, and that the viewer lookup parameters are set to the values required for images to display properly in the Shake Viewer. In particular, the default viewerGamma value is 1, which leaves the gamma of images displayed in the Viewer unchanged.

Final Cut Pro, on the other hand, makes some assumptions about the gamma of QuickTime and RGB image files that are imported into a project. The gamma of imported QuickTime and RBG image files is treated differently in sequences set to render in 8-bit or 10-bit YUV.

Note: While it is possible to recalibrate Apple displays via the Display Calibrator Assistant in Displays preferences, users should leave the gamma of their monitors to the 1.8 Standard Gamma setting when working in Final Cut Pro. ColorSync settings are not used by either Shake or Final Cut Pro for automatic color calibration or compensation of any kind.

Gamma in QuickTime Movies

When importing a QuickTime movie created with Shake into Final Cut Pro, users may notice a difference in the displayed gamma of the image. This is because Final Cut Pro automatically lowers the gamma of sequences playing in the Canvas on your computer's display. The gamma of QuickTime images remains untouched when the sequence is output to video or rendered as a QuickTime movie.

Solution

You can load Shake's viewer lookup controls into the Parameters tab, then change the viewerGamma parameter to .818 to preview how your composition will look in the Final Cut Pro Canvas. This only changes how your image is displayed in the Shake Viewer, and does nothing to change the gamma of the script's final rendered image.

What causes this?

Final Cut Pro assumes that QuickTime movies for codecs that support the YUV color space (including DV, DVCPRO 50, and the 8- and 10-bit Uncompressed 4:2:2 codecs) are created with a gamma of 2.2. This is generally true of movies captured from both NTSC and PAL sources. When you eventually output the sequence to video, or render it as a QuickTime movie, the gamma of the output is identical to that of the original, unless you've added color-correction filters of your own.

However, during playback on your computer's monitor, Final Cut Pro automatically lowers the gamma of a sequence playing in the Canvas to 1.8 for display purposes. This is to approximate the way it will look when displayed on a broadcast monitor. This onscreen compensation does not change the actual gamma of the clips in your sequence.

Gamma in RGB Image Files and Sequences

When importing a still image file or sequence from Shake into Final Cut Pro, the gamma may be incorrectly boosted when the sequence is output to video or rendered as a QuickTime movie.

Solution

Convert image sequences to QuickTime movies using a *FileOut* node in Shake for Mac OS X, prior to importing them into Final Cut Pro. This makes them easier to import, and also ensures that their gamma won't be changed. For the highest quality, use either the Uncompressed 8- or 10-bit 4:2:2 codec when performing this conversion, depending on the bit depth of the source image files. QuickTime Player is not recommended for this operation, as it may perform an unwanted bit-depth conversion with greater than 8-bit images.

What causes this?

Final Cut Pro assumes that all RGB image files are created with a gamma of 1.8. When RGB image files are imported into Final Cut Pro and edited into a sequence set to 8- or 10-bit YUV rendering, the gamma is automatically boosted to 2.2 in an attempt to match the other video files in your project. This boosted gamma is then used when the sequence is output to video or rendered as a QuickTime movie.

During playback on your computer's monitor, Final Cut Pro lowers the gamma of the sequence playing in the Canvas to 1.8 for display purposes. This is to approximate the way it will look when displayed on a broadcast monitor. The still image clips in your sequence are still boosted when the sequence is output to video or rendered as a QuickTime movie.

Important: QuickTime movies compressed using the Animation codec (which only supports the RGB color space) are also assumed to have been created with a gamma of 1.8. As a result, these clips are also boosted to 2.2 when edited into a sequence set to 8- or 10-bit YUV rendering.

Note: For more information on setting the rendering options of a sequence in the Video Processing tab of the Sequence Settings dialog, refer the *Final Cut Pro User Manual*.

Avoiding Bad Habits

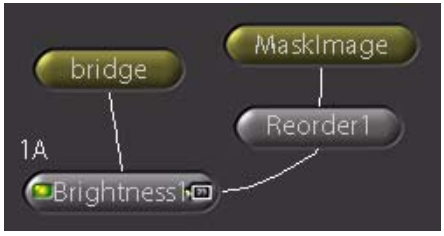
The following are common problems that come up, and are things that should generally be avoided when you're putting together scripts from scratch.

Don't Mask Layer Nodes

The side-input masks for layer nodes should not be used, as they behave counter-intuitively and will not produce the result you might expect. If you want to mask a layering node, mask the input nodes, or use the *KeyMix* node.

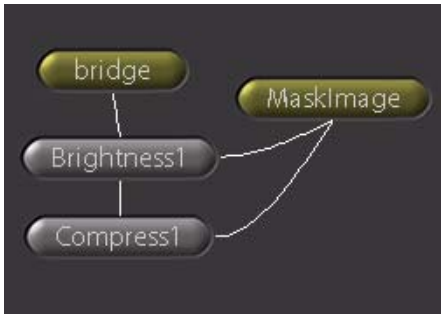
Don't Reorder Images Before Masking

This pops up a lot in client scripts, which is a *Reorder* node applied to an image before it is fed into a mask. This is unnecessary because mask inputs, as well as the *SwitchMatte* and *KeyMix* nodes, have channel selectors. There is probably no computational difference, but it is one more node in the tree.

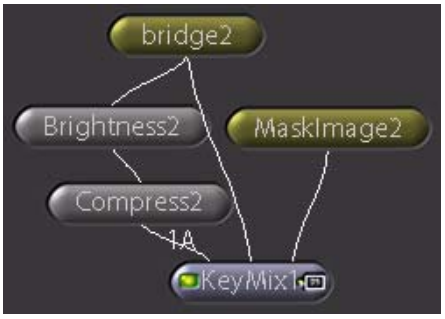


Don't Mask Concatenating Nodes

Masking a node breaks concatenation. This is bad. It slows your render and decreases quality, adds possible ringing on the mask edges, and forces multiple mask mixes. Instead, feed the tree into a *KeyMix* node.



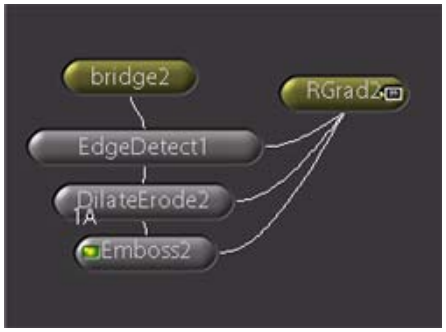
Bad



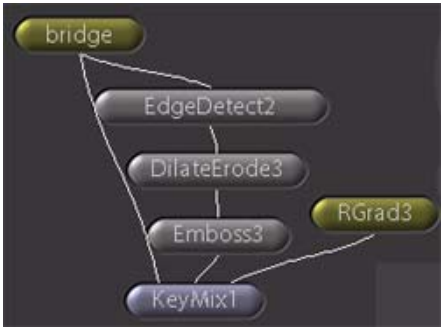
Good

Don't Apply the Same Mask to Multiple Successive Nodes

Even if the nodes do not normally concatenate, but appear one after the other along the same branch of the node tree, you get cleaner edges with the use of a *KeyMix* node. In the example below, a circular mask is applied to three filter effects. Each filter works on the previous node, so problems appear on the edges. The solution is again to use a *KeyMix* node. This yields a faster render (does not mix the mask multiple times) and a clean edge.



Bad



Good

Installing and Creating Macros

30

If there's a particular image-processing tree you've created that you would like to save for future use, you can turn it into a macro. Macros act like other nodes within Shake, except that you create them yourself using Shake's other nodes as the initial building blocks. You can then add your own expressions, scripting, and user interface elements to extend their functionality.

How to Install Macros

This section covers how to install macros on your computer, whether they're macros you've created, or macros provided by your facility, or downloaded from the web.

When you're provided with a macro, you may receive as many as three files:

- The macro itself
- A custom user UI (user interface) file for that macro
- A custom icon

Each of these files needs to be installed into a special directory for Shake to find and load it successfully.

Where to Install Macros

Macros all have a .h file extension, and are located in:

$HOME/include/startup

If this directory does not already exist, you'll need to create a series of nested directories with those exact names. For example, if you were to install the AutoDOD.h macro from the cookbook, a common location is:

/Users/myAccount/nreal/include/startup/AutoDOD.h

This is referred to as the *startup* directory, and is used for both .h preference files, and for the installation of macros.

Installing Macros Within a Script

You can also place macros inside of a script itself by copying and pasting it with a text editor. This guarantees that your macro is found by Shake when rendering that script on any machine. However, when you do this, you do not have access to any interface-building functions. Also, if you load the script back into the interface and save it out again, the macros are lost.

Where to Install Custom Interface Settings

The macro only contains the code necessary to perform the actual function. A custom UI file allows that macro to display controls for manipulating that macro's parameters. The accepted convention is that a macro's UI file name ends with the letters "UI." This makes it easier for people to whom you give your macro to know what files go where. As an example, for the AutoDOD.h macro, the accompanying UI file should be named AutoDODUI.h.

Custom UI files belong in a *ui* directory in the following location:

$HOME/nreal/include/startup/ui

Following the previous example, a common location would be:

/Users/myAccount/nreal/include/startup/ui/AutoDODUI.h

This is referred to as the *ui* directory, or the *startup/ui* directory. Files inside it are referred to as *ui* .h files.

Files that change additional default settings or add extra controls should be located in the templates directory, which is always within a *ui* directory:

/Users/myAccount/nreal/include/startup/ui/templates/defaultfilter.h

Where to Install Icons

If you're really slick, you can create your own icons to represent your new macro in the Tool tabs. Icons generally end in *.nri*.

Icons for custom macros are placed within an *icons* directory. This can be located alongside the *include* directory in the following location:

$HOME/nreal/icons/

Following the previous example, a common location would be:

/Users/my_account/nreal/icons/Layer.CopyDOD.nri

Preference File Load Order

Sometimes, macros have to be loaded in a specific order. This is mainly true if one macro uses another macro to perform its task. If you need to explicitly control the order in which macros are loaded, this can be accomplished in a variety of ways.

To explicitly control macro load order, do one of the following:

- Add an *include* statement at the beginning of the file. For example, if *macros.h* relies on *common.h* being loaded before, start *macros.h* with:

```
#include <common.h>
```

- Put all the files you want to load in a directory (for example *include/myMacros*) and create a .h file in *startup* that contains only *include* statements:

```
#include <myprefs/loadmefirst.h>
#include <myprefs/loadmesecond.h>
#include <myprefs/loadmethird.h>
```

Include files are never loaded twice, so it is okay if two .h files contain the same *#include <anyfile.h>* statement.

Creating Macros—The Basics

The MacroMaker is an interactive tool to help you create new nodes by combining previously existing nodes. Shake's scripting language is essentially C-like, so you have access to an entire programming language to manipulate your scripts. Because the MacroMaker cannot account for all possibilities, consider the MacroMaker a tool to help you get started in using Shake's language. Use the MacroMaker to build the initial files, and then modify these files to customize your macros.

Note: For online portions of this section, choose Help > Customizing Shake.

For more information, such as about macro structure, common errors when making macros, and several examples, see "Creating Macros—In Depth" on page 914. For a tutorial on making macros, see Tutorial 8, "Working With Macros," in the *Shake 4 Tutorials*.

Opening Scripts That Use Uninstalled Macros

If you open a Shake script that contains macros that you do not have on your system, you have the option to load the script using a substitute node, or not to load the script at all. For more information, see Chapter 2, "Setting a Script's Global Parameters," on page 91.

Creating the Node Structure

First, create the node structure for the function (what you want to occur in the node). This can be very simple or very complex. To help illustrate macro building, the following example creates a function that randomly scales, pans, and rotates your image, similar to *CameraShake*, but with more moving parameters.

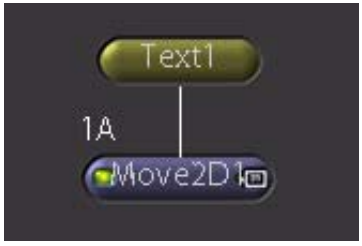
Important: The *QuickPaint*, *ColorCorrect*, *HueCurves*, and *Lookup* nodes should not be used inside of macros.

To create the node structure:

1 Click the Image Tool tab, then click the *Text* command.

The *Text* node is added to the Node View.

2 Click the Transform Tool tab, then add a *Move2D* node.



3 Enter the following parameters:

Parameter	Value
xPan	(turbulence(time,2)-.5)*20
yPan	(turbulence(time+100,2)-.5)*20
angle	(turbulence(time+200,2)-.5)*10
xScale	turbulence(time+300,2)/2+.75

In these examples, the turbulence function generates continual noise between 0 and 1 (see Chapter 31, "Expressions and Scripting," on page 935). Since you do not want all fluctuations to have the same pattern, offset the seed each time you call the function (*time, time+100, time+200, time+300*). Then, simple math is used to get the values in an appropriate range. For example, in angle, .5 is subtracted to adjust the value to a range of -.5 to .5. The result is then multiplied by 10, and yields a range between -5 and 5 degrees of rotation.

4 Drag in the playhead in the Time Bar to test the animation.

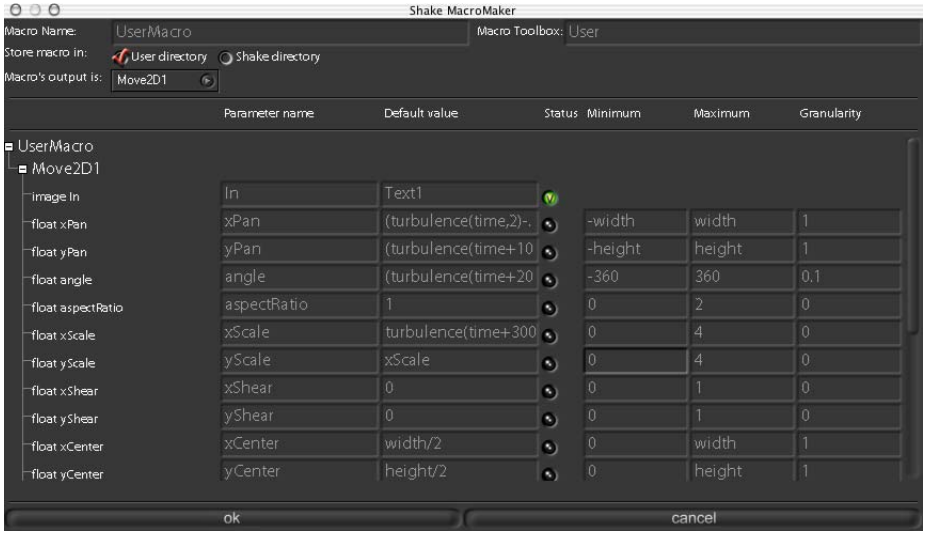
Making a Macro

Since the above steps are tedious to manually recreate, create a macro.

To create a macro:

1 In the node tree created above, select the *Move2D* node, and press Shift-M (or right-click and choose Macro > Make Macro from the shortcut menu).

The MacroMaker is launched.



In the top portion of the Shake MacroMaker window, you specify the file name, the save location, and the tab where the node appears.

2 In the Macro Name field, enter *RandomMove*.

3 In the Macro Toolbox field, enter *Transform*.

Note: Case sensitivity is always important.

4 Leave the "Store macro in" selection at the default "User directory" setting.

5 Leave the "Macro's output is set" pop-up menu set to Move2D1 (there are no other choices in this example).

Setting	Value
Macro Name	The name of the macro you create. It is also the name of the file you save (see below).
Macro Toolbox	The Tool tab that stores the macro. If a tab does not exist, it creates a new one.

Setting	Value
Store Macro in	• *User directory:* Saves the macro in your *$HOME/nreal/include/startup* as *MacroName.h* and a second ui file in *$HOME/nreal/include/startup/ui* as *MacroNameUI.h*. • *Shake Directory:* Saves the macro in the Shake distribution directory, as *<ShakeDir>/include/startup/MacroName.h* and *<ShakeDir>/include/startup/ui/MacroNameUI.h*. For more information on these directories and their functions, see "Creating and Saving .h Preference Files" on page 355.
Macro's output is	Presents a list of all nodes that are included in the macro (just one for the *Move2D* example). Select the node to pass out of the macro. Shake usually does a correct guess for this node if you have only one terminating branch.

The lower portion of the Shake MacroMaker window lists all of the nodes and the parameters that can be exposed in the created node. For example, since the image is fed into a *Move2D* node, the image In parameter is enabled.

Parameter	Value
Parameter name	The name of the slider (in the case of float and int values) or the knot input (in the case of image inputs). Arbitrarily renaming your parameters is not recommended (such as *eggHead*), as you have the benefit of Shake's default interface behaviors (pop-up menus, subtrees, color pickers, on/off switches, and so on) if they have the same name.
Default value	All current values of the nodes are fed into value fields. To set a default value for exposed parameters (that is, parameters with the visible Status light on), set the value here.
Status	Enable Status to expose this parameter. If you expose an image, it adds an input to the top of the node. If you expose anything else, it gives you a slider or value field in the Parameters tab.
Minimum	For slider-based parameters, sets the lower slider limit.
Maximum	For slider-based parameters, sets the upper slider limit.
Granularity	Sets how much the slider jumps between values.

Since you already have most of the parameters needed in this example, you only want to expose motionBlur, shutterTiming, and shutterOffset values.

6 Click the V buttons for motionBlur, shutterTiming, and shutterOffset.

7 Click OK.

The new push-button node appears in the Transform tab.

8 Add the new node to the Node View.

In the *RandomMove* parameters, only the motionBlur settings are available, and have automatically collapsed into a subtree (due to the default Shake behavior).

To modify the macro:

1 If you placed your macro in your User Directory, go into your $HOME directory, and then into the *nreal/include/startup* subdirectory.

In the *startup* directory, a new file called *RandomMove.h* appears.

2 Open the *RandomMove.h* file in a text editor:

```
image RandomMove(
  image In=0,
  float motionBlur=0,
  float shutterTiming=0.5,
  float shutterOffset=0
)
{
  Move2D1 = Move2D(
     In,
     (turbulence(time,2)-.5)*20,
     (turbulence(time+100,2)-.5)*20,
     (turbulence(time+200,2)-.5)*10,
     1,
     turbulence(time+300,2)/2+.75,
     xScale,
     0, 0,
     width/2, height/2,
     “default”, xFilter,
     “trsx”, 0,
     motionBlur,
     shutterTiming,
     shutterOffset,
     0,

  );

     return Move2D1;
}
```

Edit the macro in this file. The parameters motionBlur, shutterTiming, and shutterOffset are declared in the first few lines, and then the assigned default values. The image input In is also assigned a value of 0, so there is no expected image input when the macro is created.

```
image RandomMove(
  image In=0,
  float motionBlur=0,
  float shutterTiming=0.5,
  float shutterOffset=0
)
...
```

3 Modify the behavior of the macro. Each turbulence function has a frequency of 2, for example, turbulence(time,2). To create a new parameter to modify the frequency, add the following line (in bold):

```
image RandomMove(
  image In=0,
  float frequency = 2,
  float motionBlur=0,
  float shutterTiming=0.5,
  float shutterOffset=0
)
...
```

4 Now that you have a new parameter, you must substitute it into the macro body wherever you want that value to apply. Insert the variable into the macro body:

```
image RandomMove(
  image In=0,
  float frequency = 2,
  float motionBlur=0,
  float shutterTiming=0.5,
  float shutterOffset=0
)
{
  Move2D1 = Move2D(
     In,
     (turbulence(time,frequency)-.5)*20,
     (turbulence(time+100,frequency)-.5)*20,
     (turbulence(time+200,frequency)-.5)*10,
     1,
     turbulence(time+300,frequency)/2+.75,
     xScale,
     0, 0,
     ...
```

5 Save the file.

6 To see the results of the modification, restart Shake.

Modifying the Macro Interface

The macro file in the *startup* directory merely creates the function. The interface is built by Shake each time it is launched. Therefore, the MacroMaker also creates a second file in the *startup/ui* subdirectory that creates a button and sets slider ranges for the node. For example, if you created the new frequency slider in the above example, you may have noticed that the slider only goes from 0 to 1. You can modify this in the ui file.

To modify the macro interface:

1. In the text editor, open the *RandomMoveUI.h* file (created in the above example) in the *nreal/include/startup/ui* subdirectory.

```
nuiPushMenu("Tools");
  nuiPushToolBox("Transform");
     nuiToolBoxItem("@RandomMove",RandomMove());
  nuiPopToolBox();
nuiPopMenu();
nuiDefSlider("RandomMove.motionBlur",0,1,0,0,0);
nuiDefSlider("RandomMove.shutterTiming",0,2,0,0,0.01);
nuiDefSlider("RandomMove.shutterOffset",-1,1,0,0,0.01);
```

   The first line opens the Tool tabs. The second line opens the Transform tab. To place the macro in a different tab, change the word "Transform" to a different name. The third line creates the button. The first occurrence of the word "RandomMove" is preceded by an @ sign, indicating that there is no icon (not related to Shake's unpadded frame wildcard) for the button, so therefore the text "RandomMove" should be printed on the button. The second listing of the word "RandomMove" is the function that is called when you click the button. Because you have default arguments already supplied in the macro, you do not need to supply any arguments. If you did supply arguments, they are placed between the parentheses: (). The last lines basically say "For the *RandomMove* function, set the slider ranges for the shutterOffset parameter between -1 and 1, with some extra granularity settings." Since this sets the slider range, you can copy it and adapt it for the new frequency parameter.

2. Copy the last line of code and paste in a copy. Change the word "shutterOffset" to "frequency," and adjust the values (based on the line in bold below):

```
...
nuiDefSlider("RandomMove.shutterTiming",0,2,0,0,0.01);
nuiDefSlider("RandomMove.shutterOffset",-1,1,0,0,0.01);
nuiDefSlider("RandomMove.frequency",0,10,0,0,0.01);
```

To add an icon to the button:

1. Create a 75 x 40 pixel image.
2. Save the image as *TabName.Name.nri* to your *$HOME/nreal/icons* directory. In the above example, it would be called *Transform.RandomMove.nri*.

Note: You must strip out the alpha channel of the image. You can do this in Shake with a *SetAlpha* node set to a value of 0. Set the *FileOut* to your *$HOME/nreal/icons* directory, with the name *TabName.Name.nri*, and render the file.

3 In the *RandomMoveUI.h* file, remove the @ sign on line 3, and save the text file.

4 Restart Shake.

The *RandomMove* node appears with the icon.

Note: In all cases of the above code, case sensitivity is important. If the macro name is in all capital letters, then all occurrences of the name must also be in all capital letters. The same restriction applies to parameter and icon names.

For more information on customizing Shake, see Chapter 14, "Customizing Shake."

Creating Macros—In Depth

This section discusses additional information, such as basic macro structure, common errors when creating macros, and sample macros.

The macro files can be saved into the script that uses them, or can be saved in a file with a .h file extension. These files are saved in your *startup* directory, located either in *<ShakeDir>/include/startup*, *$HOME/nreal/include/startup*, or a *startup* directory under a directory specified by the environment variable $NR_INCLUDE_PATH. For more information, see "Setting Preferences and Customizing Shake" on page 355.

Basic Macro Structure

The following is what a macro looks like:

```
dataType MacroName(
dataType parameterName=defaultParameterValue,
dataType parameterName=defaultParameterValue,
...
)
{
Macro Body
return variable;
}
```

For example, the following function, called "angle," calculates the angle between two points, returns a float, using the atan2d function. (For more information, see the "Trig Functions (in degrees)" section of the table in "Reference Tables for Functions, Variables, and Expressions" on page 941.)

Notice that the default parameter value is optional, so if you use this particular function, all four values must be supplied:

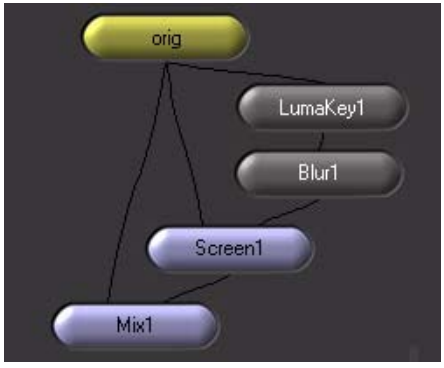
```
float angle(
  float x1,
  float y1,
  float x2,
  float y2
)
{
  return atan2d(y2-y1,x2-x1);
}
```

Because the macro is so simple (one function), there is no Macro Body per se; the function is attached to the return statement, which indicates what is spit out of the macro. To use this function, use something like the following:

```
myAngle = angle(0,0,100,100);
```

that returns 45.0.

The following is an example of an image function. It adds that "vaseline-on-the-lens" effect. The following represents the node tree, with a split-screen example of the effect.



The *LumaKey* node is used to extract only the highlights. The highlights are blurred, and then applied back on the original image with the *Screen* node, which is nice for glows and reflections. The *Mix* node is used to control how much of the original image shows through. The example image shows the original image on the left, and the macro results on the right.

Photo courtesy of Photron

The following are the nodes reformatted as a macro. The macro parameters are bold.

```
image SoftGlow(
image In=0,
float blur=0,
float lowClip=.3,
float hiClip=.9,
float percent=100
)
{
LumaKey1 = LumaKey(In, lowClip, hiClip, 0, 0, 1);
Blur1 = Blur(LumaKey1, blur, xPixels, 0, "gauss", xFilter,
  "rgba");
Screen1 = Screen(In, Blur1, 1);
Mix1 = Mix(In, Screen1, 1, percent, "rgba");
return Mix1;
}
```

The macro name is *SoftGlow*, and it outputs an image (line 1). The parameters are expecting one image input named *In* (line 2). This gives you one input at the top of the node. If you want two image inputs, there would be two image parameters, and so on. The other values are all float, controlling the macro settings (lines 3-6). These are assembled in the macro body, and the return statement indicates that *Mix1* is output. The low and hiClip parameters determine the levels of the highlights.

If you save this into a *startup .h* file, for example, *$HOME/nreal/include/startup/SoftGlow.h*, it is immediately available on the command line. You can also find a copy of this called *SoftGlow.h* in *doc/html/cook/macros*.

Type:

```
shake -help softglow
```

to return:

```
-softglow [blur] [lowClip] [hiClip] [percent]
```

File Name Versus Macro Name

Names of files have *nothing* to do with names of macros. Only the function name is important when called in the script. You can also have multiple macros per file.

Loading Image Macros Into the Interface

When you start the Shake interface, the macros do not appear in the interface. A separate file and set of functions are required to load the macros in the interface. These ui functions are saved in a subdirectory of *startup* called *ui*. The following ui code creates a button in the Filter tab with no icon that is labeled "SoftGlow," and calls up the *SoftGlow* function when clicked. Save it in *$HOME/nreal/include/startup/ui* or *$NR_INCLUDE_PATH/startup/ui*. You can also find a copy of this in *doc/html/cook/macros* called *SoftGlowUI.h*.

```
nuiPushMenu("Tools");
  nuiPushToolBox("Filter");
     nuiToolBoxItem("@SoftGlow",SoftGlow());
  nuiPopToolBox();
nuiPopMenu();
```

The @ sign indicates that no icon is associated with the button. If you have an icon, omit the @ sign (do not add the tab prefix or image type extension) and save an icon image, 75 x 40 pixels in size, in your icons directory (*<ShakeDir>/icons or $HOME/nreal/icons* or *$NR_ICON_PATH*), naming it *Filter.SoftGlow.nri*. To see this process listed step-by-step, see Tutorial 8, "Working With Macros," in the *Shake 4 Tutorials*.

The images for the icons have the following characteristics:

- 75 x 40 pixels
- No alpha channel
- Named as *TabName.FunctionName.nri*

  When calling the icon in the *ui.h* file, omit the *TabName.* and *.nri*.
- Font: 1-point Arial
- Saved in *<ShakeDir>/icons* or *$HOME/nreal/icons* or *$NR_ICON_PATH*

Typical Errors When Creating Macros

The following table contains a list of typical errors in macro creation. When diagnosing a macro, first run the macro in the command line: *shake -help myFunction*. If nothing appears, there is a problem with your *startup .h* file. If it works fine, move on to the interface, and read any error messages in the Console tab. The Console tab is your number-one diagnostic tool.

Error Behavior	Probable Cause
In the command line, the function doesn't appear when you type the following: *shake -help myFunctionName*	• The file is not saved in a *startup* directory. See above. • The file does not have a .h file extension, or it has an improper extension. • The name of the macro (the second word in the file) is not the same as what you have called in the help command.
Function does not appear in the interface.	• The ui file is not saved in a *ui* directory. See above. • The ui file does not have a .h file extension, or it has a .txt extension. • The name of the macro (the second word in the file) is not the same as what you have called in the ui file, possibly because of capitalization errors.
In the interface, the button appears without an icon.	You have not removed the @ sign in your ui.h file. Otherwise, follow Tutorial 8, "Working With Macros," in the *Shake 4 Tutorials*.
The icon appears as a dimple.	The icon cannot be found. • Make sure the icon is saved in an icons directory. See above. • The icon should be named *TabName.FunctionName.nri*. • The ui code should say: *nuiToolBoxItem("FunctionName",Function());* NOT *nuiToolBoxItem("TabName.FunctionName.nri",Function());* • Check capitalization.
The icon is fine, but nothing happens when you click it.	• Check capitalization errors. • Check that the correct function is called in the ui file and that the function exists. (For example, type *shake -help functionname* in the command line.) • Check that default arguments have been specified.

Setting Default Values for Macros

There are two places to set default values for your macros. The first is in the *startup .h* file when you declare the parameters. The following is from the example above:

```
image SoftGlow(
image In=0,
float blur=0,
float lowClip=.3,
float hiClip=.9,
float percent=100
)
...
```

Each of these has a default value assigned. Note that the image has 0, which indicates "no input."

These values are applied to both the command line or the graphical user interface defaults. If you do not supply a default argument, you must enter a value when you call the function. It is therefore recommended that you enter defaults.

The second location is in the *ui.h* file when the function is called. To override the startup defaults, enter your own in the *ui.h* file. Normally, you have something like the following in your *ui.h* file:

```
nuiPushMenu("Tools");
  nuiPushToolBox("Filter");
     nuiToolBoxItem("@SoftGlow",SoftGlow());
  nuiPopToolBox();
nuiPopMenu();
```

This sets new default values just for the interface:

```
...
  nuiToolBoxItem("@SoftGlow",SoftGlow(0,100,.5,1, 95));
...
```

Changing Default Settings

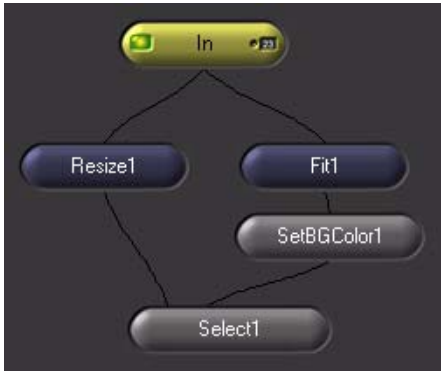
Most of Shake's functions (third-party development excepted) are stored in two files in *<ShakeDir>/include* in *nreal.h* and *nrui.h*. *nreal.h* is the complete list of all functions and settings. The *nrui.h* file builds the interface. You can modify these files to suit your needs, but it is strongly recommended that you make a backup of these files before you begin. Errors in the file may result in unexpected problems with Shake.

These files are also an excellent source for examples.

Attaching Parameter Widgets

If you take a look at "Using Parameters Controls Within Macros" on page 379, and experiment with different nodes in the interface, you see that you can attach many behaviors to parameters. This section takes a raw macro and shows different examples of behaviors to change the parameters sliders to more efficient widgets.

The macro that is created in the following example is called *VidResize*. It takes an image of any size and resizes it to video resolution. There are slider controls to specify NTSC or PAL (vidformat), to maintain the aspect ratio (keepAspect), and if you do keep the aspect ratio, the color of the exposed background area. The following image represents the original node tree.



To create the VidResize macro:

1. Quit Shake.
2. To load the macro, copy the *VidResize.h* file from *doc/html/cook/macros* to your *$HOME/nreal/include/startup* directory.
3. Copy the *VidResizeUI.h* file from *doc/html/cook/macros* into your *$HOME/nreal/include/startup/ui* directory.
4. Start the Shake interface.
5. Create a *Grad* node with the following parameter settings:
   - Set the width to 400.
   - Set the height to 200.
6. Attach the Transform–*VidResize* node and test each slider. To have any effect, the keepAspect parameter needs to jump to 2.

```
image VidResize(
  image In=0,
  int keepAspect = 1,//keeps aspect ratio or not
  int vidFormat = 0, //This select NTSC (0) or PAL (1) res
  float bgRed = 0,//if keeping aspect ratio, determines
  float bgGreen =0,//the bg color
  float bgBlue =0
)
{
  curve int yRes = vidFormat ==0?486:576;
  Fit1 = Fit(In, 720, yRes, "default", xFilter, 1);
  Resize1 = Resize(In, 720, yRes, "default", 0);
  SetBGColor1 = SetBGColor(Fit1, "rgbaz",
```

```
    bgRed, bgGreen, bgBlue, 0, 0
  );
  Select1 = Select(keepAspect, Resize1, SetBGColor1, 0, 0);
  return Select1;
}
```

The ui file looks like this:

```
nuiPushMenu("Tools");
  nuiPushToolBox("Transform");
     nuiToolBoxItem("@VidResize",VidResize());
  nuiPopToolBox();
nuiPopMenu();
```

The only tricky portions in the macro so far are the *Select* function, and the logic to choose the height. When the branch of *Select* equals 1, the first image input is passed through. When it equals 2, the second branch is fed through. The keepAspect parameter is connected to *Select* to choose the branch you want. The other tricky portion is the height logic, the first line of the macro body. It declares an internal variable (it cannot be seen outside the macro) named yRes. It tests vidFormat to see if it equals 0. If so, it sets *yRes* to equal 486, the height of an NTSC image. Otherwise, it sets yRes to 576, the standard PAL image height.

Setting Slider Ranges

The first inconvenience about the macro, when used in the interface, is that the keepAspect slider goes from 0 to 1. The values you want are 1 or 2. Since you do not want to have to manually enter the format, it is easier to set the slider range from 1 to 2. This is done in the ui file *VidResizeUI.h*. To find the format, see "Using Parameters Controls Within Macros" on page 379. If you are reading this document electronically, you can copy and paste the command from the document into the text file.

To set the slider range:

1. Load the *ui/VidResizeUI.h* file into the text editor.
2. Add the *nuiDefSlider* function to set a slider range:

```
nuiPushMenu("Tools");
  nuiPushToolBox("Transform");
     nuiToolBoxItem("@VidResize",VidResize());
  nuiPopToolBox();
nuiPopMenu();
nuiDefSlider("VidResize.keepAspect",     1,   2);
```

The word "VidResize" of VidResize.keepAspect indicates that you only want to modify the keepAspect slider in the *VidResize* function. If you do not specify this, for example, you just have *nuiDefSlider("keepAspect", 1, 2)*; and it tries to apply that slider range to all parameters named keepAspect, no matter what function they are in.

3. Save *VidResizeUI.h* and start Shake again.
4. Create the *VidResize* node (there is no need to test it on an image).

The keepAspect slider now goes from 1 to 2.

Inappropriate Behavior in All the Wrong Places

If you start to see controls on parameters that you have not created, or if you see other functions that have odd behaviors, make sure you have specified what function receives the control. If you set:

```
nuiDefSlider("depth", 1, 2);
```

anytime Shake sees a parameter named "depth" (for example, if somebody makes a macro to set bit depth to a certain value), it takes a range of 1 to 2. Therefore, ensure that you preface the depth with a function name:

```
nuiDefSlider("MyFunction.depth", 1, 2);
```

Creating an On/Off Button

Rather than using a value (1 or 2) to indicate what the slider does, you can create an on/off button. When the button is on, you want to maintain the aspect ratio. When off, you do not want to maintain the aspect ratio. Again, the format information can be found in "Using Parameters Controls Within Macros" on page 379. If you are reading this document electronically, you can copy and paste the command.

To create an on/off button:

1. Replace the slider range function in the *VidResizeUI.h* file:

```
nuiPushMenu("Tools");
  nuiPushToolBox("Transform");
     nuiToolBoxItem("@VidResize",VidResize());
  nuiPopToolBox();
nuiPopMenu();
nuxDefExprToggle("VidResize.keepAspect");
```

   The problem here is that it always returns a value of 0 or 1—0 is off and 1 is on. You want values of 1 or 2 because that is what the macro is counting on. Therefore, you must edit the macro.

2. In the startup file *VidResize.h,* add 1 to the test of keepAspect in the *Select* function:

```
...
  SetBGColor1 = SetBGColor(Fit1, "rgbaz",
     bgRed, bgGreen, bgBlue, 0, 0
  );
  Select1 = Select(keepAspect+1, Resize1, SetBGColor1, 0, 0);
  return Select1;
}
```

3. Save the file and start Shake.

The keepAspect parameter has an on/off button.

Attaching Color Pickers and Subtrees

Using a slider to select a color is not nearly as impressive to your arch foes as using the Color Picker, so attach bgRed, bgGreen, and bgBlue to a color control to interactively pick your color. For the format information, see "Using Parameters Controls Within Macros" on page 379.

Since this is an interface change, edit the ui *VidResizeUI.h* file:

```
nuiPushMenu("Tools");
  nuiPushToolBox("Transform");
     nuiToolBoxItem("@VidResize",VidResize());
  nuiPopToolBox();
nuiPopMenu();
nuxDefExprToggle("VidResize.keepAspect");
nuiPushControlGroup("VidResize.Background Color");
  nuiGroupControl("VidResize.bgRed");
  nuiGroupControl("VidResize.bgGreen");
  nuiGroupControl("VidResize.bgBlue");
nuiPopControlGroup();
nuiPushControlWidget("VidResize.Background Color",
  nuiConnectColorTriplet(kRGBToggle,kCurrentColor,1)
);
```

This file is interesting because the first five new lines are the code to make a subtree. Even if you do not add the last lines to attach the color control, the three color sliders are still organized in a subtree named Background Color.

The last three lines attach the color picker to the subtree called Background Color, and sets the picker to select an RGB (as opposed to HSV, HLS, or CMY) color control that uses the Current (as opposed to the Average, Maximum, or Minimum scrub) color.

The Color control is added to the interface.

Attaching Button Toggles

Next, attach a button to toggle the *vidFormat*. Since the 0 and 1 settings are not very intuitive for the video format selection, create buttons labeled "NTSC" and "PAL." The following examples show two ways to attach a button toggle.

To attach a button toggle:

1. Create a directory called *icons/ux* in your *$HOME/nreal* directory:

   mkdir -p $HOME/nreal/icons/ux

2. Copy all of the icons that begin with *vr_* from *doc/html/cook/macro_icons* to your *$HOME/nreal/icons/ux* directory. There are a total of eight files.

   When clicked, the first button toggles between PAL (*vr_pal.off.nri*) and NTSC (*vr_ntsc.off.nri*). Two additional buttons, *vr_pal.off.focus.nri* and *vr_ntsc.off.focus.nri*, indicate when the pointer is over the button. These are called focus buttons. To view the code, see "Using Parameters Controls Within Macros" on page 379.

3. To add the toggle button function, edit the *VidResizeUI.h* file:

```
nuiPushMenu("Tools");
  nuiPushToolBox("Transform");
     nuiToolBoxItem("@VidResize",VidResize());
  nuiPopToolBox();
nuiPopMenu();

nuxDefExprToggle("VidResize.keepAspect");

nuiPushControlGroup("VidResize.Background Color");
  nuiGroupControl("VidResize.bgRed");
  nuiGroupControl("VidResize.bgGreen");
  nuiGroupControl("VidResize.bgBlue");
nuiPopControlGroup();
nuiPushControlWidget("VidResize.Background Color",
  nuiConnectColorTriplet(kRGBToggle,kCurrentColor,1)
);

nuxDefExprToggle("VidResize.vidFormat",
  "ux/vr_ntsc.off.nri|ux/vr_ntsc.off.focus.nri",
  "ux/vr_pal.off.nri|ux/vr_pal.off.focus.nri"
);
```

The new lines list the normal button, followed by the focus button. The *icons* directory is automatically scanned, but notice you have specified the *ux* subdirectory. The value returned is always 0 for the first entry, 1 for the next entry, 2 for the third entry, and so on. You can have as many entries as you want. Each button click moves you to the next choice.

4 Save the file and start Shake again.

5 Create the *VidFormat* node again.

The vidFormat parameter has a PAL/NTSC toggle.

The button created in the above steps is a single button that toggles through multiple choices. This is fine for binary (on/off) functions, but less elegant for multiple choice toggles. In the next example, create radio buttons. Radio buttons are similar to toggle buttons, except that you simultaneously see all available buttons. This code lists only one button name. Shake automatically assumes there is an on, on.focus, off, and off.focus version of each button in the directory you specify. If you copied the *vr_* buttons earlier, you indeed have all of these buttons. The code looks like the following in "Using Parameters Controls Within Macros" on page 379.

6 Again, edit the *VidResizeUI.h* file:

```
nuiPushMenu("Tools");
  nuiPushToolBox("Transform");
     nuiToolBoxItem("@VidResize",VidResize());
  nuiPopToolBox();
nuiPopMenu();

nuxDefExprToggle("VidResize.keepAspect");

nuiPushControlGroup("VidResize.Background Color");
  nuiGroupControl("VidResize.bgRed");
  nuiGroupControl("VidResize.bgGreen");
  nuiGroupControl("VidResize.bgBlue");
nuiPopControlGroup();
nuiPushControlWidget("VidResize.Background Color",
  nuiConnectColorTriplet(kRGBToggle,kCurrentColor,1)
);

nuxDefRadioBtnControl(
  "VidResize.vidFormat", 1, 1, 0,
  "0|ux/vr_ntsc",
  "1|ux/vr_pal"
);
```

The numbers immediately to the left of the icon listing, for example, the 0 in "0|ux/vr_ntsc", show the value returned when that button is clicked. The nice thing about radio buttons is that they can return strings, float, or int. For example, the channel parameter in the *KeyMix* node selects the channel to do the masking, with R,G,B, or A returned, all strings. Because your macro *VidResize* only understands 0 or 1 to be meaningful, use 0 and 1 as your return values.

7 Save the text file and start Shake again.

The radio buttons appear in the *VidFormat* parameters.

Making Radio or Toggle Buttons

There is an unofficial function to create these buttons. That's right, you can use a macro to help make your macros.

In *doc/html/cook/macros*, copy *radiobutton.h* and *relief.h* to your *startup* directory.

In the command line, type something like the following:

shake -radio "NTSC size" 79 vr_ntsc $HOME/nreal/icons/ux -t 1-4 -v

This creates four buttons named *vr_ntsc.on.nri*, *vr_ntsc.on.focus.nri*, *vr_ntsc.off.nri*, and *vr_ntsc.off.focus.nri* in *$HOME/nreal/icons/ux*. Each button is 77 pixels wide and each one says "NTSC size." You must do this in the command line because at the moment the *FileOut* does not work in a macro unless it's the command line, and frames 1-4 create the four different files.

Attaching Pop-Up Menus

Although the radio buttons are pretty cool, another frequent option is a pop-up menu. The pop-up menu is good when there are more choices than space in a standard Parameters tab. The only catch is that they return strings (words), not numbers, so you have to add some logic in your macro to interpret the strings. The following is the ui code, which can also be found in "Using Parameters Controls Within Macros" on page 379.

To attach a pop-up menu:

1 Add the following code to the ui file to create a pop-up menu:

```
nuiPushMenu("Tools");
  nuiPushToolBox("Transform");
     nuiToolBoxItem("@VidResize",VidResize());
  nuiPopToolBox();
nuiPopMenu();

nuxDefExprToggle("VidResize.keepAspect");

nuiPushControlGroup("VidResize.Background Color");
  nuiGroupControl("VidResize.bgRed");
  nuiGroupControl("VidResize.bgGreen");
  nuiGroupControl("VidResize.bgBlue");
nuiPopControlGroup();
nuiPushControlWidget("VidResize.Background Color",
  nuiConnectColorPControl(kRGBToggle,kCurrentColor,1)
);
nuxDefMultiChoice("VidResize.vidFormat", "NTSC|PAL");
```

You can add as many entries as you want, and each entry is separated by the "|" symbol. There is a problem with pop-up menus, however. As mentioned above, pop-up menus only return the word you entered. Therefore, you must change the startup macro file. In *VidResize.h* in the *startup* directory, change vidFormat to be a string instead of an integer, placing its default parameter as either NTSC or PAL, both in quotes. You also use an *if/else* statement below it to evaluate vidFormat and assign the proper yRes value based on the vidFormat value.

2 Edit the *VidResize.h* file in *startup* to change the vidFormat logic to deal with strings instead of integers:

```
image VidResize(
image In=0,
int keepAspect=1, //keeps aspect ratio or not
string vidFormat="NTSC", //This select NTSC  or PAL  res
float bgRed=0, //if keeping aspect ratio, determines
float bgGreen=0, //the bg color
float bgBlue=0
)
{
if (vidFormat=="NTSC"){
  yRes=486;
  } else {
  yRes=576;
  }
//  curve int yRes = vidFormat==0?486:576;
  Fit1 = Fit(In, 720, yRes, "default", xFilter, 1);
  Resize1 = Resize(In, 720, yRes, "default", 0);
  SetBGColor1 = SetBGColor(Fit1, "rgbaz",
  bgRed, bgGreen, bgBlue, 0,0
```

```
  );
  Select1 = Select(keepAspect+1, Resize1, SetBGColor1, 0, 0);

  return Select1;
}
```

3 Save the file and start Shake again.

The pop-up menu appears in the parameters.

You can alternatively avoid the use of the *if/else* statement and use something similar to what you had before:

```
curve int yRes = vidFormat=="NTSC"?486:576;
```

The reason the *if/else* is used is that you usually have multiple entries on your pop-up menu, and the conditional expression gets a bit unwieldy, so it's better to create a long *if/else* statement. It's unwieldy and long, like that run-on sentence.

Standard Script Commands and Variables

The following tables include the standard script commands and variables.

Standard Script Commands

Script Controls	Command-Line Equivalent	Description
SetTimeRange("1-100");	`-t 1-100`	The time range, for example, frames 1-200. Note this is always in quotes.
SetFieldRendering(1):	`-fldr 1`	Field Rendering. • 0 = off • 1 = odd - PAL • 2 = even - NTSC
SetFps(24);	`-fps 24`	Frames per second.
SetMotionBlur(1, 1,0);	`-motion 1 1 , -shutter 1 1`	Your three global motion blur parameters.
SetQuality(1);	`-fast 0 or 1`	Low (0) or high (1) quality.
SetProxyFilter("default");	`-proxyfilter`	The default filter, taken from the "Filters Within Transform Nodes" on page 862.
SetProxyScale(1,1);	`-proxyscale 1 1`	Proxy scale and ratio for the script. See Chapter 4, "Using Proxies," on page 137.

Script Controls	Command-Line Equivalent	Description
SetPixelScale(1,1):	`-pixelscale 1 1`	Pixel scale and ratio for the script. See Chapter 4, "Using Proxies," on page 137.
SetDefaultWidth(720); SetDefaultHeight(480); SetDefaultAspect(1); SetDefaultBytes(1); SetDefaultViewerAspect(1);		If a node goes to black or if you create an image node such as *RGrad*, it takes this resolution by default.
SetFormat("Custom")		Sets your defaults automatically for resolution and aspect from the precreated list of formats. These formats are stored in your *startup .h* file in the following format: *DefFormatType("Name", width, height, aspectRation, framesPerSecond, fieldRendering);*

Variables

Common Variables	Description
width	Returns the width of the current node.
height	Returns the height of the current node.
bytes	Returns the bit depth in bytes, either 1, 2, or 4.
in, outPoint	Returns the in and out frames of the clip in time.
time	The current frame number.
width/2	The center of the image in X.
height/2	The center of the image in Y.
dod[0], dod[1], dod[2], dod[3]	The left, bottom, right, and top edges of the Domain of Definition (DOD) of the current image.
parameterName	The value of a parameter within the same node.
NodeName.parameterName	The value of a parameter in a separate node named *NodeName*.

Macro Examples

The following are several examples of macros. Many of these macros can be found in Chapter 32, "The Cookbook," on page 963, but they are not installed by default.

Parameter Testing: AutoFit

This macro resizes an image to fit a specified size along one dimension. For example, suppose the images in a document are 300, 250, or 166 pixels wide. No matter what size the screen snapshot, you can quickly resize it to one of the standard sizes and easily keep the aspect ratio.

```
shake uboat.iff -autofit 166 w
```

This calculates an image that is 166 x 125 pixels in size. It is not necessary to calculate the height on your own.

Here is the code:

```
image AutoFit(image img=0, int size=166, int sizeIs=1)
{
  curve w=0;
  curve h=1;
  return Resize(img,
     sizeIs ? size*width/height :size ,
     sizeIs ? size : size*height/width
  );
}
```

The first line creates the parameters. Note that calculate is expecting an integer. The next two lines assign a value to both *w* and *h*. This way, the user can type in 0 or 1 to calculate width or height, or enter *w* or *h*, which is easier to remember. Since Shake assigns values to *w* and *h*, it is able to determine what axis to calculate. The Resize function uses embedded *if/then* statements to test the *sizeIs* parameter. Also, you do not change the value of *w* or *h* during processing, so they are not declared as the curve int type of data.

Text Manipulation I: RandomLetter

This function generates a random letter up to a certain frame, at which point a static letter appears. You can select the color, switch frame, position, and size of the letter, as well as the font. This function uses the standard C *stringf* function to pick a random letter, and assigns it to the variable *rdLetter*. This is pumped into the *Text* function, which uses a conditional expression to evaluate if the current time is before the *staticFrame* time (set to frame 10 by default).

```
image RandomLetter(
  int width=720,
  int height=486,
  int bytes=1,
  const char * letter="A",
  int staticFrame=10,
  float seed=time,
  const char * font="Courier",
  float xFontScale=100,
  float yFontScale=100,
  float xPos=width/2,
  float yPos=height/2,
  float red=1,
  float green=1,
  float blue=1,
  float alpha=1
  )
```

```
{
  curve string rdLetter = stringf("%c",
     'A'+(int)floor(rnd1d(seed,time)*26));
  return Text(
  width, height, bytes,
  time < staticFrame?"{rdLetter}":"{letter}",
  font,
  xFontScale, yFontScale, 1, xPos, yPos,
  0, 2, 2, red, green, blue, alpha, 0, 0, 0, 45
  );
}
```

Text Manipulation II: RadioButton

This is an excerpt from the RadioButton function that is used to generate radio buttons for the interface. The radio button code requires four icons to support it: *name.on.nri*, *name.on.focus.nri*, *name.off.nri*, and *name.off.focus.nri*. Since it is tedious to write out four separate files, automatically change the file extensions over time with this excerpt from the macro:

```
image RadioButton(
const char *text="linear",
...
string fileName=text,
string filePath="/Documents/Shake/icons/ux/radio/",
int branch=time,
..
)
{
  curve string fileState =
     branch == 1 ? ".on.nri" :
     branch == 2 ? ".on.focus.nri" :
     branch == 3 ? ".off.nri" : ".off.focus.nri";
  curve string fileOutName = filePath + "/"+
     fileName + fileState;
...
  return FileOut(Saturation1, fileOutName);
```

Since you typically calculate this with a command such as:

```
shake -radio "HelloWorld" -t 1-4
```

it generates *HelloWorld.on.nri* at frame 1, *HelloWorld.on.focus.nri* at frame 2, *HelloWorld.off.nri* at frame 3, and *HelloWorld.off.focus.nri* at frame 4.

Text Manipulation III: A Banner

This little trick takes a string of letters and prints it, one letter at a time. It declares a variable within the string section of a *Text* node:

```
Text1 = Text(720, 486, 1,
  {{  string logo = "My Logo Here";
     stringf("%c", logo[(int) clamp(time-1, 0,strlen(logo))])
  }},
     "Courier", 100, xFontScale, 1, width/2, height/2,
     0, 2, 2, 1, 1, 1, 1, 0, 0, 0, 45);
```

This uses *strlen* to determine the length of the string and extract the letter that corresponds to the current frame.

Text Manipulation IV: Text With a Loop to Make a Clock Face

This eager little example lays out a clock face. A *for* loop is used to count from 1 to 12, and prints the number into a *Text* function with a *stringf* function. (You should just be able to print the value of count with {count}, but it doesn't work. Go figure.) The *cosd* and *sind* functions are also used to calculate the position of the number on the clock face. Keep in mind that zero degrees in Shake points east, as it does in all Cartesian math. The *falloffRadius* serves no purpose in the function except to complete the onscreen control set to give a center and a radius widget:

```
image Clock(
  image In=0,
  float xCenter=width/2,
  float yCenter=height/2,
  float radius=150,
  float falloffRadius=0
)
{
NumComp=Black(In.width,In.height,1);
for (int count=1;count<=12;++count)
{
  NumComp = Over(
     Text(In.width, In.height, 1, {{ stringf("%d",count) }},
     "Courier", radius*.25, xFontScale, 1,
  cosd(60+(count-1)*-30)*radius+xCenter,
  sind(60+(count-1)*-30)*radius+yCenter,
     0, 2, 2, 1, 1, 1, 1, 0, 0, 0, 45),
  NumComp
  );
}

  return Over(NumComp,In);
}
```

Text Manipulation V: Extracting Part of a String

This function can be used to extract the file name from a *FileOut* or *FileIn* node so you can print it on a slate. Use it in a *Text* function.

```
const char *getBaseFilename(const char *fileName)
{
  extern const char * strrchr(const char *, char);
  const char *baseName = strrchr(fileName, '/');
  return baseName ? baseName+1 : fileName;
}
```

To use it, place a line in the text parameter of a *Text* function, such as:

```
{getBaseFilename(FileIn1.imageName)}
```

Tiling Example: TileExample

This allows you to take an image and tile it into rows and columns, similar to the *Tile* node in the Other tab. However, this one also randomly moves each tile, as well as scales the tiles down. The random movement is generated with the turbulence function (see Chapter 31, "Expressions and Scripting," on page 935). Because of this, it is less efficient than the *Tile* function, which can be viewed in the *<ShakeDir>/include/nreal.h* file.

```
image TileExample(
  image In=0,
  int xTile=3,
  int yTile=3,
  float angle=0,
  float xScale=1,
  float yScale=1,
  float random=30,
  float frequency=10
)
{
result = Black(In.width, In.height, In.bytes);
curve float xFactor=(In.width/xTile)*.5;
curve float yFactor=(In.height/yTile)*.5;

for (int rows=1; rows<=yTile; ++rows){
  for (int cols=1; cols<=xTile; ++cols){

  Move2D1 = Move2D(In,
  -width/2+xFactor+((cols-1)*xFactor*2)+
  (turbulence(time+(cols+1)*(rows+1),frequency)-.5)*random,
  -height/2+yFactor+((rows-1)*yFactor*2)+
  (turbulence(time+(cols+1)/(rows+1),frequency)-.5)*random,
  angle, 1,
  1.0/xTile*xScale, 1.0/yTile*yScale
  );
  result=Over(Move2D1,result);
  }
}
return result;
```

Expressions and Scripting

31

One of the more powerful aspects of Shake is its ability to use a wide variety of expressions and script code directly within any parameter of the application.

What's in This Chapter

This chapter covers a variety of advanced topics relating to expressions and scripting directly within Shake.

The following topics are covered:

Linking Parameters

By linking variables and parameters, you can access information in any node and use it in a different node.

To link to a parameter that's within the same node:

- Type the parameter name into a parameter field, then press Return (or Enter).

For example, the *Move2D* node links the yScale parameter as equal to the xScale parameter by default. To show the expression editor, enter any letter into the value field, then press Return. A plus sign appears next to the parameter. Click the plus sign (+) to expand the parameter and enter your expression.

To link to a parameter within a different node:

- Enter the node name, followed by a period, followed by the parameter name. For example:

```
node.parameter
```

In the following example, the value parameter is linked to another value parameter from the *Fade1* node, and multiplied by 2.

You can declare a variable anywhere within a script. However, to make the variable available in the interface, you must precede it with the curve declaration:

```
curve parameter_name = expression;
```

For example, use the following to declare the *my_val* variable for the above example:

```
curve my_val = 1;
```

To clone a node in the Node View:

1. Copy a node.
2. Do one of the following:
   - Right-click in the Node View, then choose Edit > Paste Linked from the shortcut menu.
   - Press Shift-Command-V or Shift-Control-V.

This links all parameters in the cloned node to those of the original. The cloned parameter is named with the original node's name, followed by "_clone" and a number if there is more than one clone in the node tree.

Important: When you modify any parameter in a cloned node, its link to the original node is broken, so be careful when making adjustments with onscreen controls.

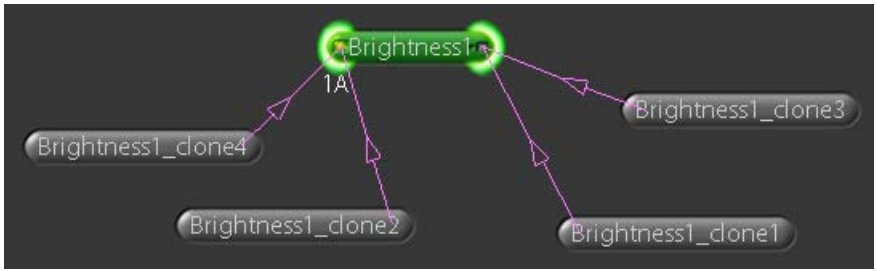
Viewing Links in the Node View

To help you make sense of what's happening in the node tree, you can view the links connecting one node to another. Links are indicated for cloned nodes, as well as for nodes that use expressions referencing a parameter within another node.

To view the links between nodes in the Node View, do one of the following:

- Right-click in the Node View, then choose Enhanced Node View from the shortcut menu.
- Press Control-E.

The links appear as lines with arrows pointing to the node that's being linked to.



Linking to a Parameter at a Different Frame

Ordinarily, links to another parameter produce that parameter's value at the current frame. If the parameter is animated and you want to obtain a value at a specific frame, or at a frame that's offset from the current time, you can use the following syntax to view a value at a different frame:

```
parameterName@@time
```

In the above syntax, *time* could be substituted with a specific frame number, or with an expression that calculates an offset from the *time* variable that references the current position of the playhead. For example, the following expression obtains the value from the yPan parameter of the *Move2D1* node at the previous frame:

```
Move2D1.yPan@@(time-1)
```

Variables

You can declare your own variables, as shown above. However, each image node also carries other information about that node, using the width, height, and bytes variables. When you refer to these, they work exactly the same as above.

For example, the default center of rotation on *Move2D* is set to:

```
width/2
```

This places the center at the midpoint of the current image.

When referring to a variable from a different node, place the node name before the variable:

```
node_name.width
```

In some cases, problems may occur. For example, in a *Resize* node, if you set the *Resize* equal to *width/2*, you potentially cause a loop because it is changing width based upon a value that is looking for the width. To solve this, Shake always refers to the input width, height, and bit depth when you refer to these from inside of that node. Therefore, *width/2* takes the incoming width and divides it by 2. When you refer to the width, height, and bit depth of a different node, you use that node's output values.

At any time, you can also use the variable *time*, which refers to the current frame number. For example, enter *cos(time/5)*50* in the angle parameter value field in a *Rotate* node for a nice "rocking" motion.

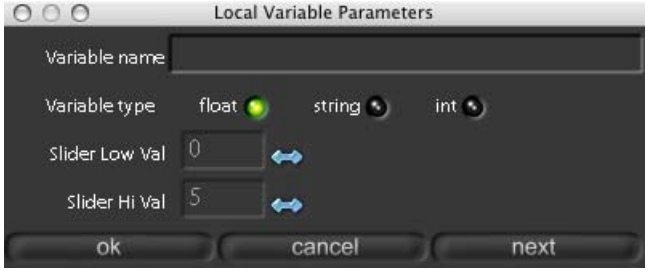
> ***Warning:*** When renaming a node, don't use a name that's also used by a local variable within that node.

Creating and Using Local Variables

You can create additional parameters of your own, with extra sliders and text entry fields, in order to build more complex expressions with interactive input.

To create a local variable within a node in your script:

1. Pick the node you want to add a local variable to, and open its parameters into the Parameters tab.
2. Right-click anywhere within the Parameters tab (not directly on a field), then choose Create Local Variable from the shortcut menu.
3. When the Local Variable Parameters window appears, define the settings that variable will be created with.



- *Variable name:* The name for that variable that appears in the Parameters tab.
- *Variable type:* Whether the variable is a float (decimal places of precision), a string (alpha-numeric text), or an integer (whole numbers).

- *Slider Low Val:* For float and int variables, the lowest value the slider will represent.
- *Slider Hi Val:* For float and int variables, the highest value the slider will represent.

4 When you're done, click OK to create the variable and go back to your project, cancel to close the window without creating a new variable, or next to continue creating new variables.

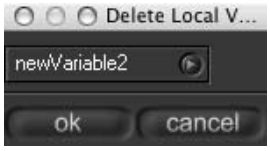
New local variables that you create appear within a subtree at the bottom of the other node parameters. This subtree appears only after at least one new variable has been created, and is named localParameters.

Once created, your own local variables can be used and referenced by expressions just like any other parameter in Shake.

To remove a local variable:

1 Right-click anywhere within the Parameters tab (not directly on a field), then choose Delete Local Variable from the shortcut menu.

2 When the Delete local variable window appears, choose a local variable to delete from the pop-up menu, then click OK.



For a tutorial on using local variables in expressions, see Tutorial 4, "Working With Expressions," in the *Shake 4 Tutorials*.

Expressions

The following section provides examples of using expressions (which can help out the lazy compositor by doing your work for you). In any parameter, you can combine any value with a math expression, trigonometry function, an animated curve, a variable, or even a conditional expression.

For example, as mentioned above, the center of an image can be found by using:

```
xCenter = width/2
yCenter = height/2
```

These take the per-image width and height variables and divide them by 2.

You can type an expression in any field. Some nodes, such as *ColorX*, *WarpX*, and *TimeX*, even support locally declared variables. For more information and a list of examples, see "ColorX" on page 647.

If you are using the command-line method, you may have to enclose your expressions in quotes to avoid problems with the operating system reading the command. For example, don't use:

```
shake my_image.iff -rot 45*6
```

Instead, use:

```
shake my_image.iff -rot "45*6"
```

Examples	Explanation
1/2.2	1 divided by 2.2. Gives you the inverse of 2.2 gamma.
2*Linear(0,0@1,200@20)	Multiplies the value of an animated curve by 2.
2*my_curve	Multiplies a variable by 2.
sqrt(my_curve-my_center)/2	Subtracts *my_center* from *my_curve*, takes the result square root, and then divides by 2.
time>20?1:0	If time is greater than 20, then the parameter is 1, otherwise it equals 0.
cos(time/5)*50	Gives a smooth ping-pong between -50 and 50.

Precedence

The above operators are listed in order of precedence—the order Shake evaluates each operator, left to right. If this is difficult to keep up with (and it is), make liberal use of parentheses to force the order of evaluation. For instance:

```
a = 1 + 2 * 4 -2
```

This expression does "2*4" first, since the "*" has precedence over "+" and "-" which gives you "a=1+8-2." Then from left to right, Shake does "1+8," giving "a=9-2," finally resulting in "a=7." To add and subtract before multiplying, use parentheses to control the evaluation.

```
a = (1 + 2) * (4 - 2)
```

This results in "a=3*2" or "a=6."

Note: In any expression, parentheses have the highest precedence.

Reference Tables for Functions, Variables, and Expressions

All of the math functions available in Shake can be found in the *include/nreal.h* file. You can declare your own functions in your own .h file.

To set an expression on a string (text) parameter, you need to add a : (colon) at the start of the expression; otherwise, it is treated as text rather than compiled and evaluated.

Arithmetic Operators	Definition
*	Multiply
/	Divide
+	Add
-	Subtract

Relational Operators	Definition
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal to
!=	Not equal to

Logical Operators	Definition
&&	And
||	Or
!	Not

Conditional Expression	Definition
expr1?expr2:expr3	If expr1 is true (non-zero), then to expr2, else do expr3.

Global Variables	Definition
time	Current frame number.

The following table shows variables that are carried by each node.

Image Variables	Definition
parameterName	Value of *parameterName* from inside of that node.
nodeName.parameterName	Value of *parameterName* in nodeName from outside of that node.
parameterName@@time	Allows you to access a value at a different frame. For example: *Blur1.xPixel@@(time-3)* looks at the value from 3 frames earlier.

Image Variables	Definition
bytes	The number of bytes in that image. This takes the input bit depth when called from inside of the node, and the output bit depth when called from outside of the node.
width	Width of the image. Takes the input width when called from inside of the node, and the output width when called from outside of the node.
height	Height of the image. Takes the input height when called from inside of the node, and the output height when called from outside of the node.
_curImageName	Returns the name of the actual file being used for the current frame. Useful when plugged into a *Text* node: *{FileIn1._curImageName}*
dod[0], dod[1], dod[2], dod[3]	The variable for the Domain of Definition (DOD): xMin, yMin, xMax, yMax, respectively.

The following table shows channel variables used in nodes such as *ColorX*, *LayerX*, *Reorder*, etc. Check the documentation for specific support of any variable.

In-Node Variables	Definition
nr, ng, nb, na, nz	New red, green, blue, alpha, Z channel.
r, g, b, a, z	Original red, green, blue, alpha, Z channel.
l	Luminance channel for *Reorder*.
n	Null channel. Strips out the alpha in *Reorder* when used like this: *rgbn*
r2, g2, b2, a2, z2	Second image's channel for *LayerX*.

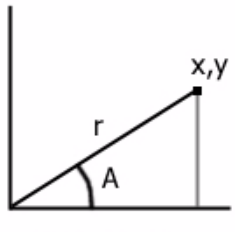
Math Functions	Definition
abs(x)	Integer absolute value. abs(-4) = 4. Be careful, as this returns an integer, not a float. Use *fabs* for float.
biasedGain(value, gain, bias)	Gives a *ContrastLum*-like curve that gives roll-off between two values.
cbrt(x)	Cubic root. cbrt(8) = 2
ceil(x)	Truncates to next integer. ceil(5.3) = 6
clamp(x, lo, hi)	Clamps x to between lo and hi. clamp(1.5,0,1) = 1
exp(x)	Natural exponent. exp(0) = 1
fabs(x)	Float absolute value. fabs(-4.1) = 4.1
floor(x)	Truncates to next lowest integer. floor(5.8) = 5
fmod(x,y)	Float modulus. Returns the remainder in float. fmod(11.45,3) = 2, for example, (3x3+2.45 = 11.45)
log(x)	Natural log. log(1) = 0

Math Functions	Definition
log10(x)	Returns base 10 log. log10(10) = 1
M_PI	A variable set to pi at 20 decimal places.
max(a,b)	Returns maximum between a and b. max(5,10) = 10
max3(a,b,c)	Returns maximum between a, b, and c. max3(5,2,4) = 5
min(a,b)	Returns minimum between a and b. min(5,10) = 5
min3(a,b,c)	Returns minimum between a, b, and c. min3(5,2,4) = 2
a%b	Modulus. 27%20 = 7
pow(x,y)	Returns x to the y power. pow(2,4) = 16
round(x)	Rounds number off. Values below x.5 are rounded to x, values equal to or above x.5 are rounded to x+1. round(4.3) = 4
sqrt(x)	Square root. sqrt(9) = 3

Noise Functions	These are ideal for WarpX and ColorX.
noise(seed)	1-dimensional cubic spline interpolation of noise.
noise2d(seed,seed)	2d noise.
noise3d(seed,seed,seed)	3d noise.
noise4d(seed,seed,seed,seed)	4d noise.
lnoise(seed)	1d linear interpolation of noise.
lnoise2d(seed,seed)	2d noise.
lnoise3d(seed,seed,seed)	3d noise.
lnoise4d(seed,seed,seed,seed)	4d noise.
fnoise(x,xScale)	1d fractal noise based on noise().
fnoise2d(x,y,xScale,yScale)	
fnoise3d(x, y, z, xScale, yScale, zScale)	
turbulence(x, xScale)	A cheaper, rougher version of fnoise().
turbulence2d(x, y, xScale, yScale)	Continuous 2d noise.
turbulence3d(x, y, z, xScale, yScale, zScale)	Continuous 3d noise.
rnd(seed)	Hash-based pseudo-random numbers. Non-hash based RNG (like rand() or drand48()) should not be used in Shake because they cannot be reproduced from one machine to another. Also, even on the same machine, repeated evaluations of the same node at the same time would produce different results.
rnd1d(seed, seed)	1d random value.

Noise Functions	These are ideal for WarpX and ColorX.
rnd2d(seed,seed,seed)	2d random value.
rnd3d(seed,seed,seed,seed)	3d random value.
rnd4d(seed,seed,seed, seed,seed)	4d random value.

Trig Functions (in radians)	Definition
M_PI	A variable set to pi at 20 decimal places.
acos(A)	Arc cosine in radians.
asin(A)	Arc sine.
atan(A)	Arc tangent.
atan2(y,x)	Returns the radian verifying sin(a) = y and cos(a) = x.
cos(A)	Cosine.
sin(A)	Sin.

Trig Functions (in degrees)	Definition
 sind (A) = y/r cosd (A) = x/r tand (A) = y/x atan2d(y,x) = A	Welcome back to trigonometry! For those of you who may have forgotten, here is a helpful chart for some commonly used equations.
acosd(A)	Arc cosine in degrees.
asind(A)	Arc sine in degrees.
atand(A)	Arc tangent in degrees.
atan2d(y,x)	Returns the angle verifying sin(a) = y and cos(a) = x.
cosd(A)	Cosine in degrees.
distance(x1,y1,x2,y2)	Calculates the distance between two points, (x1,y1) and (x2, y2).
sind(A)	Sin in degrees.
tand(A)	Tangent in degrees.

String Functions	Definition
stringf("xyz", ...)	Since you basically can write books on this, here is an example. Otherwise, it is recommended to purchase a book on C. There are also several examples in Chapter 30, "Installing and Creating Macros," on page 905. This example takes the *scriptName* parameter and uses the system function echo to print it: *extern "C" int system(const char*);* *const char *z= stringf("echo %s",scriptName);* *system(z);*
printf("xyz", ...)	
strlen("mystring")	Returns the length of the string.
strsub(*const char *string,* *int offset,* *int length*	Extracts a string from another string.

Curve Functions

The curve functions with implicit time (Linear, CSpline, and so on.) all assume that time is the first argument, so the following statements are identical:

```
LinearV(time,0,1@1,20@20)
Linear(0,1@1,20@20)
```

You can, however, adjust the time value explicitly with the V version of each curve type. For more information on spline types, see "More About Splines" on page 316.

These are the cycle type codes:

- 0 = KeepValue
- 1 = KeepSlope
- 2 = RepeatValue
- 3 = MirrorValue
- 4 = OffsetValue

Curve Functions	Definition
biasedGain(x,gain,bias)	Gives a smoothly-ramped interpolation between 0 and 1, similar to Shake's contrast curve. Gain increases the contrast, and bias offsets the center.
Linear(cycle, value@key1, value@key2, ...)	Linear interpolation from value at keyframe 1 to value at keyframe 2, and so on.
LinearV(time_value, cycle, value@key1, value@key2, ...)	Linear interpolation from value at keyframe 1 to value at keyframe 2, and so on.

Curve Functions	Definition
CSpline(cycle, value@key1, value@key2, ...)	Cardinal-spline interpolation, also known as Catmull-Rom splines.
CSplineV(time_value, cycle, value@key1, value@key2, ...)	Cardinal-spline interpolation, also known as Catmull-Rom splines.
JSpline(cycle, value@key1, value@key2, ...)	Jeffress-spline interpolation.
JSplineV(time_value, cycle, value@key1, value@key2, ...)	Jeffress-spline interpolation.
NSpline(cycle, value@key1, value@key2, ...)	Natural-spline interpolation.
NSplineV(time_value, cycle, value@key1, value@key2,...)	Natural-spline interpolation.
Hermite(cycle, [value,tangent1,tangent2]@key1, [value,tangent1,tangent2]@key2, ...)	Hermite-spline interpolation.
HermiteV(time_value, cycle, [value,tangent1,tangent2]@key1, [value,tangent1,tangent2]@key2, ...)	Hermite-spline interpolation.

The following expressions provide functions for curve analysis.

Function	Definition
getCurveMinMax(int minOrMax, int begFrame, int endFrame, float curveCurrentValue, const char *curveName);	Float. Returns the min or max value of the specified curve (plug) over the specified frame range. If minOrMax is set to 0, then it returns the min value. If minOrMax is set to 1, then it returns the max value.
getCurveAvg(int begFrame, int endFrame, float curveCurrentValue, const char *curveName);	Float. Returns the average value of the specified curve over the specified frame range.

Using Signal Generators Within Expressions

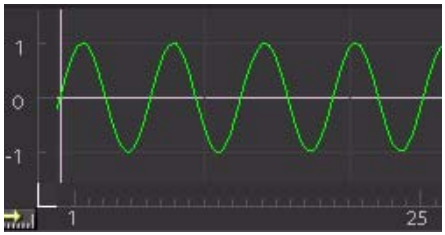
This section illustrates the use of the various signal generators that are available for Shake expressions. They can be used to create either predictable or random patterns of values, and mathematically customized to adjust their offset, frequency, and amplitude.

Signal Generators

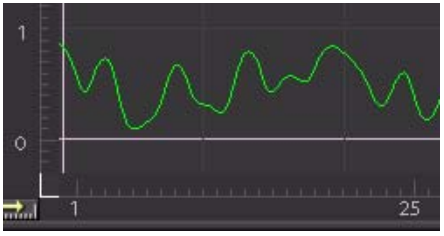
The following noise and trig functions all generate changing values over time. To animate a parameter using these functions , you supply the variable *time* for the function to operate upon.

Note: You can copy and paste most of the expressions found in this section into your own scripts for easy use.

cos(time)

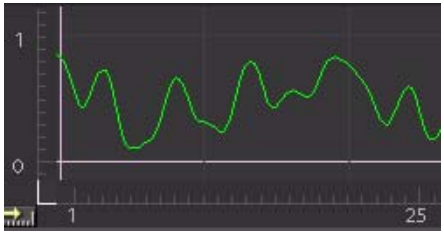


sin(time)

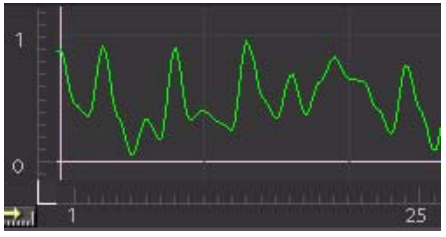


noise(time)

lnoise(time)

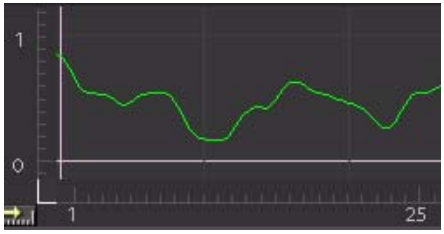


fnoise(time,1)

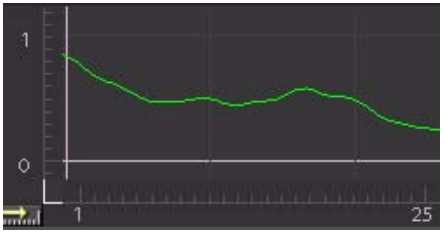


turbulence(time,1)

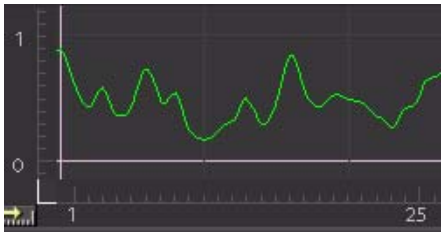
fnoise() and *turbulence()* have additional frequency factors to the noise.



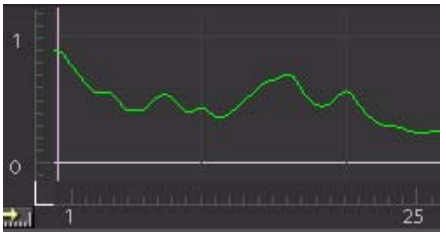
fnoise(time,2)



fnoise(time,5)



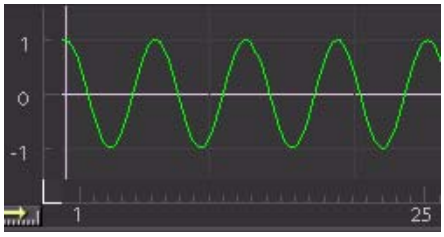
turbulence(time,2)



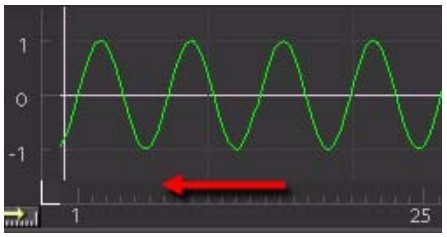
turbulence(time,5)

Offsetting a Generator Function

To offset a function's starting value, add a value to *time*.



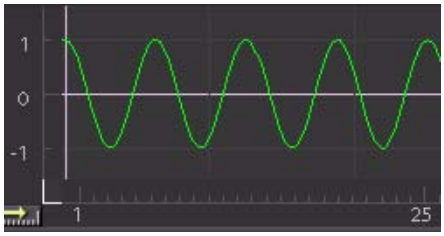
cos(time)



cos(time+10)

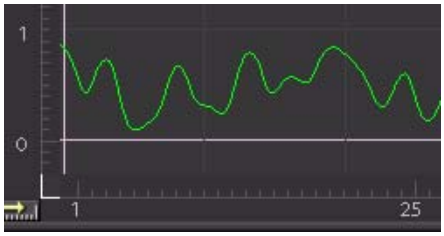
Changing the Frequency of a Generator Function

To change a function's frequency, multiply or divide *time* by a value.The exceptions are the noise functions *fnoise()* and *turbulence()*—both of which have frequency controls of their own (values are not modified below 1, so you may still have to modify *time*).

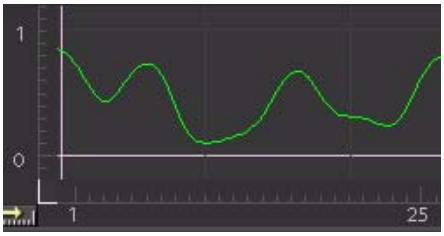


cos(time)

cos(time/3)



noise(time)



noise(time/2)

Frequency and Continuous Versus Discontinuous Noise

The functions *noise*, *lnoise*, *fnoise,* and *turbulence* are continuous noise generators, meaning you can draw a continuous smooth curve between the values. The following two expressions provide examples:

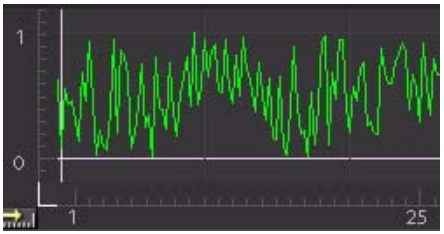
Function	Result
`noise(1)`	Is equal to .554
`noise(1.1`	is equal to .5182

As a result, you can safely predict that *noise(1.05)* is near and probably between these two values (in fact, it equals .5358).

The *rnd()* function creates discontinuous noise. The following two expressions provide examples:

Function	Result
`rnd(1)`	Is equal to .4612
`rnd(1.1`	is equal to .4536

You might have guessed that *rnd(1.05)* is between those, but it in fact equals .0174, not .458. This is why it is called discontinuous noise. Examining the neighboring values does not help you to arrive at a safe guess for the in-between values. For this reason, frequency changes have no practical effects on the curve.



rnd(time)

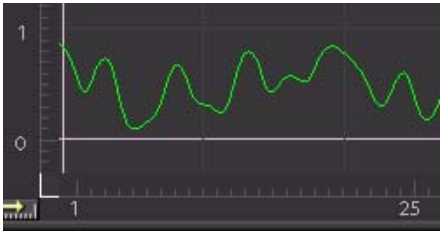
Setting Ranges for Expressions

The noise generators return values between 0 and 1. *sin()* and *cos()* return values between -1 and 1. To adjust the range, use addition and multiplication. For example, suppose you want to spit out values between 100 and 400. To make a noise generator do that, subtract the low value from the high value. This is your multiplier. Then add the lower value of your range. Thus, you have:

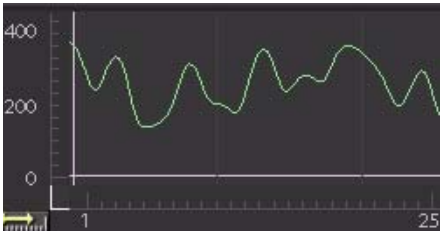
```
noise(time)*300+100
```

As *cos* and *sin* return values between -1 and 1, you have to offset the output by 1 and multiply by half of the difference between the two:

```
(sin(time)+1)*150+100
```



noise(time)

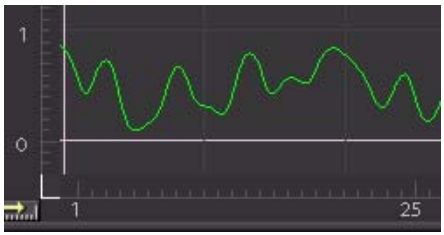


noise(time)*300+100

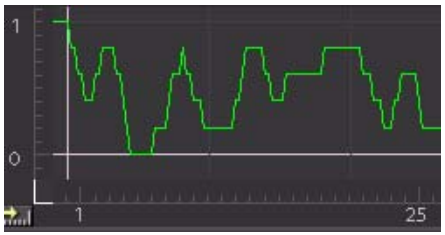
Modifying Noise

You can use other functions like *ceil()*, *floor()*, or *round()* to help you break a curve into steps. Ceiling pushes a number to the next highest integer, floor drops it to the next lowest, and round rounds off the value.

In this example, to break a *noise()* function into 5 steps between 0 and 1, multiply the value by 6 (float values of 0 to 6), knock off the decimal places with a *floor()* function (returning values of 0, 1, 2, 3, 4, 5), and then divide by 5, returning values of 0, .2, .4, .6, .8, and 1.

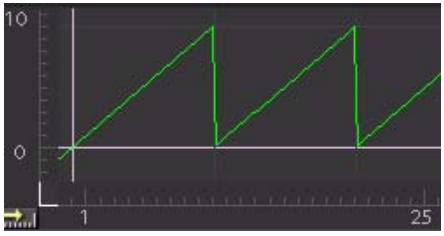


noise(time)



floor(noise(time)*6)/5

Another helpful expression is modulus, written as *a%b*. This divides *a* by *b* and returns only the remainder. This is helpful to fit an infinite range of values into a repeating limit.



time%10:

A good application of modulus is if you have an angle of any potential amount (for example, 4599) but you need to fit it into a range of 0 to 360. You would use *angle%360,* which equals 279.

Script Manual

When Shake saves a script, it creates a file that essentially uses the C programming language. This makes the product open and flexible, as you can add your own functions, or use programming structures to procedurally perform what would otherwise be tedious operations. You can also quickly make minor modifications that might be cumbersome to do in the Shake interface (for example, changing a *FileIn* to read *BG2.iff* instead of *BG1.rla*). This section of the guide explains the basic principles of this scripting language, and how they can be manipulated to create your own macros.

The examples in "Attaching Parameter Widgets" on page 919, provide step-by-step user interface scripting examples.

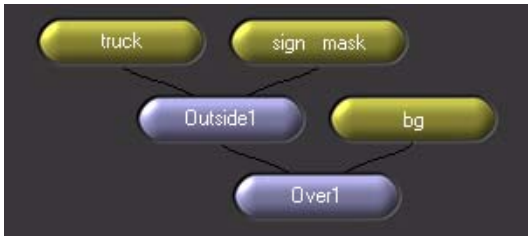
See Tutorial 8, "Working With Macros," in the *Shake 4 Tutorials* for information on making macros interactively or in a script.

Scripting Controls

To generate a test script, go to the *Tutorial_Misc/truck/* directory within the *Tutorial_Media* directory. Enter the following command to create a script and save it as *start.shk*:

```
shake truck.iff -outside sign_mask.iff -over bg.iff -savescript
     start.shk
```

The truck is composited over the background, with the sign mask as a holdout mask, and a script named *start.shk* is saved. The composite itself is unimportant, but the tree structure is important. The following image shows how the script appears in the interface.



To test the script result from the command line, type the following:

```
shake -script start.shk
```

The image of the truck is composited over the street image. The following is the code for the script itself:

```
SetTimeRange("1");
SetFieldRendering(0);
SetFps(24);
SetMotionBlur(1, 1, 0);
SetQuality(1);
SetUseProxy("Base");
SetProxyFilter("default");
SetPixelScale(1, 1);
SetUseProxyOnMissing(1);
SetDefaultWidth(720);
SetDefaultHeight(486);
SetDefaultBytes(1);
SetDefaultAspect(1);
SetDefaultViewerAspect(1);
SetTimecodeMode("24 FPS");
SetDisplayThumbnails(1);
SetThumbSize(15);
SetThumbSizeRelative(0);
SetThumbAlphaBlend(1);
```

```
// Input nodes
bg = SFileIn("bg.iff", "Auto", 0, 0);
sign_mask = SFileIn("sign_mask.iff", "Auto", 0, 0);
truck = SFileIn("truck.iff", "Auto", 0, 0);
// Processing nodes
Outside1 = Outside(truck, sign_mask, 1);
Over1 = Over(Outside1, bg, 1, 0, 0);
```

The first section contains controls for the script. The controls are all optional or can be overridden on the command line. The following sections discuss the body of the script, the sections listed under the "Input nodes" and "Processing nodes" comments.

Why "SFileIn" and Not "FileIn" Node?

The *SFileIn* node is an improvement on the older *FileIn* node. The interface button *FileIn* is linked to *SFileIn*, but the older name is retained. The *SFileIn* node allows extra subfunctions for timing to be attached.

Variables and Data Types

Shake assigns types of data to a variable name. A variable works as a sort of shopping cart to carry around your information to be used again somewhere else. The variables in the following code (excerpted from above) are bold.

The comment lines (lines starting with //) are omitted:

```
bg = SFileIn("bg.iff", "Auto", 0, 0);
sign_mask = SFileIn("sign_mask.iff", "Auto", 0, 0);
truck = SFileIn("truck.iff", "Auto", 0, 0);
Outside1 = Outside(truck, sign_mask, 1);
Over1 = Over(Outside1, bg, 1, 0, 0);
```

The above code assigns three variables (*bg, sign_mask, truck*) to three *SFileIn* nodes. These are connected to an *Outside* node and then to an *Over* node, which are also assigned variable names, *Outside1* and *Over1*.

Two Ways to Load a Script Into the Interface:

- Save the script and load it into the interface with the Terminal command *shake start.shk,* or click the Load button in the interface.
- Copy the script as text from the HTML browser/text editor and paste it into the Node View (press Command-V or Control-V). When you paste it into the interface, it points out that the filepaths are local, and probably not correct in terms of the location of the images and where the interface expects to find the images. Therefore, browse to the directory that contains the images, then click OK.

A good reason to use a script is that you can quickly set the path for the images by using copy and paste functions in a text editor.

Because the above script was created with a local filepath (in the *Tutorial_Media/truck* directory), the images can only be found if the script is run from the truck directory. For example, a local directory for the truck image might be:

/myMachine/documents/truck/truck.iff

In order to run the script from any location—so the images can be found regardless of the location of the saved script—the absolute path of the images is required. The following is an example of an absolute directory for the same image (truck):

/Server02/VolumeX/Scene12/truck/truck.iff

To reset the filepaths to an absolute path:

1. To determine the absolute path of the images, go into the directory into which you copied the tutorial media (using the command line) and type the Present Working Directory command:

```
pwd
```

2. Enter the filepath before the image names to make the path absolute. Shake does not prompt you to find the absolute path each time you paste it into the interface. Changes are in bold.

```
bg = SFileIn("/Server02/VolumeX/Scene12/truck/
  bg.iff", "Auto", 0, 0);
sign_mask = SFileIn("/Server02/VolumeX/Scene12/truck/
  sign_mask.iff", "Auto", 0, 0);
truck = SFileIn("/Server02/VolumeX/Scene12/truck/
  truck.iff", "Auto", 0, 0);
Outside1 = Outside(truck, sign_mask, 1);
Over1 = Over(Outside1, bg, 1, 0, 0);
```

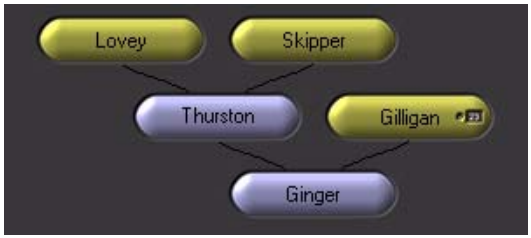
The left side is the variable name. You can change the variable name, but you must change the names wherever they appear.

3. Change the variable names:

```
Gilligan = SFileIn("/Server02/VolumeX/Scene12/truck/
  bg.iff", "Auto", 0, 0);
Skipper = SFileIn("/Server02/VolumeX/Scene12/truck/
  sign_mask.iff", "Auto", 0, 0);
Lovey = SFileIn("/Server02/VolumeX/Scene12/truck/
  truck.iff", "Auto", 0, 0);
Thurston = Outside(Lovey, Skipper, 1);
Ginger = Over(Thurston, Gilligan, 1, 0, 0);
```

Therefore, the left side of the above lines is the variable that you assign a value. The right side is the value, usually coming from a function such as *SFileIn*, *Outside,* or *Over*. The function is called by its name, followed by its arguments between parentheses. The arguments, called parameters, are very specifically organized and always specify the same thing. For example, the third parameter of *Outside* is 1—a numeric code to determine whether you take the foreground or background resolution (see "*Outside*" on page 466). The line ends with a semicolon.

4 Feed the modified script back into the Shake interface.



The following steps illustrate how variables can help you. Insert the *Mult* color-correction node to change the truck color. The script format for *Mult* (in "*Mult*" on page 644) looks like the following:

```
image Mult(
  image In,
  float red,
  float green,
  float blue,
  float alpha,
  float depth
);
```

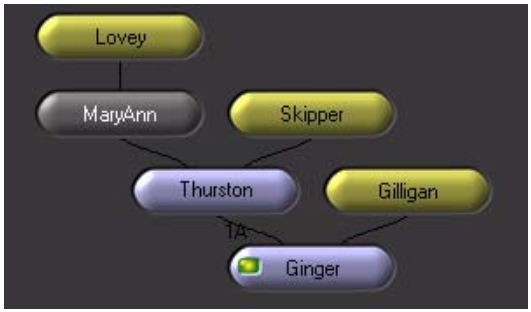
To add the *Mult* node, assign a variable name (here, *MaryAnn*) and then the parameters. Since *Lovey* is the variable name of the truck, it is fed in as the image. The numbers turn the truck yellow. Since *Thurston* (the *Outside* node) previously loaded the *Lovey* node, switch it to *MaryAnn*, the new *Mult* node.

Note: The premultiplied state of the truck image is ignored for this example.

5 Insert the *Mult* node into the script:

```
Gilligan = SFileIn("/Server02/VolumeX/Scene12/truck/
  bg.iff", "Auto", 0, 0);
Skipper = SFileIn("/Server02/VolumeX/Scene12/truck/
  sign_mask.iff", "Auto", 0, 0);
Lovey = SFileIn("/Server02/VolumeX/Scene12/truck
  truck.iff", "Auto", 0, 0);
MaryAnn = Mult(Lovey, 1, 1, .2);
Thurston = Outside(MaryAnn, Skipper, 1);
Ginger = Over(Thurston, Gilligan, 1, 0, 0);
```

The result appears as follows in the interface:



Only four parameters are entered for the *Mult* node—the alpha and depth parameters are omitted. Therefore, the alpha and depth parameters default to a value of 1. You can also see that *Mult* looks for two types of data: An *image* (labeled *In*) and *floats* (labeled *red, green, blue,* and so on.). A *float* is any number that contains a decimal place, for example, 1.2, 10.00, .00001, or 100,000.00. The following table lists the five types of data.

Data Type	Description
image	An image.
float	A number with a decimal place, such as .001, 1.01, or 10,000.00.
int	Integer. A number with no decimal place, such as 0, 1, or 10,000.
string	"this is a string"
curve float or curve int	A special case of float or int that designates that the number has the possibility of being changed during script execution or when tuning it in the interface.

Shake is usually smart enough to convert an integer into a float, so when the following is entered:

```
MaryAnn = Mult(Lovey, 1, 1, .2);
```

Shake does not freak out that you enter two integers (1, 1) for the red and green multipliers. However, this is an infrequent case concerning the conversion of floats and integers, as you cannot put in a string or an image for those arguments. For example:

```
MaryAnn = Mult(Lovey,"1", Gilligan, .2);
```

does not work because you are drastically mixing your data types (plugging an image into a float). There is one exception to this: when you enter 0 as an image input, indicating that you do not want to designate any image input.

To Designate That a Function Has No Image Input

Place a 0 in the image position. This example has no image input for the background image, the second argument:

```
MaryAnn = Over(Thurston, 0);
```

So far, you have only assigned variables to image types. In this next example, create a float variable so you can multiply the truck image by the same amount on the red, green, and blue channels. Call this variable *mulVal*.

Note: For these examples, you must save the script and use Load or Reload Script—the Copy/Paste does not work properly.

To create a float variable to multiply an image by equal amounts in the red, green and blue channels:

1 Create a script variable and plug it into the *Mult* node:

```
float mulVal = .6;
Gilligan = SFileIn("/Server02/VolumeX/Scene12/
  truck/bg.iff", "Auto", 0, 0);
Skipper = SFileIn("/Server02/VolumeX/Scene12/truck
  sign_mask.iff", "Auto", 0, 0);
Lovey = SFileIn("/Server02/VolumeX/Scene12/truck
  truck.iff", "Auto", 0, 0);
MaryAnn = Mult(Lovey, mulVal, mulVal, mulVal);
Thurston = Outside(MaryAnn, Skipper, 1);
Ginger = Over(Thurston, Gilligan, 1, 0, 0);
```

Mult now takes .6 for its red, green, and blue parameters. To change all three, modify *mulVal* on the first line of the script and execute the script again. This is swell. However, here is a problem. The variable *mulVal* is only applied when you load the script—Shake does not retain it for later use. When the script is loaded (with Load Script, not Copy/Paste) into the interface, the *Mult* parameters all read .6, not *mulVal*. There is no place in the interface that you can find the *mulVal* parameter and modify it and have the *Mult* take the argument. If you are immediately executing the script, this is not a problem. If you are loading the script and are going to interactively edit it, this is a problem. You therefore must declare *mulVal* as a curve float, a special type of float that tells Shake to wait until frame calculation to resolve the variable.

2 Convert the *mulVal* variable to a changeable curve type:

```
curve float mulVal = .6;
Gilligan = SFileIn("/Server02/VolumeX/Scene12/doc/pix
  truck/bg.iff", "Auto", 0, 0);
Skipper = SFileIn("/Server02/VolumeX/Scene12/truck/
```

```
  sign_mask.iff", "Auto", 0, 0);
Lovey = SFileIn("/Server02/VolumeX/Scene12/truck/
  truck.iff", "Auto", 0, 0);
MaryAnn = Mult(Lovey, mulVal, mulVal, mulVal);
Thurston = Outside(MaryAnn, Skipper, 1);
Ginger = Over(Thurston, Gilligan, 1, 0, 0);
```

3 Load the script into the interface with Load Script.

4 Open the local Parameters subtree in the Globals tab to reveal the *mulVal* slider.

In short, if you load a script to be modified in the interface, you probably want to declare it as a curve type of data. If you are going to execute the script on the command line and it is not going to change (that is, it is not animated), you can omit the curve data type. There are some examples in the following sections.

Functions

Shake is a collection of functions. Some functions modify images, such as *Blur*. Some return a number, like *distance()* (calculates the distance between two points) or *cosd()* (a cosine function in degrees, not radians). Other functions build the interface, such as *nuiToolBoxItem*. The *nuiToolBoxItem* function loads a button into the interface and attaches a function to the button. When you call a function, you assign a variable to it.

The following example reads the angle parameter for the *Rotate1* node, and returns the cosine in degrees, assigning it to the variable named *myCos*:

```
curve float myCos = cosd(Rotate1.angle);
```

You can also use functions inside of other functions. In this example, the *cosd* function is inside of a *Rotate* function:

```
Rotate1 = Rotate(FileIn1, cosd(45));
```

When you place a function inside of another, it is called a "nested function." In both cases, you call the function, and then enclose its parameters in parentheses. When you assign a variable to a function, terminate the line with a semicolon, as shown in both of the examples above. Everything in between the parentheses can be formatted however you want, so the following two examples are identical:

```
Mult1 = Mult(FileIn1, 1,1,1);
and
Mult1 = Mult(
  FileIn1,
  1,
  1,
  1
);
```

Function Formats

So, where are all of these functions and how do you find their formats? Typically, they are organized by the type of data they manipulate.

Here is a handy way to get a function format: Create the node(s) in the interface, select and copy the node(s) (press Command-C or Control-C), and paste the nodes into a text editor.

For more information:

- For image functions, see their relative chapters. For example, for information on the *Rand* function, see "*Rand*" on page 602.
- For mathematical functions, see "Expressions" on page 939.
- For Shake settings, see "Setting Preferences and Customizing Shake" on page 355.
- For interface building functions, see "Setting Preferences and Customizing Shake" on page 355.
- Examples abound in *<ShakeDir>/include/nreal.h* and *<ShakeDir>/include/nrui.h*, as well as in Chapter 32, "The Cookbook," on page 963.

Script Comments

To temporarily comment out lines in a script, use the following symbols:

```
# This line is commented out
// This line is also commented out
This is not commented out //But this is
/*
All of
these lines
are
commented out
*/
```

Conditional Statements

Like in C, you can use conditional statements in your scripts. These are particularly useful in macros, where a conditional parameter is input by the user. Keep in mind that the conditional statements require you to work in the script and cannot be built with interface tools.

Conditional Expression

To simply switch a parameter, use the in-parameter conditional expression *expr1?expr2:expr3*, which reads as, "if *expr1* is true, use *expr2*; otherwise, use *expr3*." For example, *time>10?0:1* reads as, "if time is greater than 10, then set the value to 0; otherwise, set it to 1."

In the interface, you can also use the *Select* node to switch between any number of input nodes. (For more information on the *Select* node, see "*Select*" on page 471.) This strategy allows you to stay within the interface without resorting to the script. However, the difference is that Shake only builds the part of the script that it needs for conditional statements listed below. *Select* has the overhead of all of its input nodes.

Finally, ++*i* or --*i* is supported for increments. *i*++ is as well, but returns a warning message.

If/Else

This is used to express decisions. If the test expression is true, then the first statement is executed. If false (and an "else" part exists), then the second statement is executed. If "else" is left off, Shake does nothing and moves on.

```
if (expression evaluates to non-zero) {
  do_this
} else if {
  do_this
} else if {
  do_this
} else {
  do_this
}
```

or just

```
if (expression evaluates to non-zero) {
  do_this
} else if {
  do_this
}
```

For

Normally, the three expressions of the "for" loop are an initialization (a=1), a relational test (a<10), and an increment (a++): for (a=1; a<10; a++). So "a" is set to one, checked if it is less than 10, then if that is true, a statement performed, and "a" is incremented by one. The test is checked again and this cycle continues until "a" is found to be not less than 10 (untrue).

```
for (initialize; test; increment)
{
  do_this
}
```

While

If the given expression is true or non-zero, the following statements are performed, and the expression is reevaluated. The cycle continues until the expression becomes false.

```
while (this_expression_is_true)
{
  do_this
}
```

If there is no initialization or reinitialization, "while" often makes more sense than "for."

Do/While

This variation of "while" is different in that it tests at the bottom of the loop, so the statement body is done at least once. In the "while" above, the test may be false the first time and so nothing is done. Note on all of the other statements, a semicolon was not needed. This expression does need it at the end.

```
do {
  do_this
} while (this_expression_is_true);
```

The Cookbook

32

"The Cookbook" contains tips and techniques for Shake that don't fit neatly into other categories.

Cookbook Summary

The Cookbook contains a wide variety of techniques that have been accumulated over the years, and is provided to give you some shortcuts and ideas for different approaches to different kinds of shots. None of the methods covered in this chapter are intended to be the only way to perform these tasks—Shake's very nature makes it possible to accomplish the same things in many different ways.

Coloring Tips

The following topics cover different ways for performing color correction in Shake.

Tinting

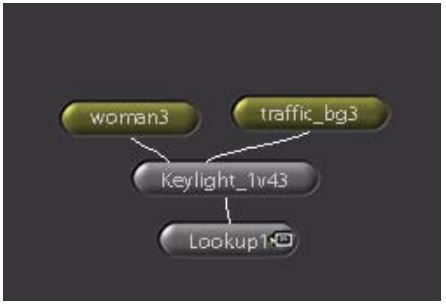
The following five techniques demonstrate the versatility of the color-correction nodes. Each one applies a tint to an image, meaning the midtones are pushed toward a certain color while leaving the blacks and whites alone. None of these techniques has any particular superiority over the others—they just illustrate several ways to do the same thing.

The following images are courtesy of Skippingstone from their short film *Doppelganger*.

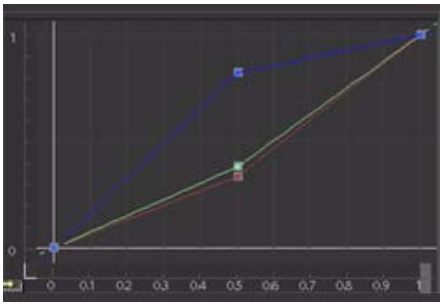
- *Brightness* + *Mult*: These nodes concatenate in the following node tree. This setup does not work well if you use a pure color in a single *Mult* node (in this case, pure blue with a value of 0,0,1) because the zeroes drop the midtones out completely on the red and green channels. The *Brightness* node is set to approximately 3, helping to maintain the red and green channels when the blue multiplier brings them back down (.3, .3, .8 in *Mult1*).

- *Monochrome* + *Brightness* + *Mult*: This is identical, except you get a purer blue color since you have made a monochrome image before applying the color correction. Note that the *Monochrome* node does not concatenate with the *Brightness* node.

- *Lookup*: This example uses a *Lookup* curve, setting the midpoints of the curve to .3, .32, .8.



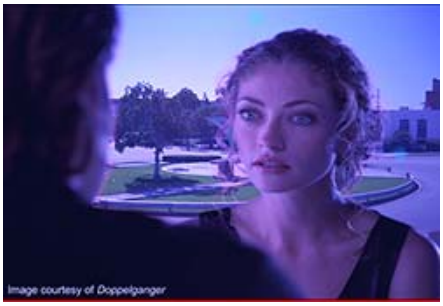
The curve looks like the following (the curves are Linear to mimic a *Tint* function from a different package):

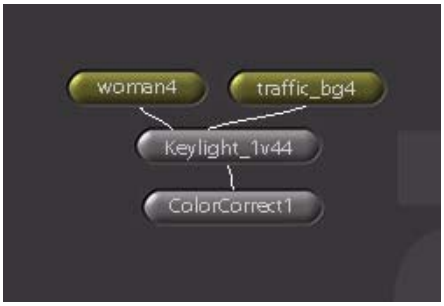


- *ColorMatch*: This is similar in theory to the *Lookup* node, as it allows you to push the lows, mids, and highs. However, the internal math helps reduce solarization (hills and peaks in the curves), so you maintain a little bit more of the input color.

You can get interesting adjustments of your values if you adjust the source points on the *ColorMatch*. (For example, hold down O and drag left on the highSource color control.)

- *ColorCorrect*: This is an Add on the Mid areas using -.2, -2, .5.





- Using *Mix*: You may end up with several nodes to achieve a particular color correction. This may be awkward to tune. Therefore, a convenient way to quickly adjust a result is to mix it back into the input image with a Layer–*Mix* node. Naturally, it is always faster to process by adjusting the original color nodes, but using a *Mix* may be easier for you to keep a handle on things.

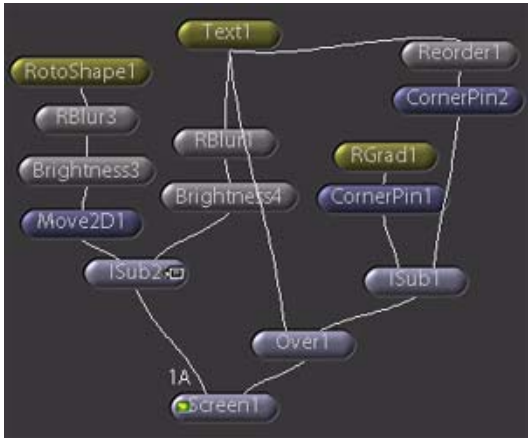
Filtering Tips

This section illustrates creative uses of Shake's filtering nodes.

Volumetric Lighting

This script can be found in *doc/html/cook/scripts/volumetric.shk*. This simple hack gives you fake volumetric lighting by using Filter–*RBlur*. One of the key principles is that *RBlur* is dog-slow, so it is better, if you can, to apply a radial blur on a low-resolution element and then scale it up. To get the volume effect, subtract one *RBlur* from another. You should also drop your quality down (to .25, for example) when testing.

In this tree, *RBlur3* is generating the main cone of light. *Move2D1* is used to scale it up, saving you a little rendering time. *CornerPin2* is used to generate the "shadow" element and place it on the ground in perspective. *RBlur1* is at full resolution to maintain the crispness of the rays.



Keying Tips

This section covers keying techniques.

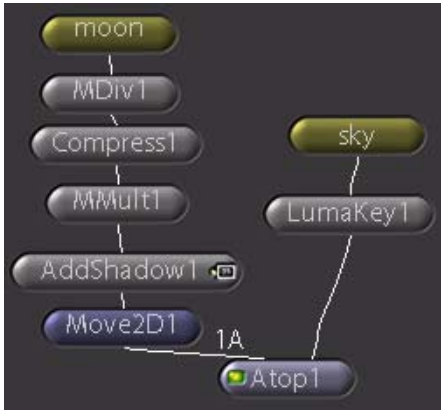
Keying Clouds

This script can be found in *doc/html/cook/scripts/clouds.shk*. The images used are the *moon.iff* and *sky.iff* files found in the *Tutorial_Media* directory.

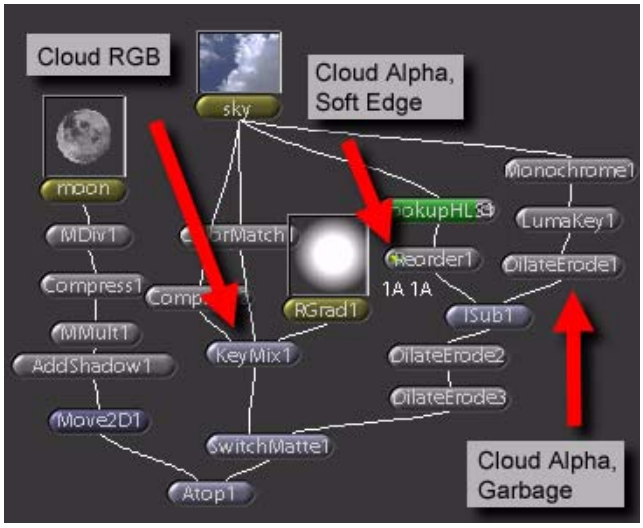
This is one potential technique for keying clouds, but may also be useful for flames. It discusses several approaches.This script puts the moon behind the clouds:

You might think a color-based key would work, but in this image nearly everything is a shade of blue, so that won't work the way you want.

Another tactic would be to use a *LumaKey* node. In this attempt, the moon is corrected (using a *Compress* node, an Other–*AddShadow* node to add the white glow, and then positioned with a *Move2D* node) to adjust the color and position of the moon. A key is pulled on the clouds with *LumaKey*, and a Layer–*Atop* is used to layer the moon only where there is alpha in the background. This displays two typical problems with this sort of keying—there's a black edge, and there is not complete opacity in the thick part of the clouds. This is not what you want.



In this next attempt, there are three main branches. The first, identical to the second attempt, manipulates the moon. The second branch, terminating in a *KeyMix* node, works on the RGB of the clouds. The *KeyMix* mixes a darkened cloud image with a brighter cloud image through the *RGrad* node, giving it the "glow" through the clouds. The third branch works on the key of the clouds. This is divided into two sub-branches, one of which pulls a soft key on the clouds, and the second of which pulls a hard key to act as a garbage mask.



Move2D1

KeyMix1

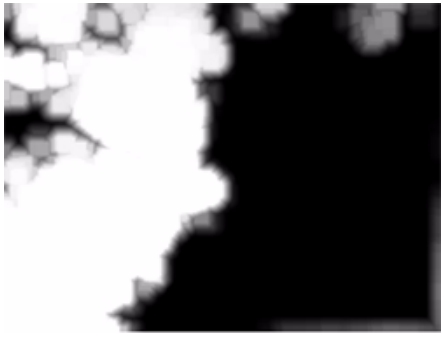
A luminance key is used here. The blue channel has less contrast than the red channel, so first insert a Color–*Monochrome* node, then boost the rWeight up and the g- and bWeight down before you key.

Red channel

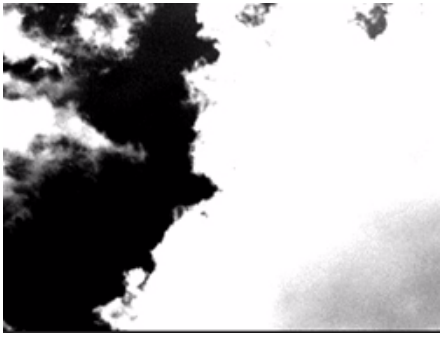
Blue channel

A Color–*LookupHLS* node manipulates the luminance, then switches that into the alpha with Color–*Reorder* (a macro begging to happen) to key the deep part of the clouds in the lower-right corner. The curve looks like the following:

The *LumaKey* gets the edges of the cloud, and is enhanced by a *DilateErode*, and then is subtracted from the soft key.



DilateErode3



LumaKey1

You still get black edges, so sprinkle Filter–*DilateErode* nodes liberally. For the nodes attached to *ISub*, the first chews into the edge, and the second *DilateErode* softens it by activating the soften parameter.

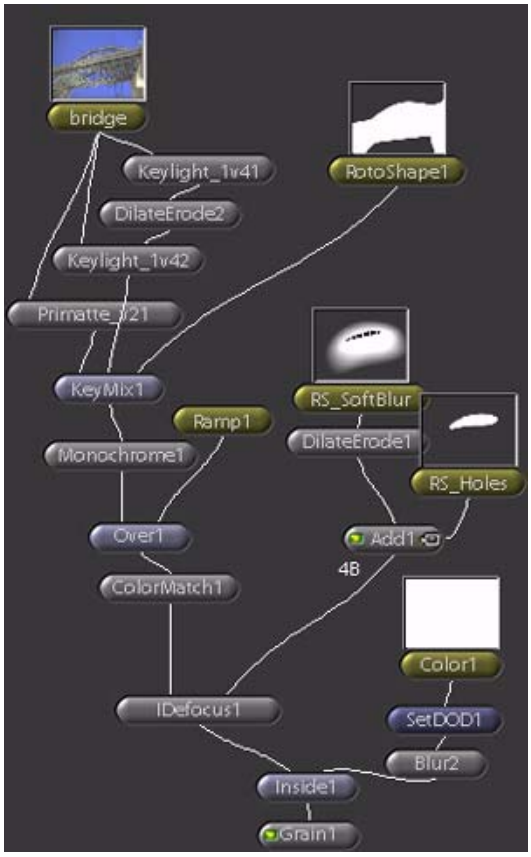
As a final touch, the x/yCenter of *RGrad* is linked to the *Move2D* x/yPan, adding an offset for the center of the moon, *Move2D1*.xPan+155, *Move2D1*.yPan+160. You can modify the position of the moon and the glow on the clouds follows.

Vignette

This script can be found in *doc/html/cook/scripts/vignette.shk*.

This example keys out the night sky, turns the scene to daylight, and adds a vignette to the borders.

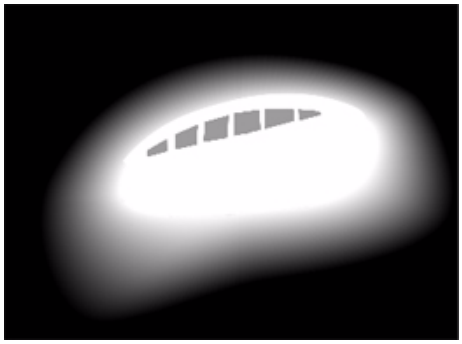
This tree involves a lot of masking. Remember, rotoscoping is your friend. (Just not a very fun friend.)



The first branch, down to *KeyMix1*, pulls a key on the sky. The first *Keylight* pulls a hard key to be fed into the second *Keylight*. This is the core key for the bridge. This is keymixed with a *Primatte*-based key for the blue area through *RotoShape1*.

Once the key is pulled, a *Monochrome* is applied and composited over a *Ramp*. This is then tinted with *ColorMatch*. A good trick is to drag the color controls while holding T down—this is for temperature and can be used to warm up or cool down a color.

The image is then defocused with a mask. This result is masked by the node *Inside1* with a square mask (*Blur2*) to get the black frame.



Add1

Blur2

Layering Tips

The following examples illustrate tips for layering.

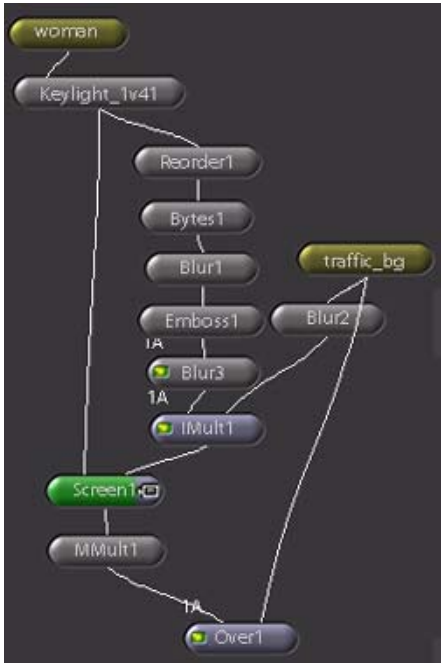
Bleeding Background Color Into the Foreground

This script can be found in *doc/html/cook/scripts/edgelight.shk*.

This script, which has an exaggerated effect for purposes of illustration, helps blend in some of the background color onto the foreground material. Note this is one variation—there are several ways to do this.

Normal composite

Composite with exaggerated rim



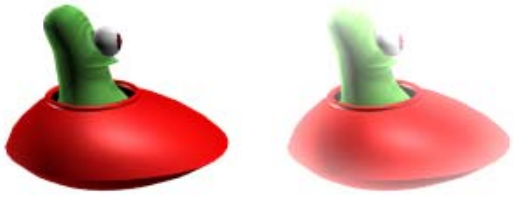
- *Keylight*: Extracts an unpremultiplied plate.
- *Reorder*: Places the alpha into the RGB.
- *Bytes*: Boosts it up to 16 bits—you are sure to get banding on the *Blur+Emboss* in 8 bits.
- *Emboss*: Extracts a sense of lighting direction. The elevation is set to 0.
- *Blur2*: If you do not blur the background, it makes her look transparent.
- *IMult1*: Blends the color into the "lit" areas. Note it is assumed that *Blur2* has an alpha of 1. If not, you must insert a Color–*SetAlpha*.
- *Screen*: You can also use an *IAdd*.

Emboss1

IMult1

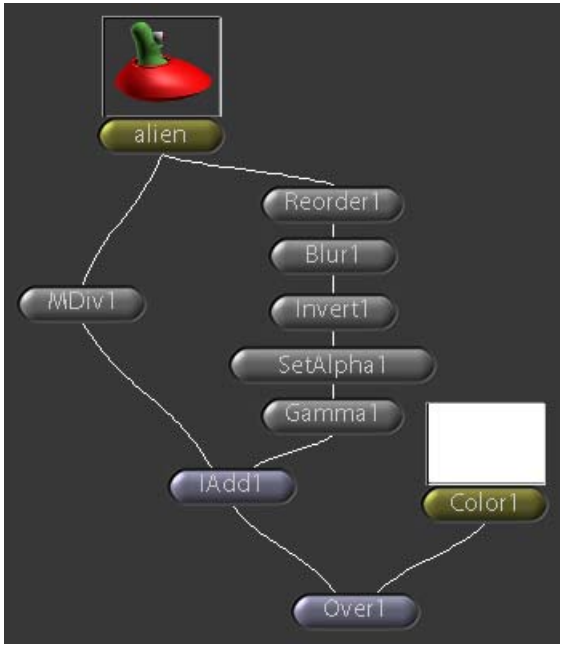
Background Flare

This script can be found in *doc/html/cook/scripts/backlight2.shk*.



This script translates D.W. Kim's fine backlighting page from http://highend3d.com into Shake terms. There are other ways to do this.

The script begins with a Color–*Reorder* that puts the alpha into the RGB channels. It then Filter–*Blurs* it, inverts that, removes the alpha channel with a Color–*SetAlpha* set to 0, and then is added back onto the plate. The Color–*MDiv* is used because the *IAdd* disrupts the premultiplication status. The Color–*Gamma* can be used to tune the intensity of the flare. Finally, preMultiply is activated in the Layer–*Over*.



Transform Tips

This section covers advanced techniques for transforming images.

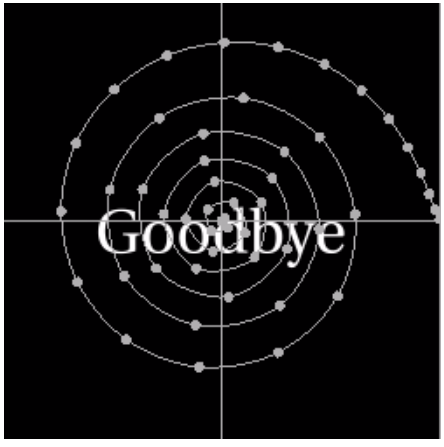
Spiral Down

This animates something in a spiral pattern, running ever smaller (or larger) concentric rings. First, right-click in the parameters area, choose Create Local Variable from the shortcut menu, then name the local variable "mul."

You control the speed toward the center and the direction by animating the mul value. To slow down the degrees in each frame, multiply time by a second localParameter, for example, freq:

```
sin(time*freq)*mul
```

If you animate freq from near 0 to 1, it looks something like the following:



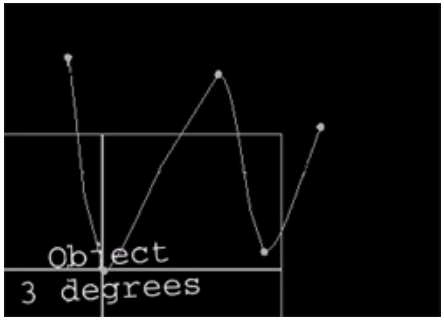
The path does not show up by default, as there are no curves.

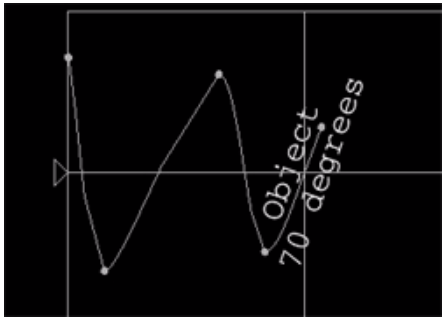
To burn the path in:

1. Set timeRange in the Globals tab (1-50, for example).
2. Right-click over the xPan expression, then choose Save Expression from the shortcut menu.
3. In the lower-right corner of the Browser is a setting for Auto or Raw. Set it to Raw.
4. Enter a name for your curve, for example, "curveX.txt."
5. Do the same thing for yPan, saving it as "curveY.txt."
6. Create a second *Pan* node.
7. Right-click, then select Load Expression on the xPan parameter.
8. Set your format in the Browser as Raw.
9. Do the same for yPan.

Auto Orient

This script can be found in *doc/html/cook/scripts/autoorient.shk*.





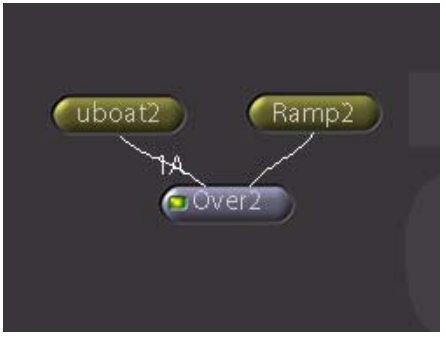
This script demonstrates how to set up a transform so that an element automatically rotates according to the tangent of the move. Although you can create this effect with one node, it is better to use two so that you can continue to modify the position without accidentally eliminating the expression to do the rotation. If you want to animate or modify the rotation, insert a second *Rotate* node—the transforms concatenate.

The rotation is determined by obtaining the position just before and just after the current node. This is done by using the @@ time notation: *(xPan@@time-.5)* returns the xPan position half a frame earlier. These coordinates are then fed into the *atan2d* function which returns the angle between two points.

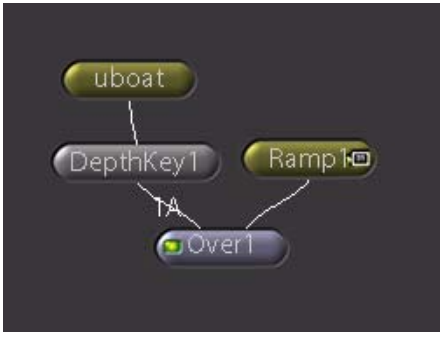
Creating Depth With Fog

The *uboat.iff* image can be found in the *Tutorial_Media* directory. The background is a simple *Ramp*. The trick is to apply depth-cueing to the U-boat:

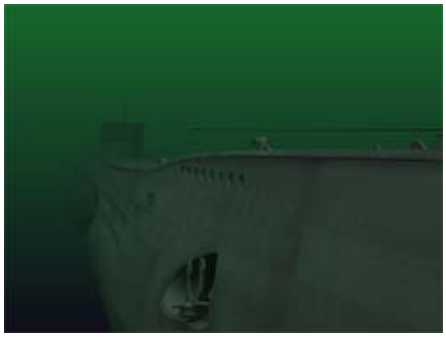
The first composite, without any fog, is not so stellar:

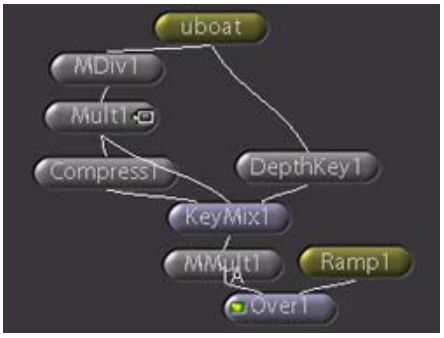


The second approach is to use Key–*DepthKey* (distance of 45) to pull a transparency key. Not so good, as you get artifacts along the edges:



This more complex approach uses the *DepthKey* as a mask for a color correction, in this case, *Compress*, which has identical hi and lo colors. A *KeyMix* is used to get the concatenation with *Mult* and *Compress*. The *Mult* is used to tint the boat green; *Compress* gives the feeling of depth:

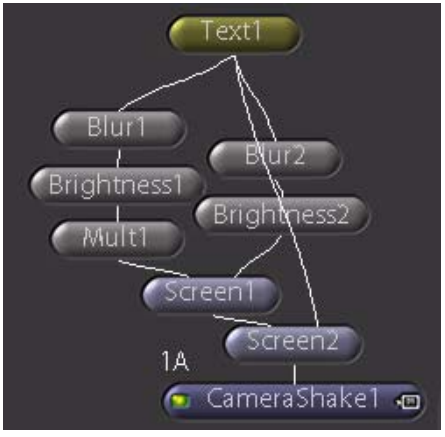




Text Treatments

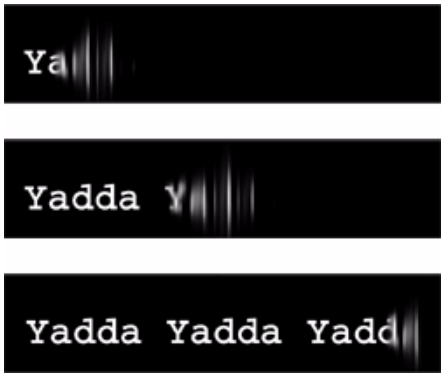
The following series of scripts plays with text treatments, and is stored as *doc/html/cook/scripts/car_ad_text.shk*, numbers 1 through 8.

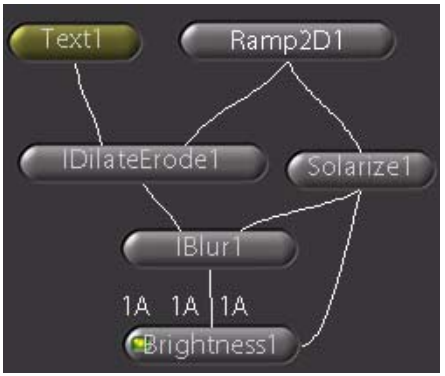
Script 1



- *Blur1*: Only yPixels is set.
- *Mult1*: A blue color is applied.
- *Blur2*: A small x/yBlur to help the text glow white.
- As with every script here, Transform–*CameraShake* is your friend.

Script 2

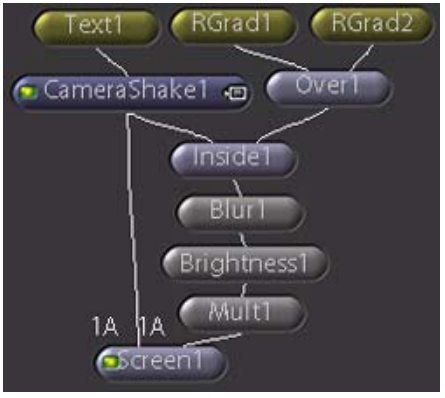




This script depends on *Ramp2D*, a macro stored in *doc/html/cook/macros*. The script will not load without this macro loaded into your *$HOME/nreal/include/startup* directory. It creates an animated mask traveling across the text, driving both a *IDilateErode* and an *IBlur* node. The *IDilateErode* has a value of -4 for X and Y. The *Solarize* is used to boost up the middle of the *Ramp2D* and to drop the outer ends to black. Like script 1, you only set the yPixel blur in *IBlur1*.

Here is the *Ramp2D*:

Script 3

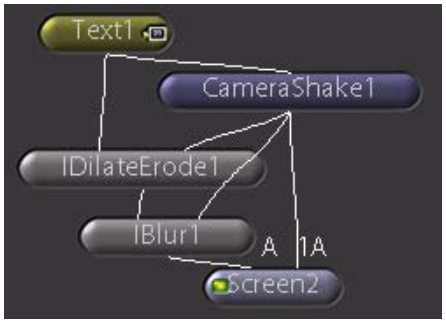


This uses the *noise*() function to randomize the xCenter of the *RGrads*. The text is then held *Inside* of these two animated shapes, and a process similar to Script 1 is applied:

- *RGrad1* xCenter expression: `noise(time/4)*Text1.width`
- *RGrad2* xCenter expression: `noise(time/4+100)*Text1.width`

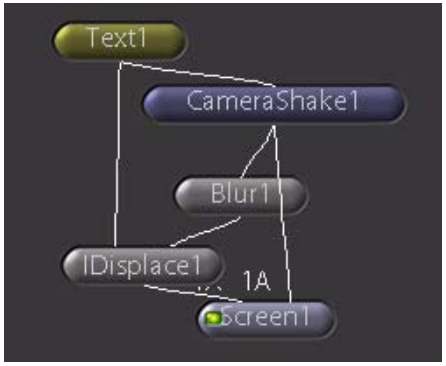
By adding 100 to the *noise* function's seed, you do not have an overlap in the animation. You can also change *time/4* to increase or decrease the frequency, that is, *time/10* or *time/2*. See "Expressions" on page 939.

Script 4



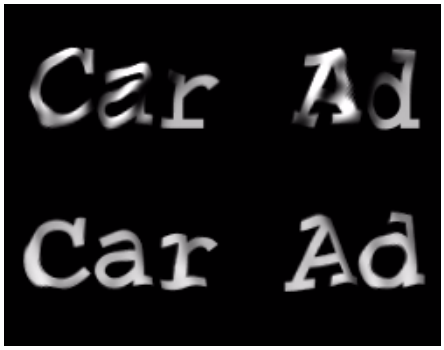
This script uses a heavily motion-blurred *CameraShake* (frequency is set to 6, amplitude is set to 50 and 100 for x and y). This then drives an expanding *IDilateErode* and *IBlur* to create an interesting interaction with the text.

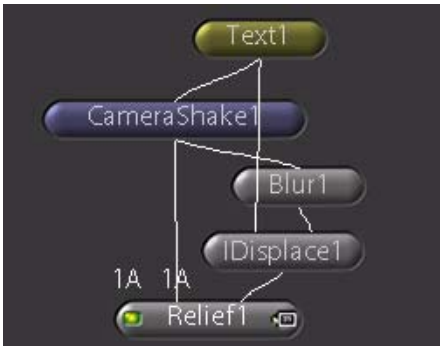
Script 5



The same *CameraShake* from Script 4 is used to feed an *IDisplace*. The *Blur* helps soften the warping image.

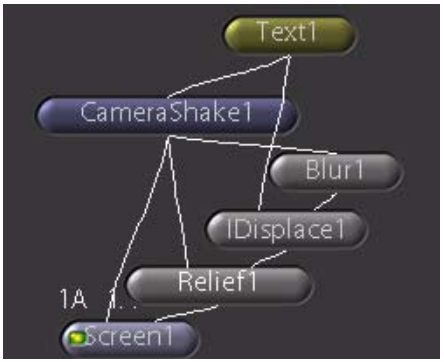
Script 6





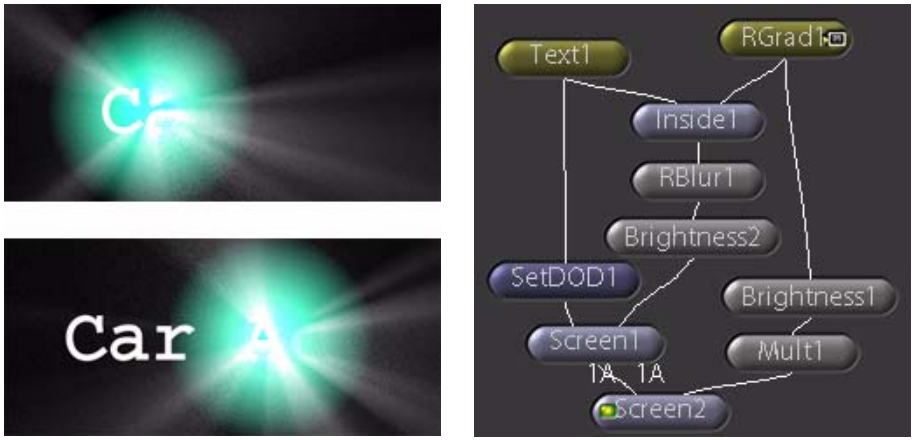
This is the same as Script 5, except the *Screen* is replaced with the *Relief* macro, found in *doc/html/cook/macros*. The script does not open without the *Relief* macro. The affect parameter of *Relief* is set to image 2.

Script 7



This is the same as Script 6, but the result is fed back over the motion-blurred text.

Script 8



This script is driven off of the position of the *RGrad*. The center of the *RBlur* is set to the center of the *RGrad*, and the right parameter of *SetDOD1* is set to *RGrad1.xCenter+10*, which unmasks the text as the *RGrad* moves to the right.

Installing and Using Cookbook Macros

All Shake macros can be found installed on your hard drive in the Shake directory, inside the *doc/html/cook/macros* directory. These macros can also be found in the *Documention/Cookbook Extras/macros* folder of the installation disc. Each macro consists of a .h file containing the actual macro, an optional user interface (UI) file, and an even more optional icon (.nri) file. To be installed, each of these files must be copied to the appropriate location on your hard drive for Shake to see them.

Note: Not every macro will have .h, UI, and .nri files. Some macros only have a .h file, while most usually have both .h and UI files.

To install a macro:

1. If there isn't already a *$HOME/nreal/include/startup* directory on your hard drive, create this directory structure.

   If necessary, create additional *$HOME/nreal/include/startup/ui* and *$HOME/nreal/icon* directories as well.

2. Place the .h file of the macro you want to install into the *$HOME/nreal/include/startup* directory.
3. Place the files containing a UI into the *$HOME/nreal/include/startup/ui* directory.
4. If the macro has one or more icon files, (which end with .nri) place them in the *$HOME/nreal/icon* directory.

Command-Line Macros

The following macros are designed to make some quick fixes. They are available for use from the command line.

FrameFill Macro

This is used in an emergency to fill in a missing frame when you have to go to film in, say, two minutes. It takes the frames next to the missing frame and averages them together.

For example:

```
shake -framefill bus2.#.jpg -t 41, 45
```

UnPin Macro

This is used to extract a texture map from an image. List out the four corner points (lower left first, counterclockwise). You can also set the antialiasing. This is a macro of the *CornerPin* node with inverseTransform activated. You usually use two Terminals to do this, one to test your coordinates, the second to test the command. You can get the coordinates by scrubbing the image—the coordinates appear in the title bar.

For example:

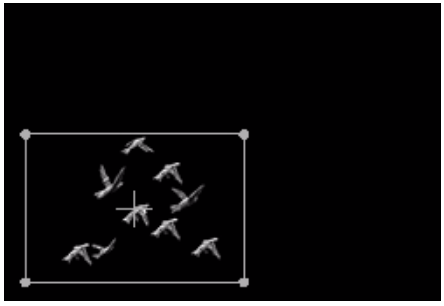
```
shake bus2.0040.jpg -unpin 91 170 417 154 418 2 42 94 274
```

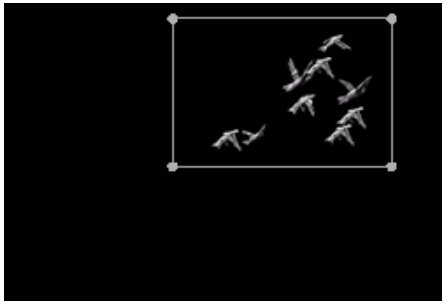
Image Macros

The following macros add Shake-generated image nodes to the Image tab.

Flock Macro

The following bird clip can be found in *doc/html/cook/macros/Flock/bird*. This takes a cycling clip and propagates copies of it offset in time, position, and scaling. There is a clip of birds provided as an example that you can use, royalty free.





You can use the box to roughly position the birds. The birds are also positioned with two extra movements, a gentle cosine function and some random noise. The freqX and freqY parameters control the frequency of the cosine movement (an up and down wave). The vibrationX and vibrationY parameters, along with the vibFreqX and vibFreqY parameters, control the random movement.

Manga Macro

This is an example of using NGLRender.

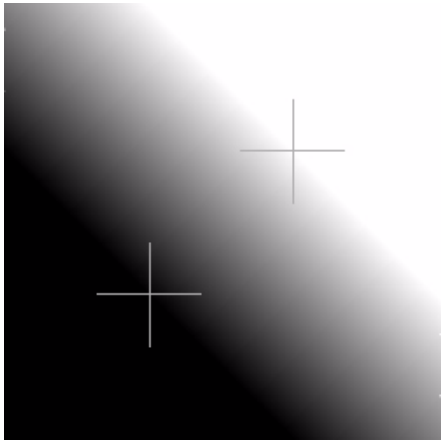
The second image has a *Twirl* applied with a very low antiAliasing value. Many kudos to Tom Tsuchiya of VPJ in Tokyo for information on Concentration Lines ("Syuuchyuu-sen").

Rain Macro

This macro can be used to generate rain to throw into a background. The rain is divided into three sheets, fg, mid, and bg. The lighting controls affect the height of the sheets. By using a low value, you can get a fake feeling of depth. Sort of.

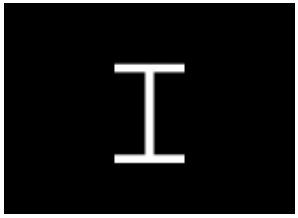
Ramp2D Macro

This uses the NGLRender drawing routines to draw a polygon on the screen to give you a ramp between any two points. The wedgeSize parameter should be brought down if you start to see artifacts along the edges.

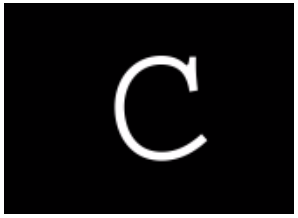


RandomLetter Macro

This generates a random letter up until the staticFrame number, at which point it becomes a letter of your choosing.



Frame 5



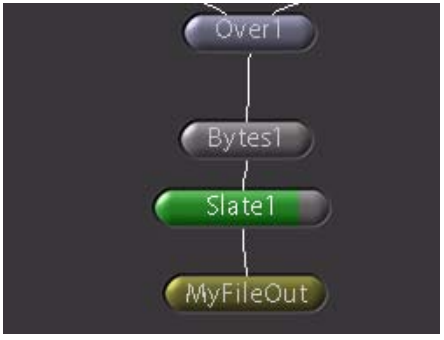
Frame 6

Slate Macro

This generates a slate giving information on the show, animator, frame range, and so on. Although you can use it to generate a frame, typically you attach it to the end of your script before the *FileOut*. If you create the *FileOut* first, you can link the *Slate* to print the *FileOut* file. You must precede it with a *:* (colon), for example:

```
:MyFileOut.imageName
```

The slate appears up to the markFrame parameter, which is 0 by default. It also loads the script name and the frame range into the slate, adds the current date, and gives you the option to stamp the current frame onto the output images.

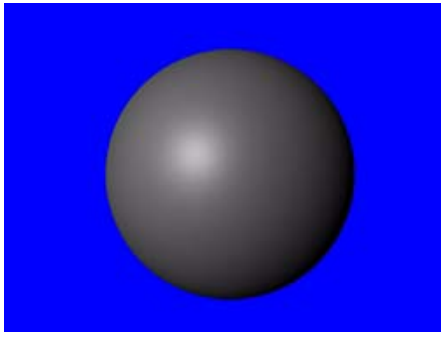


Color Macros

The following macros give you additional ways to manipulate color in your scripts.

AEPreMult Macro

This macro is intended to be used on an image that has a solid non-black background color which is still considered "premultiplied." By applying this function, you turn the background color to black. This may not work with all images.



ColorGrade Macro

This macro allows you to pick three color levels from a source image (typically shadows, midtones, and highlights) and match them to a target image. This is similar to *ColorMatch*, except it does not have the special math to protect it from solarizing. As a consequence, it is more accurate.

In this example by Richard Liukis for his short film, *Taste It All*, the plates were scanned with an unfortunately strong green cast. The shots were telecined down to video, color corrected with DaVinci, and edited. This same color correction needed to be applied to the film plates, so the video version and the logarithmic plates were both read in to Shake. A Color–*LogLin* was applied to the 2K plates, followed by a *ColorGrade*. You can see the green is even more pronounced in the default *LogLin*. The *ColorGrade* was used to match to the telecined version and then a second *LogLin* was applied to return it to log space.

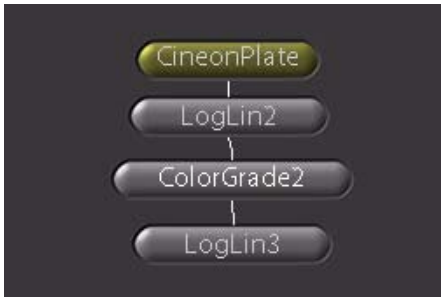
Input log 2K plate

Linearized 2K plate

Telecined reference footage

Color-graded plate

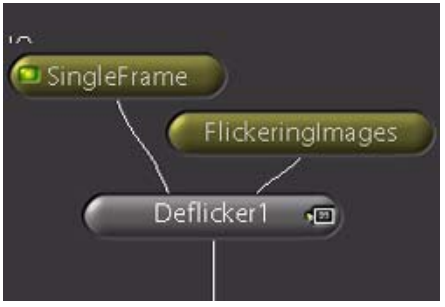
This example has a slightly high saturation, a slight blue cast, and punchier whites (but then again, 30 seconds were spent on it). Note because the tree is made of three concatenating color corrections, it was not necessary to convert up to float bit depth before the *LogLin*.



Deflicker Macro

This macro is helpful for reducing the flicker on an image. The macro takes two inputs: The first input is your reference frame, which should be a single frame from the clip you want to affect; the second input is the sequence you want to remove flickering from (a bluescreen image in this example). To use the macro, place the crop box on an area that is representative of an area that does not change its content, that is, over a portion of sky, rather than on the moving traffic below.

The first frame is usually a still reference frame. The second input is the flickering sequence. Position the box while looking at Input1 (*SingleFrame* in this example).



Temp Macro

This macro slides the midtones to warmer or cooler colors. Color-temperature-cool, not Fonzie-cool.

Original

Warmer

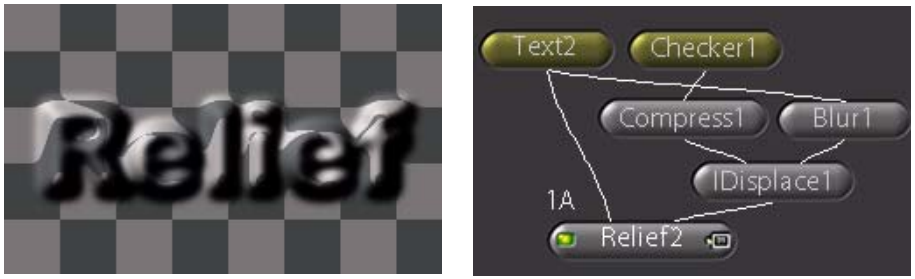
Cooler

Relief Macro

In the following, Example 1 has affect set to image1; the other two are set to image2.

Example 1: affect = image1

Example 2: affect = image2



Example 3: Relief + IDisplace

Key Macros

These macros help you further manipulate keyed images.

AlphaClamp Macro

This macro clamps the alpha channel to 0 and 1. This is helpful for float images and when pulling *LumaKeys*, as you can arrive at negative values that can have unwanted effects in your RGB channels. This macro is unnecessary for 8- or 16-bit images.

DeSpill Macro

The *HueCurves* node has a problem with processing float values. This macro mimics *Keylight*'s spill suppression, allowing you to pick a spill color.

KeyChew Macro

This is intended to give a more natural chewing or expansion of the matte edge than the result from *DilateErode*. This macro works only on the alpha channel. It also eliminates any mid-range alpha areas (reflections and shadows).

Note: This clamps your alpha channel to between 0 and 1, so be careful with float images. It maintains your float RGB channels properly.

Chewing in, DilateErode

Chewing in, KeyChew

Expanding out, DilateErode

Expanding out, KeyChew

Transform Macros

The following macros give you additional options when transforming images.

AutoFit Macro

This macro resizes an element if you only know one dimension of the output and you have to figure out the second one. You can use it in the command line. The second parameter determines what the first parameter specifies. You can provide either *w, h,* or 0 (width) or 1 (height):

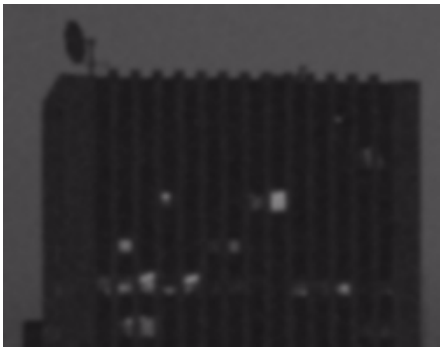
```
shake woman.sgi -autofit 2048 w
```

This resizes the *woman.sgi* image from the *Tutorial_Media* directory to 2048 x 1383, maintaining its aspect ratio.

PreTrack Macro

This macro helps when tracking noisy film plates. It drops the blue channel out, which tends to have the thickest grain, and also applies a slight blur to the footage. Once the track has been done, disable or remove the *PreTrack* node.

Note: The *Tracker* nodes have the ability to add blur while tracking.



RotateFit Macro

This macro resizes the frame to fit the new boundaries of the image. This version uses math to figure out the boundary, but you can also put the variables *dod[0] - dod[3]* into a *Crop* node.

Warping With the SpeedBump Macro

This macro creates a nifty bump with a shadow on your title.

Utility Macros

The following macros let you address Maya's Z-depth output, float elements used with the screen node, and let you manipulate the DOD.

MayaZ Depth Macro

The MayaZ macro converts the -1/z values that Maya outputs for the Z channel.

ScreenFloat Macro

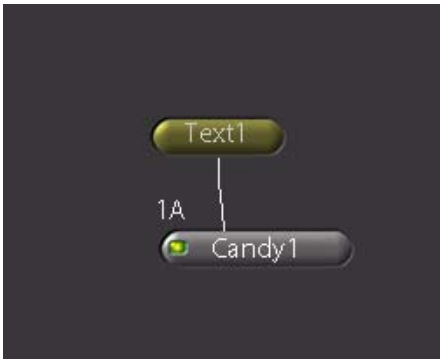
This macro lets you use the Screen function with float elements. You have to set the high value. If you use the normal float, you get negative artifacts, both aesthetically and numerically.

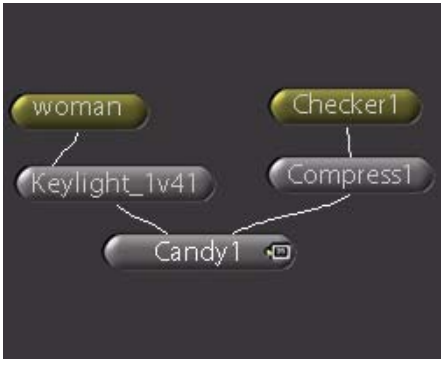
CopyDOD Macro

This macro copies, adds, or intersects the DOD between two images, and lets you set the background color.

Candy Macro

With this macro, the drop shadow appears only on the background image's alpha plane. It just seemed like a good idea at the time. If you do not prefer that, disable shadow generation with the useShadow parameter and use the normal *AddShadow* or *DropShadow* node.





MakeNodeIcon Macro

This macro is used to make the icons for the function tabs. Typically, you insert an image that is 700 x 300 pixels and the macro fits everything inside. Have fun.

AltIcon Macro

This macro is used to make the alternative icons. You insert a roughly 250 x 250 image into the macro. It requires the Relief macro to be installed. It also calls on the font Arial. If you do not have this font, search for the word in the macro file and substitute an appropriate font.

VLUTButton Macro

This is the macro used to make the VLUT buttons. It requires the *ViewerButton.h, WallPaper.h,* and *RoundButton.h* files to be installed. None of these need the UI files, just the .h startup files. Additionally, *Roundbutton.h* makes a call to the *round_button.iff* file, also included in the *doc/html/cook/macros/VLUTbutton* directory. Place the image somewhere and point the macro file to that new location inside of the Roundbutton macro.

Finally, it calls on the Arial font. If you do not have this font, search for the word and substitute an appropriate font.

The default arguments are:

```
text = “vlut {time}”
```

and

```
focus = 0 (off)
```

RadioButton Macro

This macro is used to create those swell radio buttons. You typically use this only in command-line mode, as it does some automatic output file naming for you. The first step is to create and specify a directory where you want to place the icons. This is typically *$HOME/nreal/icons/ux/radio*, as they tend to pile up, but you can place them anywhere.

Open the *radiobutton.h* file and look for the filePath declaration, line 6 below:

```
image RadioButton(
  const char *text="linear",
  // lengths are 74, 37, 53
  int length = 74,
  string fileName=text,
  string filePath="$HOME/nreal/icons/ux/radio/",
  int branch=time,
...
```

Make sure that directory exists. The standard image lengths are 37, 53, and 74 pixels long. The shortest you can practically do is 19.

Finally, it calls on the font Arial. If you do not have this font, search for the word and substitute an appropriate font.

The parameters are:

- *"text":* You can type it, or if you want more than one word, enclose it in "quotation marks."
- *length:* Output pixel width. Standard lengths are 37, 53, and 74 pixels.
- fileName: "text" by default. The macro appends *.on.nri*, *on.focus.nri*, *.off.nri,* and *.off.focus.nri* for frames 1, 2, 3, and 4.
- *filePath:* Redirect the output directory without editing the file.
- xScale: Scale the text on the X axis if you have to squeeze the letters a bit. The default is 1.
- *zoom:* Creates a button 19 pixels high by default, but you can scale it up. The default value is .25.
- *branch:* Specifies the state you want the macro. Normally you do not have to touch this, you just change the frame number, to which this parameter is set.
  - 1 = on
  - 2 = on + focus
  - 3 = off
  - 4 = off + focus

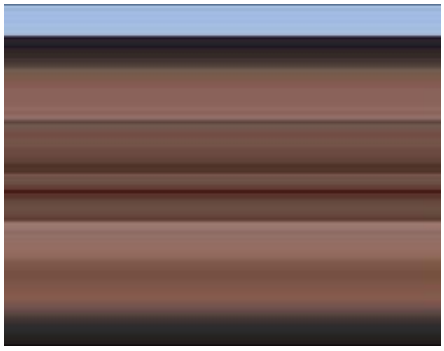
So, an example:

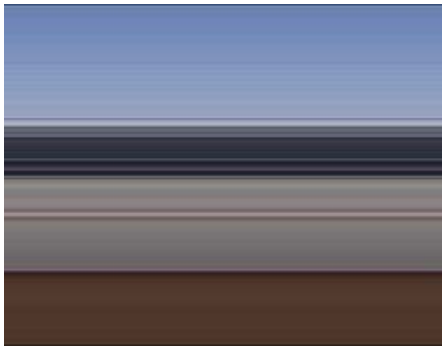
```
shake -radiob "Not A Dufus" 53 NotADufus -t 1-4 -v
```

This creates four files, *NotADufus.on.nri, NotADufus.on.focus.nri, NotADufus.off.nri,* and *NotADufus.off.focus.nri*, that are all 53 pixels wide and say "Not A Dufus" with different states of being illuminated and focused.

Wallpaper Macro

This helps with button icons, but it is also an interesting way to quickly generate animated backgrounds. It takes one vertical line from the input image and copies it across the image. By animating the line, you get a continuous-noise generation (of sorts).





Wedge Macro

This command helps pull an exposure wedge on logarithmic files. You pick an initial exposure setting, and then how far you want the wedging bracket to step (22 points, 45 points, 90 points, and so on). It then goes through 48 steps of color, brightness, and contrast. You can do this on the command line:

```
shake mycineonframe.cin -wedge -t 1-48 -fo mywedge.#.cin
```

or

```
shake mycineonframe.cin -wedge -10 15 -9 -t 1-48 -fo mywedge.#.cin
```

Using Environment Variables for Projects

You can set up projects using environment variables to better manage your different shots. Environment variables point Shake to the right directories without too much difficulty even if you move your project to different drives or machines. What is an environment variable? It is a word that is known by the computer to have a certain value. For example, the environment variable $HOME (environment variables are recognized by the $ in front of the word) is */Users/mylogin* on Mac OS X. For Linux, it is typically */usr/people/mylogin*. In both Mac OS X and Linux, you have variables such as $TEMP and $TMP to point to directories where temporary files are stored. Software can simply be coded to dump temporary data into $TEMP, rather than have to find a specific directory.

You can use these in Shake by setting a variable for your project and its directory location. For example, you have a project on *//MyMachine/BigDrive/MonsterFilm/Shot8*. If you set a variable, for example, *$myproj*, to point to that directory, Shake can always open to that directory. If you later move the project to *//MyBackUpMachine/OtherBigDrive/MonsterFilm/Shot8*, you do not have to go into each Shake script and change your *FileIn/Out* paths—just change the environment variable before you run Shake.

To set an environment variable in the Terminal, open either your *.tcshrc*, your *.aliases*, or another file that is read when you open a shell. Enter a line similar to the following:

```
setenv myproj /Documents/shot1
```

Once you have saved the file, type:

```
source .tcshrc
```

You have now set your environment variable. All shell windows that you have already created, and any open applications must be relaunched to read the new variable. For variables you have set using the *environment.plist* file, you have to log out and log in again.

To Test Your Environment Variable

There is a simple way to test if your environment variable exists. In a Terminal, type *echo $myproj*

and the proper value should be returned.

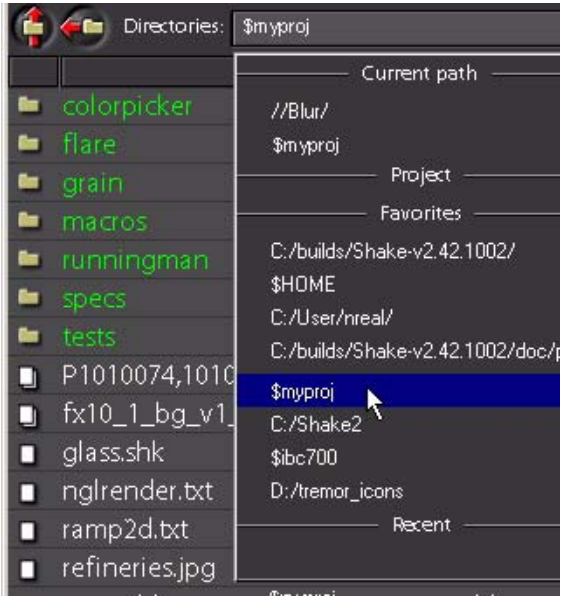
For the Shake portion:

1 Go to your *$HOME/nreal/include/startup/ui* directory and create a text file called *paths.h*. Enter the following lines. Note the / at the end of the second and third entries:

```
nuiFileBrowserAddFavorite("$myproj");
gui.fileBrowser.lastImageDir= "$myproj/" ;
gui.fileBrowser.lastScriptDir= "$myproj/" ;
```

2 Create a directory that *$myproj* points to, that is, if you set it to */Documents/shot1*, then create */Documents/shot1*.

3 When you relaunch Shake, it should launch to *$myproj*. In the Directories pull-down menu, you also see *$myproj* listed.



4 When you read an image from this location, Shake keeps the environment variable (*$myproj*) in the path. Therefore, whenever you move the entire project, reset the environment variable to point to the new path.

5 If you batch-render your project, the background machines must also understand the environment variable.

You can take further advantage of environment variables and projects by adding your own startup directory into the project area. Specify a shot-specific cache directory, assuming you have disk space to burn. This is only useful if you are working on several shots at once, as it keeps cached files around on a per-shot basis. Be extremely careful with this, however, because you can end up with gigs of data that you do not need if you do not clean up after yourself. Kind of like real life.

To set per-project settings for Shake:

1. As an example, in your project directory, create *startup/ui* directories:

   /usr/shot1/startup/ui

2. In your startup directory, place a file to relocate your cache. Create a text file called *cache.h* and add these lines, obviously changing *MyShotName*:

   ```
   diskCache.cacheLocation="/var/tmp/Shake/MyShotName";
   diskCache.cacheSize = 500;
   ```

   The second line indicates the size in MB of your cache. Set it to something appropriate according to how much disk space you have to spare. Shake automatically creates that directory the first time it needs it. Remember to remove this cache directory when cleaning up your project.

3. To have per-shot macros or settings like Formats, add them into the startup and ui directories.

4. Return to set another environment variable, following the steps outlined above. You want to set *NR_INCLUDE_PATH* to your project directory:

   ```
   setenv NR_INCLUDE_PATH $myproj
   ```

Keyboard Shortcuts and Hot Keys

A Appendix

Keyboard Shortcuts in Shake

In some instances, the keyboard shortcuts vary on different platforms. In the following tables, the Mac OS X commands appear first, followed by the equivalent Linux commands.

Note: In many instances, both platform options work on Mac OS X.

General Application Commands

The following keyboard shortcuts are for general project management.

Command	Description
Command-N or Control-N	Create New script.
Command-O or Control-O	Open script.
Command-S or Control-S	Save script.
Shift-Command-S or Shift-Control-S	Save script as.
Shift-Command-O or Shift-Control-O	Recover script.
Command-Z or Control-Z	Undo, by default up to 100 steps.
Command-Y or Control-Y	Redo your steps unless you have changed values after several undos.
Command-F or Control-F	Find nodes. Opens the Select Nodes by Name window, which lets you select nodes that match criteria in a search string.
Command-X or Control-X	Cut.
Command-C or Control-C	Copy.
Command-V or Control-V	Paste.

Navigating in Time

The following keyboard shortcuts let you move the playhead backward and forward in time while you work.

Command	Description
Forward Arrow	Move forward one frame.
Back Arrow	Move back one frame.
.	Begin forward playback.
Up Arrow	Jump to next keyframe.
Down Arrow	Jump to previous keyframe.
Home	Fit the current time range into the Time Bar.
Shift-.	Begin cached playback.
T	Toggle the Time Bar between timecode and frame display.

General Windowing Keyboard Shortcuts and Modifiers

The following keyboard shortcuts and modifiers are available for any region in Shake.

Command	Description
Option-click and drag, or middle mouse button and drag	Pan within that region of the window.
Space bar	Expand interface region to occupy the full Shake window area, and collapse again.
Command-Option-click or Control-Alt-click and drag, or Control-middle mouse button and drag	Zoom into or out of the Curve Editor, Node View, and Time Bar.
Esc	Stop processing.
U	Update the Viewer.
Shift-middle click and drag a tab, or Shift-Option-click or Shift-Alt-click and drag a tab	Tear off a tab to use as a floating window.
T	Toggle between Time Code and Frame View in the Curve Editor, Time View, and Time Bar.

Saving and Restoring Favorite Views

The following keyboard shortcuts let you define and restore favorite views in the Viewer, Node View, Curve Editor, Time View, or Parameters tab.

Command	Description
Shift-F1-5	Save favorite view of any interface area (Viewer, Node View, Curve Editor, Time View, or Parameters tab) to be accessable via the F1, F2, F3, F4, or F5 key.
F1-5	Restore favorite view framing. Results in that area being panned and zoomed to the saved position.
Option-F1-5	Restore favorite view framing and state. Results in the restoration of additional state information, depending on the region of the interface. This can include the node being viewed, the currently loaded curves in the Curve Editor, or the parameters being tweaked in the Parameters tab.

The Viewer

The following keyboard shortcuts help you use the Viewer.

Command	Description
N	Create/Copy New Viewer.
F	Fit frame to image.
Shift-F	Fit frame to window.
Control-F	Fit Viewer to image.
Option-drag or Alt-drag	Pan image.
+ or –	Zoom image in Viewer.
Home	Reset view.
R, G, B, A, C	Toggle red, green, blue, alpha, and color channel views.
1 (top number row)	Toggle between the A/B image buffers.
2 (top number row)	Toggle color channels. Shift-2 toggles in reverse.
3 (top number row)	Toggle update mode. Shift-3 toggles in reverse.
4 (top number row)	Toggle among Viewer lookup tables (VLUTs). Shift-4 toggles in reverse.
5 (top number row)	Toggle DOD mode.
6 (top number row)	Toggle through Viewer scripts.
7 (top number row)	Toggle compare mode.
8 (top number row)	Toggle onscreen transform controls.

Flipbook Keyboard Shortcuts

The following keyboard shortcuts are available for any open Flipbook.

Command	Description
. (period; consider it as >)	Play forward.
, (comma; consider it as <)	Play backward.
Shift->	Ping-pong playback.
Shift-drag	Shuttle playback.
Option-drag or Alt-drag	Reveal compare buffer, if set.
Control->	Play through once.
Space bar	Stop playing and rendering.
?	Continue rendering.
R, G, B, A, C	View a specific channel.
L	Toggle numeric representation of channel values.
Home	Reset zoom to 1.
– or = (near Delete key or Backspace key)	Zoom in and out.
+ or – (on numeric keypad)	Increase playback rate.
T	Toggle real-time playback.
D	Double/single buffer toggle, SGI only.
Left Arrow or Right Arrow keys	Advance playhead forward or backward through frames.
H	If in compare mode, set to horizontal split.
V	If in compare mode, set to vertical split.
S	If in compare mode, switch split.
Esc	Close window.

Tool Tab Keyboard Modifiers

The following keyboard modifiers help you create new nodes in the Node View in different ways.

Command	Description
Click node in Tool tab	Insert node into tree.
Shift-click node in Tool tab	Create new branch in tree.
Control-click node in Tool tab	Replace currently selected node in tree.
Shift-Control-click in Tool tab	Create a new node tree.
Option-drag or Alt-drag	Pan window.

Node View

The following keyboard shortcuts and modifiers help you work within the Node View.

Command	Description
Shift	When you move a node and the grid is enabled, holding down Shift allows the node to move freely. When the grid is disabled, holding down Shift locks it to the grid. Grid width and height parameters are in the guiControls subtree of the Globals tab.
Del (near Home/End keys) or Delete	Delete the selected nodes. If the branching is not complicated, the noodles between the parent(s) and children automatically reattach to each other.
+	Zoom into the Node View (also use Command-Option-click or Control-Alt-click and drag, or Control-middle click and drag).
–	Zoom out of the Node View (also use Command-Option-click or Control-Alt-click and drag, or Control-middle mouse button and drag).
Home	Center all nodes.
O	Turn on the Overview window to help navigate in the Node View.
Command-E or Control-E	Toggle enhanced Node View on and off.
F	Frame all selected nodes into the Node View.
Shift-F	Frame all selected nodes into the Node View, but maintain zoom level.
Command-F or Control-F	Activate nodes according to what you enter in the Search string field in the Select Nodes by Name window. • *Select by name*: Enter the search string, and matching nodes are immediately activated. For example, if you enter just "f," *FileIn1* and *Fade* are selected. If you enter "fi," just the *FileIn1* is selected. • *Select by type*: Selects by node type. For example, enter "Transform," and all *Move2D* and *Move3D* nodes are selected. • *Select by expression*: Allows you to enter an expression. For example, to find all nodes with an angle parameter greater than 180, enter "angle >180." *Match case*: Sets case sensitivity.
L	Perform an automated layout on the selected nodes.
Shift-L	Stack nodes closely to each other vertically.
X	Snap all selected nodes into the same column.
Y	Snap all selected nodes into the same row.
I	Turn off selected nodes when activated. Select the nodes again and press I to reactivate them. You can also load the parameters into the Parameter View and enable ignoreNode.
E	Pull the active nodes from the tree, and reconnect the remaining nodes to each other.
R	Activate/refresh the thumbnails for selected nodes.

Command	Description
T	Turn on/off selected node thumbnails. If you haven't yet created a thumbnail (R), this does nothing.
C	Display the RGB channels for thumbnails.
A	Display the alpha channel of thumbnails.

Selecting Nodes

The following keyboard shortcuts let you select different ranges of nodes in the Node View.

Command	Description
Command-A or Control-A	Select all nodes.
Shift-A	Select all nodes attached to the current group.
!	All selected nodes are deactivated, all deactivated nodes are activated.
Shift-U	Add all nodes upstream from the currently active nodes to the active group.
Shift-D	Add all nodes downstream from the currently active nodes to the active group.
Shift-Up Arrow	Add one upstream node to the current selection.
Shift-Down Arrow	Add one downstream node to the current selection.

Grouping Nodes

The following keyboard shortcuts let you create and manage groups in the Node View.

Command	Description
G	Visually collapse selected nodes into one node. When saved out again, they are remembered as several nodes. To ungroup the nodes, press G again.
Shift-G	Temporarily activate Grid when moving nodes, or consolidate two or more groups into a larger group.
Command-G or Control-G	Ungroup nodes.

Macro Shortcuts in the Node View

The following shortcuts let you create, open, and close macros in the Node View.

Command	Description
Shift-M	Launch the MacroMaker with the selected nodes as the macro body.

Command	Description
B	Open a macro into a subwindow so you can review wiring and parameters. You cannot change the nodes inside of the subwindow.
Option-B or Alt-B	Close the macro subwindow when the pointer is placed outside of the open macro.

QuickPaint

The following keyboard shortcuts are available in the *QuickPaint* node.

Key	Function
F9	Use last brush.
F10 or P	Pick color.
F11	Toggle between hard/soft brush.
Z	Magnet drag in Edit mode.

Note: In Mac OS X, Exposé is mapped to F9-F12 by default. To use these keys in Shake, disable the Exposé keyboard shortcuts in System Preferences.

The Curve Editor

The following keyboard shortcuts and modifiers help you make adjustments in the Curve Editor.

Command	Description
Option-drag or Alt-drag	Pan window.
Command-Option-drag or Control-Alt-drag	Zoom window.
+ or – (near Delete or Backspace key)	Zooms in and out.
Option-drag or Alt-drag numbered axis	Pan only in that direction.
Drag on numbered axis	Scale that direction.
S	Sync time to current keyframe.
T	Toggle time code/frame display.
Shift-drag	Select keyframes.
Command-drag or Control-drag	Deselect keyframes.
Command-A or Control-A	Select all curves.
Shift-A	Select all control points on active curves.
B	When drag-selecting keys, this keeps the Manipulator Box active to enable transform controls.
Q	Allows you to pan the selected points.

Command	Description
W	Scales the selected points, with the initial click point the center of scaling.
E	A non-linear scaling of the points.
K	Insert a key at the frame the pointer is at over the selected spline.
V	Toggle visibility of selected curves.
X, Y	Allow movement only on X or Y axis.
H	Flatten tangents horizontally.
Command-click or Control-click tangent	Break tangent.
Shift-click tangent	Rejoin broken tangents.
Del (near Home/End keys) or Delete	Delete active keyframes.
Delete or Backspace	Remove curves from Editor (does not delete the curves).
F, Control-F	Frame selected curves.
Shift-F	Frame selected control points.
Home	Frame all curves.

Parameters Tab Shortcuts and Modifiers

The following keyboard shortcuts help you to make adjustments in the Parameters tabs.

Command	Description
Control-drag	Activate virtual sliders.
Tab	Advance to next value field.
Shift-Tab	Go to previous value field.
Shift-Left/Right Arrow	Increase value field value by 10 x normal increment.
Control-Left/Right Arrow	Increase value field value by normal increment.
Option or Alt-Left/Right Arrow	Increase value field value by 1/10 x normal increment.
Right-click	Access pop-up menus.
Right-click Color Picker	Pop up color palette.
Drag parameter name	Copy parameter to target parameter.
Shift-drag parameter name	Link parameter from target parameter.

MultiPlane Node Keyboard Shortcuts

The following keyboard shortcuts let you choose angles from the multi-pane Viewer interface presented by the MultiPlane node. To switch a pane's angle, position the pointer within that pane, and press one of these keys on the numeric keypad.

Numeric Keypad	Description
0 (numeric keypad)	Cycle through every angle in the Viewer pane with the pointer positioned within it.
1 (numeric keypad)	Display the currently selected renderCamera angle in the Viewer pane with the pointer positioned within it.
2 (numeric keypad)	Display the Front angle in the Viewer pane with the pointer positioned within it.
3 (numeric keypad)	Display the Top angle in the Viewer pane with the pointer positioned within it.
4 (numeric keypad)	Display the Side angle in the Viewer pane with the pointer positioned within it.
5 (numeric keypad)	Display the Perspective angle in the Viewer pane with the pointer positioned within it.

Keyboard Modifiers for Transform and MultiPlane Nodes

The following keyboard modifiers let you make adjustments to images in transform nodes, and to layers and cameras in the *MultiPlane* node.

Keyboard	Description
Control-drag	Drag the center point of selected layer to a new location.
Press Q or P and drag	Pan selected layer around global axis.
Press W or O and drag	Rotate selected layer.
Press E or I and drag	Scale selected layer.
V-drag	Rotate the camera about its Z axis.
S-drag	Rotats the camera about the X and Y axes, about the camera's own center point, changing the position of the camera target.
Z-drag	Pan the camera in and out along the Z axis.
D-drag	Pan the camera and camera target together along the X and Y axes.
X-drag	Pivot the camera about the camera target's orbit point. Also pivot the Persp or Camera view in the Viewer.
T-drag	Move the camera and camera target together in any view.

Keyboard Modifiers for Color Adjustments

The following chart lists all the keyboard shortcuts for color adjustments within a color control. To use, press one of the listed channel keys, and drag within the color control to make the listed adjustment.

Keyboard	Channel	Description
R	Red	Adjust red channel independently.
G	Green	Adjust green channel independently
B	Blue	Adjust blue channel independently.
O	Offset	Boost or lower all color channels relative to one another.
H	Hue	Adjuss all channels by rotating around the ColorWheel.
S	Saturation	Adjuss color saturation, according to the HSV model.
V	Value	Adjust color "brightness," according to the HSV model.
T	Temperature	Adjust overall color between reds and blues.
C	Cyan	Adjust cyan according to the CYMK colorspace model.
M	Magenta	Adjust Magenta according to CMYK.
Y	Yellow	Adjust Yellow according to CMYK.
L	Luminance	Adjust black level, otherwise referred to as luminance.

The Shake Command-Line Manual

B Appendix

Shake started in its infancy as a command-line compositor—you can conceivably execute a 500-node script that is typed out in a Terminal. "Conceivably," but not practically, since nodes such as *Primatte*, *Stabilize*, *QuickPaint*, and *RotoShape* have unwieldy formats. And, of course, you would have to type out 500 nodes on the command line, which is impractical. However, using the Terminal remains an ideal method to execute many daily image-processing functions, such as:

- Image resizing
- Bit depth or channel reordering
- Standardized color correcting (log to lin conversion, gamma corrections, and so on)
- Converting file format
- Flipbook-rendering an image sequence
- Executing scripts
- Accessing image information
- Quick compositing of 3D-rendered elements over backgrounds

These functions can be rendered in the command line more quickly and efficiently than in the graphical interface (if you are comfortable with typing in the Terminal), since they involve relatively straightforward commands. The other major use of the command line is to execute scripts that you have created in the interface and then saved to disk.

This section discusses some general principles about the command-line shell, and then lists several examples. The last section is a list of frequently used functions. Since every node can potentially be used, this not a complete list.

Note: All example images are located in the *doc/pix* directory. All examples assume you are in this directory or below, as noted.

Viewing, Converting, and Writing Images

Shake does three basic things on the command line: It executes image operations, executes scripts, or views images. The following are some simple examples to start the discussion.

To display images:

- Type the name of the images, for example:

```
shake truck.iff bg.iff sign_mask.iff
```

Note: For instructions on how to use the Flipbook Viewer, see "Previewing Your Script Using the Flipbook" on page 90.

To convert or save a file:

- Append the *-fileout*, or *-fo*:

```
shake truck.iff -fo test.rla
```

The above command saves an image called *test.rla* in the .rla format. For a list of supported formats and their extensions, see "About Image Input" on page 107.

To compare two images:

- Load the first image and then the second after the *-compare* flag:

```
shake bg.iff -compare sign_mask.iff
```

On most operating systems, use Option-click or Alt-click to drag between the two images. Press H, V, and F for horizontal, vertical, and fading wipes. For Linux systems, you must Shift-Control-drag.

Time and Viewing Image Sequences

Shake assumes time is set to frame 1 unless you specify a time range. This is done with the *-t* option, and a frame numbering symbol, such as:

```
shake alien/alien.#.iff -t 1-50
```

The frame numbering is substituted with a symbol (# in the above example) which indicates that the file is padded to four places. The following table lists other symbols that can be used in frame numbering.

Shake Format	Reads/Writes
image.#.iff	image.0001.iff, image.0002.iff, etc.
image.%04d.iff	image.0001.iff, image.0002.iff, etc.
image.@.iff	image.1.iff, image.2.iff, etc.
image.%d.iff	image.1.iff, image.2.iff, etc.
image.@@@.iff, image.###.iff	image.001.iff, image.002.iff, etc.
image.%03d.iff	image.001.iff, image.002.iff, etc.
image.10-50#.iff	image.0010.iff at frame 1, 11 at frame 2, etc.

The *-t* option is extremely flexible. You can choose to render frame ranges, stepped ranges, individual frames, or any combination.

Time Range	Number of Frames	Frames Rendered
-t 1-100	100	1, 2, 3... 100
-t 1-100x2	50	1, 3, 5... 99
-t 1-100x20	5	1, 21, 41... 81
-t 1-20,30-40	31	1, 2, 3... 20, and 30, 31, 32... 40
-t 1-10x2,15,18,20-25	13	1, 3, 5... 9, 15, 18, 20, 21, 22... 25
-t 100-1	100	100, 99, 98... 2

To convert a sequence of images, use a line similar to the following:

```
shake alien/alien.#.iff -fo test.@.jpg -t 1-30 -v
```

The following table includes some clever ways to renumber image sequences:

Renumbering Clips	Description
shake bus2.40-79#.jpg -t 1-40 -fo toto.#.iff -v	Shifts frame numbering to start at 1.
shake bus2.#.jpg -t 40-79 -fo toto.#-39.iff -v	Also shifts frame numbering to start at 1.
shake bus2.#.jpg -t 40-79 -fo toto.80-#.iff -v	Reverses timing starting at frame 1.
shake bus2.#.jpg -t 40-79 -fo toto.118-#.iff -v	Reverses timing within the same frame range.
shake bus2.#.jpg -t 40-79 -fo toto.@.iff -v	Unpads the clip.
shake bus2.40-79x2#.jpg -t 1-20 -fo toto.#.iff -v	Halves the timing of the clip.
shake bus2.40-79x.5#.jpg -t 1-80 -fo toto.#.iff -v	Doubles the timing of the clip.

Appending Functions

Append optional functions with the - (dash) sign to preface each function call, followed by a space for any arguments it might take. Not all functions take arguments, but most do.

In the following, the *Blur* command has an argument of 50 (the amount of blur you want to apply):

```
shake truck.iff -blur 50
```

You can append as many functions as you want:

```
shake truck.iff -blur 50 -invert r -z 2
```

The following is a good example of a common command-line test of 3D-rendered imagery:

```
shake truck.iff -outside sign_mask.iff -over bg.iff
```

As long as there is no ambiguity, it is not necessary to type the entire function name. For example:

```
shake bg.iff -bri 2
```

calls the *Brightness* function since there are no functions starting with *bri*. However, calling:

```
shake bg.iff -con 2
```

informs you that it cannot choose between *Conform*, *Constraint*, *ContrastLum*, *ContrastRGB*, or *Convolve*. To solve the problem, replace your command with enough letters to end the ambiguity:

```
shake bg.iff -contrastl 2
```

This calls the *ContrastLum* function.

In regard to capitalization and function names, function names are always capitalized in the interface. In the command line, function names are always lowercase so you do not have to press the Shift key. The exception is when you call a function in quotes, such as the *Linear* function used to animate parameters:

```
shake truck.iff -blur "Linear(0,0@1,20@20)" -t 1-20
```

Note: For more information on animation curves, see "More About Splines" on page 316.

Most functions are image manipulators—they modify the image you load. Others are controls to modify how Shake executes the command. For example, *-brightness 2* brightens an image, but *-fps 24* loads a Flipbook preset to 24 frames per second. Although image functions always occur in a linear fashion—the order of commands matters—controls can be placed anywhere. For example, the following lines are identical:

```
shake alien/alien.#.iff -rotate 45 -t 1-50 -cpus 2
shake -cpus 2 -t 1-50 alien/alien.#.iff -rotate 45
```

However, the following two lines are different:

```
shake truck/truck.iff -pan 200 0 -rotate 135
shake truck/truck.iff -rotate 135 -pan 200 0
```

If you place a second control on a line, it replaces the previous setting. For example:

```
shake alien/alien.#.iff -t 1-50 -fps 24 -bri 2 -fps 30 -t 10-20
```

plays at 30 frames per second and renders only frames 10-20.

Getting Help

How do you know what *Blur* is expecting? Aside from using the product non-stop for five years, you can also type:

```
shake -help blur
```

to return what arguments that function takes. *Blur* has six arguments:

```
-blur [xPixels] [yPixels] [spread] [xFilter] [yFilter] [channels]
```

Often, you don't need to specify every argument. Shake gives you an error message if it expects more arguments than you have supplied. For additional information, refer to the relative function pages. For example, the *Blur* function can be found on page 864 in Chapter 28, "Filters."

You also do not have to know the entire function name. For example, your request for help on the *Blur* node probably produced information about seven different functions, so try:

```
shake -help mul
```

and you see that Shake simply looks for the string "mul." To quickly launch these docs:

```
shake -doc
```

Argument Flow

Looking at the arguments in the *Blur* function, it expects an image as its first input, which is labeled *In*:

```
image Blur(
  image In,
  float xPixels,
  float yPixels,
  int spread,
  const char * xFilter,
  const char * yFilter,
  const char * channels
);
```

However, the command line differs from the interface in that it always assumes the first image argument is coming from the previous argument in a linear flow, so omit the first image argument. Therefore, in *Blur*, the first argument on the command line is the xPixels parameter.

Shake assumes the previous image is fed into the function that follows. However, if you have multiple input images, only the last image has the effect applied.

In this example, only *sign_mask.iff* is blurred:

```
shake truck.iff bg.iff sign_mask.iff -blur 50
```

Occasionally, you want to perform different operations on two different images within the same command. This can be done with the branching commands *-fi* (*FileIn*) and *-label*. Label a branch with the *-label* function, and start a new branch with the *-fi* function. The following example blurs the background, and then assigns it a temporary label of BG. It then reads the truck, brightens it (the truck, but not the background), and composites the result over the blurred background:

```
shake bg.iff -blur 50 -label BG -fi truck.iff -bri 2 -over BG
```

Scripts

Execute pre-saved scripts with the *-exec* command. These scripts are usually generated from the interface. The *-exec* function scans the script and executes all *FileOut* nodes. If there are no *FileOut* nodes in the script, the function does nothing.

```
shake -exec my_script.shk -v
```

The *-v* stands for "verbose," and provides feedback on the progress of the render. You can also override many settings in the script. For example:

```
shake -exec my_script.shk -v -proxys .5 1 -t 20-30 -motion 1 1
```

renders at half-proxy scale, full-proxy ratio, only frames 20 to 30, and activates motion blur without having to edit the script itself.

You can save a script from the command line with the *-savescript* function:

```
shake truck/truck.iff -outside truck/sign_mask.iff -over truck/bg.iff
     -savescript toto.shk
```

If there is no *FileOut* in the script, you can test the script with the *-script* command. It is similar to the *-exec* function in that it executes *FileOut* nodes. However, it also views every branch in the script, and that can be awkward when rendering. It is really just for testing scripts:

```
shake -script toto.shk
```

Command-Line Controls

In the following tables, < > indicate a mandatory argument, and [ ] indicate an optional argument.

The following functions are unique to the command line.

Time Range Control	Description
-t <frameRange>	See "Time and Viewing Image Sequences" on page 1016.

I/O Functions	Description
-exec <script.shk>	Renders only *FileOut* nodes in a script. See "Time and Viewing Image Sequences" on page 1016.
-fi <image>	*FileIn*. Allows branching operations. See "Argument Flow" on page 1019.

I/O Functions	Description
-fo <image>	*FileOut*. Writes the image to disk in the format of the file extension. If no extension is given, it is saved in .iff format.
-savescript <script.shk>	Saves all of the previous commands into a script for later execution. Also helpful to get scripting formats.
-script <script.shk>	Reads a script and tests it. All branches are viewed; all *FileOut* nodes are rendered to disk.

Viewing Controls	Description
-compare <image> Example: *shake bg.iff -blur 20 -compare bg.iff*	Allows you to compare two images. Option-click or Alt-click (Shift-Control-click on Linux) to drag between the images. Press H or V to split the screen between horizontal and vertical splits.
-fps <framesPerSecond>	Frames per second for the playback. You can also press + or – on the number keypad to set this. The actual and target fps are displayed at the top of the Flipbook.
-gui	Launches your functions in the interface.
-monitor [keepFrames] [w] [h]	With no options, this only displays the current frame, discarding the frame when you go to the next frame. When *keepFrames* is set to 1, it keeps all frames in memory. When set to 0, it discards them. You can also specify the width and height of the monitor window.
-view [zoom]	Can be used to view intermediate stages of a string of nodes. You can also list an optional zoom level for the Viewer.

Information Controls	Description
-doc	Launches this fine and flawless documentation.
-help, -h [functionName]	Without any arguments, lists all functions available in Shake. When you supply a string, it matches any functions with that string in it. See "Getting Help" on page 1019.
-info	Lists size, type, bit depth, and channel information regarding your image.
-version	Prints the version of Shake.
-v, -vv	Verbose. When *-v* is on, the rendering time is printed for each frame. *-vv* displays a percentage of render completion as it renders.

Render Controls	Description
-cpus <procs>	The number of CPUs to use. Default is 1.
-createdirs	Used with *-renderproxies* to create the necessary subdirectories for the proxies. It does not affect *FileOut* renders.
-fldr <field>	Turns on field rendering. 1 = odd/PAL, 0 = even/NTSC.
-mem <Mb>	Sets the maximum amount of memory to use.

Render Controls	Description
-motion <quality> <override>	Sets the motion blur quality. In the command line, you must enable *override* (set it to 1). When executing a script, you either multiply what is saved as the global motion blur value (override = 0), or override it (override = 1).
-node <nodeName>	Only executes that node. For use with *-exec*.
-nocache	Disables the call to the cache, forcing complete recomputation of each element.
-noexec	Does nothing except compile the script to see if there are errors.
-pixelscale <scale> [ratio]	Sets the *pixelscale* and *ratio*. Not frequently used in the command line. See Chapter 4, "Using Proxies," on page 137.
-proxyscale <scale> [ratio]	Sets the proxyscale and ratio. Does a low-resolution test render of your commands. You can specify two numbers—the scale and ratio, or p1/p2/p3 which uses the preset proxies. See Chapter 4, "Using Proxies," on page 137.
-renderproxies [p1] [p2] [p3]	Renders *FileIn* nodes already saved into a script as the preset proxies. If used without arguments, renders what is saved in the script, otherwise you can specify which subproxies you want, p1, p2 , and/or p3. Usually used with *-createdirs*.
-sequential	Sequentially executes functions. For use with *-exec*. When you have multiple *FileOut* nodes, it may be more efficient to render them one at a time with this flag, rather than simultaneously.
-shutter <shutterTiming> [offset]	Sets motion blur shutter controls.

Quality Controls	Description
-fast [quality]	When no argument is given (just *-fast*), anti-aliasing is disabled to speed the render. *-fast 1* sets quality to 0 (low), while *-fast 0* sets quality to 1 (high).

Masking Controls	Description
-evenonly <image> Example: *shake bg.iff -rotate 45 -evenonly bg.iff*	Only executes the previous commands on the even fields of the image, and mixes it back in with the image you supply.
-isolate <image> <channel> Example: *shake bg.iff -blur 50 -isolate bg.iff a*	Only executes the previous commands on the channels you specify, and mixes it back in with the image you supply.
-mask <image> [channel] [percent] [invert] Example: *shake bg.iff -blur 50 -mask truck.iff*	Masks off the previous commands you specify, with the mask coming from the image. You can specify the channel and percentage, or invert the mask.

Masking Controls	Description
-oddonly <image>	Only executes the previous commands on the odd fields of the image, and mixes it back in with the image you supply.
-roi <left> <bottom> <w> <h>	Only executes a square portion of the image that you specify.

Labeling Controls	Description
-curve [type] <name> <defaultValue>	Creates a global variable available to the script that is loaded or other functions executed. See below for an example. You should be aware that defining variables from the command line using the -curve flag will produce a duplicate variable if that variable already exists within that script. Afterwards, this script may fail to load.
-label <name>	Allows you to temporarily label an image for later use in the command line.

Frequently Used Functions

Since you can use *any* of the functions in Shake, the following tables of frequently used functions are only partial lists. However, the tables represent the functions that are both practical to use (not too many parameters), and are useful in the command line, such as file resizing, basic channel manipulation, and quick generation of test elements. Typically, the functions have more options than are listed here. For more information on the functions, see the relative function sections in the appropriate chapter. For example, the *Ramp* image function is located in Chapter 21, "Paint."

Image Functions	Description
-addtext <text>	A quick way to generate text, as you only have to supply the text itself, without having to enter the width, height, and bit depth.
-black [w] [h] [bitDepth]	A quick way to generate black.
-color <w> <h> <bitDepth> <r> <g> <b> [a] [z]	A color field.
-ramp [w] [h] [bitDepth] [orientation]	A ramp. Use an orientation of 1 for vertical, 0 for horizontal.
-text <w> <h> <bitDepth> <text>	Text.

Color Functions	Description
-brightness <value>	Multiplies the RGB channels by value.
-contrastlum <contrast> [center]	Performs a contrast, with the pivot point at center.
-delogc	A log to linear color conversion to quickly see Cineon plates.
-gamma <value>	Gamma.
-logc	A linear to log color conversion to convert files into log color space.
-luminance	A quick function to generate a black-and-white image based on the luminance. Generates a 1-channel image.

Color Functions	Description
-mult <r> <g> <b> [a] [z]	Multiplies color on a per-channel basis.
-saturation <value>	Saturation change.

Channel Functions	Description
-colorspace Example: *shake bg.iff -colorsp rgb hls*	Converts images to a different colorspace. Indicate the source space and the destination space.
-copy <image> <clipmode> <channels>	Copies a channel from your listed image to the incoming image. *Clipmode* indicates the resolution you want, with 1 as the second image, 0 as the input image.
-forcergb	Forces a BW or BWA image into RGB or RGBA format. This is unnecessary in the Shake interface and can be handled by *FileOut* for output, but is awkward in the command line, thus this function's raison-d'etre.
-reorder <channels> Examples: *shake truck.iff -reorder aaaa* *shake truck.iff -reorder rgbl* *shake bg.iff -reorder grb*	Swaps channels. Using letter codes (r, g, b, a, z, l for luminance, and n for null), indicate what channel should go in the r, g, b, a, and z channels.
-setalpha <value>	Sets the alpha to your value. To remove an alpha channel from an image, enter 0.
-switchmatte <image>	Similar to *Copy*, except only the alpha is swapped. The returned image is premultiplied.

Compositing Functions	Description
-inside <maskImage>	An inclusion matte. Multiplies the incoming image by the alpha channel of the image you specify.
-isuba <Image>	Extracts the absolute difference between the two images.
-mix <Image> [percent]	Mixes two images together. 50 percent is half of each image.
-outside <maskImage>	An exclusion matte. Multiplies the incoming image by the inverse of the alpha channel of the image you specify.
-over <backgroundImage>	Puts the incoming image, assumed to be the premultiplied foreground, over the background.
-under <foregroundImage>	Puts the incoming image beneath the image you list, which is assumed to be premultiplied.
-screen <image>	Performs the *Screen* operation, which is good for reflections and glows.

Resizing Functions	Description
-addborders <xBorder> <yBorder>	Pads the image out with black around the edges.
-crop <left> <bot> <right> <top>	Crops the image to the two corners you specify. The origin is in the lower-left corner.
-fit <xRes> <yRes>	Resizes the image to the resolution you specify, maintaining the aspect ratio.
-resize <xRes> <yRes>	Resizes the image to the resolution you specify.
-window <left> <bot> <xRes> <yRes>	Crops the window at the specified lower-left corner, and to the resolution you specify.
-z <zoom>	A shortcut for *Zoom*; both X and Y are zoomed by the same amount.
-zoom <xZoom> <yZoom>	Zooms the image in X and Y independently.

Filter Functions	Description
-blur <xPixels> [yPixels]	*Blur*

Transform Functions	Description
-flip	Turns the image upside down.
-flop	Turns the image backward.
-pan <xPan> <yPan>	Pans the image.
-rotate <angle>	Rotates the image.
-scale <xScale> <yScale>	Scales the image without changing the resolution.

Video Field Functions	Description
-deinterlace <field> [mode]	Deinterlaces the image. 0 equals the even field, 1 equals the odd field. For mode: 0 = replication of the line below. 1 = interpolation of the missing line between two lines above and below. 2 = blur. 50 percent of the line, 25 percent each of the two lines above and below the missing line.
-evenonly <image> Example: *shake bg.iff -rotate 45 -evenonly bg.iff*	Executes the previous commands on the even fields of the image, and mixes it back in with the image you list.
-field <field>	Extracts just one field, creating a half-height image. 0 = even field, 1 = odd field.
-fldr <field>	Activates field rendering. 1 = odd/PAL, 0 = even/NTSC.

Video Field Functions	Description
-interlace <image> <clipMode> <field>	Interlaces the two images. When *clipMode* is set to 1, the BG resolution is taken; when set to 0, the incoming image resolution is taken. 0 = even field, 1 = odd field.
-oddonly <image>	Only executes the previous commands on the odd fields of the image, and mixes it back in with the image you supply.
-pulldown <image> <field> [offset] Example: *shake bus/bus2.40-79#.jpg 0 -t 1-50*	Acts as a *FileIn*, converts from 24 fps to 30 fps, interlacing the frames to add the extra frames. 0 = even field, 1 = odd field. The offset amount is the frame that the interlacing starts after the beginning.
-pullup <image> <field> [offset]	Acts as a *FileIn*. Removes the 3:2 pulldown interlacing, converts from 30 fps to 24 fps. 0 = even field, 1 = odd field. The offset amount is the frame that the interlacing starts after the beginning.
-swapfields	Switches the odd and even fields.

Other Functions	Description
-average <image> <sStart> <sEnd> <dStart> <dEnd> <bytes> <gain> <scale>	Acts as a *-bytes* <bytes>, averaging frames down from *sStart* and *sEnd* to *dStart* and *dEnd*. Usually use 1,1,1 for the last three arguments.
-bytes <bytes>	1 = 8 bits, 2 = 16 bits, 4 = float.

Examples

Looking at Images	Description
shake bg.iff sign_mask.iff	
*shake **	Uses the UNIX-style wildcard, *. This means "anything."
*shake *.iff*	Indicates "anything with an .iff extension."
shake bg.iff -compare sign_mask.iff	Compares the two images, using Option-click or Alt-click to drag between the two. For Linux, Shift-Control-click and drag.

Launching Flipbooks
shake alien/alien.#.iff -t 1-50
shake bus/bus2.#.jpg -t 40-79
shake bus/bus2.40-79#.jpg -t 1-40
shake fan/fan.@.iff -t 1-5

Cool Text Tricks	Description
shake -addtext %t -t 1-20	Prints time code.
shake -addtext %T -t 1-20	Prints full time code.
shake -addtext %f -t 1-20	Prints the current frame.

Cool Text Tricks	Description
shake -addtext %F -t 1-20	Prints the padded current frame.
shake -addtext "%D, %d %M"	Prints the current date.

Converting Image File Formats	Description
shake alien.#.iff -t 1-50 -fo temp.@.sgi	Writes 50 unpadded images in the .sgi format.
shake fan.@.iff -t 1-5 -fo fan.#.tif	Writes 5 padded images in the .tif format.

Adding and Removing Channels	Description
shake alien.0001.iff -lum -fo toto.iff	Creates a black-and-white image with an alpha channel.
shake alien.0001.iff -reorder rrra -fo tutu.iff	Creates a black-and-white image with an alpha channel.
shake tutu.iff -forcergb	Makes a 1-channel image into a 3-channel image without modifying the image.
shake alien.0001.iff -reorder rgbn	Removes the alpha channel.
shake alien.0001.iff -setalpha 0	Removes the alpha channel.
shake alien.0001.iff -reorder rgbl	Puts the luminance into the alpha channel.

Changing Bit Depth	Description
shake bg.iff -bytes 1 -fo bit8.iff	Converts the image to 8 bits (what this image was anyway).
shake bg.iff -bytes 2 -fo bit16.iff	Converts the image to 16 bits.
shake bg.iff -bytes 4 -fo bit32.iff	Converts the image to float.

Manipulating Fields	Description
shake -pulldown bus/bus2.40-79#.jpg 0 -t 1-50 -fo fps30.#.iff -v	3:2 pulldown, converting from 24 fps to 30 fps. The 1-50 is derived by subtracting 40 from 79, adding 1, and then multiplying that by 1.25.
shake -pullup fps30.#.iff 0 -t 1-40 -fo fps24.#.iff -v	3:2 pulldown converting from 30 fps to 24 fps. The 1-40 is derived by dividing 50 (the amount of images) by 1.25.

Averaging Frames	Description
shake -average bus2.40-79#.jpg 1 40 1 20 1 1 1 -t 1-20	Averages the 40 frames down to 20 frames.
shake bus2.40-79x2#.jpg -mix bus2.41-79x2#.jpg -t 1-20	Uses the frame numbering steps in the *FileIn* to get the averaging, and mixes them together with the *Mix* function.

Resizing Images	Description
shake bg.iff -z 3	Zooms the image up by 3.
shake bg.iff -z .333 *shake bg.iff -z 1/3*	Both zoom the image down by 3.

Resizing Images	Description
shake bg.iff -zoom 2 1	Zooms the image to twice as wide.
shake bg.iff -resize 720 486	Zooms the image to NTSC resolution.
shake bg.iff -fit 720 486	Zooms the image to NTSC resolution, maintaining the original aspect ratio.

Renumbering Clips	Description
shake bus2.40-79#.jpg -t 1-40 -fo toto.#.iff -v	Shifts frame numbering to start at 1.
shake bus2.#.jpg -t 40-79 -fo toto.#-39.iff -v	Also shifts frame numbering to start at 1.
shake bus2.#.jpg -t 40-79 -fo toto.118-#.iff -v	Reverses timing.
shake bus2.#.jpg -t 40-79 -fo toto.@.iff -v	Unpads the clip.
shake bus2.40-79x2#.jpg -t 1-20 -fo toto.#.iff -v	Halves the timing of the clip.
shake bus2.40-79x.5#.jpg -t 1-80 -fo toto.#.iff -v	Doubles the timing of the clip.

Compositing
shake truck.iff -outside sign_mask.iff -over bg.iff
shake bg.iff -under truck.iff

Animating Parameters	Description
shake bg.iff -blur "Linear(0,0@1,100@20)" -t 1-20	A linear animation from a value of 0 at frame 1 to a value of 100 at frame 20.
shake bg.iff -rotate "Linear(0,0@1,360@11)" -t 1-10	Animates a rotation from 0 degrees at frame 1 to 360 degrees at frame 11. Notice only up to frame 10 is rendered, hopefully giving a smooth loop.
shake bg.iff -rotate "Linear(0,0@1,360@11)" -t 1-10 -motion .5 1	Sets motion blur to half quality.
shake truck.iff -curve A "JSpline(1,0@1,10@5,300@15)" -pan A A -t 1-15	Creates a temporary curve named *A*. It has values of 0 at frame 1, 10 at frame 5, and 300 at frame 15. It also continues its slope after frame 15. This curve is then fed into the x and y pan parameters of the *Pan*.
*shake bg.iff -rotate "time*time" -t 1-75 -motion .5 1 -cpus 2*	Uses the *time* variable, placing it in quotes to multiply it by itself. As the frame count increases, the angle increases. Also, both CPUs are enabled for a 2-processor computer.

Animating Parameters	Description
shake truck.iff -pan "cos(time.5)*100" "sin(time)*50" -motion .5 1 -t 1-13*	Uses the *cos* and *sin* functions to animate the truck in a figure eight.
shake bg.iff -rotate "cos(time.5)*50+25" -t 1-13 -motion .5 1*	

Substituting Values with -curve

If you have the following in a script called *myscript.shk*:

Text1 = Text(720, 486, 1, {{ stringf("myVal = %d %s", myVal, slatestring); }}, "Utopia Regular", 100, xFontScale, 1, width/2, height/2, 0, 2, 2, 1, 1, 1, 1, 0, 0, 0, 45);

you substitute values in with the *-curve* option:

shake -curve int myInt 5 -curve string slatestring "SuperSlate" -script myscript.shk

and you get a nice image that says "myVal = 5 SuperSlate".

Or you can use:

shake -curve int myVal 5 -curve string slatestring "SuperSlate" -text 720 486 1 ' :stringf(\\\"myVal = %d\\n%s\\\", myVal, slatestring) ; '

Note the extra backslashes: This quotes it once for the shell and then again for Shake. The shell sees \\ and makes it \ then \" and makes it " so you get a result of \" going into the Shake wrapper script, that is then passed properly into the Shake executable. Whew.

Tips

The following section contains tips and tricks for command-line usage.

File Completion

The following is a shortcut to typing file names using the file completion feature in the shell. When you press Tab, all potential files that match what you type are listed. For example:

```
shake Tab
```

lists

```
shake truck/
```

Press Tab again to list all three image files within that directory. If you type another "t," that is:

```
shake truck/Tab
```

lists

```
shake truck/truck.iff
```

Therefore, you can type the entire line with very few keystrokes. If you type the following:

```
shake Tab Tab Tab Tab Tab
```

the following line is listed:

```
shake truck/truck.iff truck/bg.iff truck/sign_mask.iff
```

So press Return.

Repeating Previous Commands

There are two ways to repeat previous commands:

- Press the Up Arrow on the command line. Each time you press it, the previous command is listed, stepping back through your history. Change portions of the command with the Left Arrow and Right Arrow keys. Press the Down Arrow key to take you to the next command in the history list.
- Press the ! key. Press !! to repeat the last command (although pressing the Up Arrow key is easier). Type *!s* to repeat the last command that started with "s."

Wildcards

Use wildcards in the command line so you don't need to type much. The following table lists some of the wildcards.

Wildcards	
*	*shake **
Matches everything of any length.	Shows all files within that directory simultaneously.
	*shake *.iff*
	Shows all files within that directory with a .iff extension.
	*shake *g**
	Displays *bg.iff* and *sign_mask.iff*.
?	*shake alien.000?.iff*
This is used to match anything for only that position.	Displays the first nine images simultaneously.
	shake alien.002?.iff
	Displays all frames in the twenties.
[range-range]	shake alien.000[1-5].iff
This can describe a range, between letters or numbers.	This will display the first five images simultaneously.
	*shake [l-z]**
	Displays all images that start with the letters l to z, lowercase.

Math and Expressions

When performing math on the command line, enclose the math in "quotation marks":

```
shake truck.iff -pan "cos(time*.5)*100" "sin(time)*50" -motion .5 1 -
    t 1-13
```

Multiple Words

To use more than one word in a parameter that expects a string (letters), again, use "quotation marks":

```
shake -addtext "kilroy was here"
```

Note: This comes up frequently for systems running Mac OS X.

Index

A

B

C

D

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z