



Shake 3 Tutorials



Apple Computer, Inc.

© 2003 Apple Computer, Inc. All rights reserved.

Under the copyright laws, this manual may not be copied, in whole or in part, without the written consent of Apple. Your rights to the software are governed by the accompanying software license agreement.

The Apple logo is a trademark of Apple Computer, Inc., registered in the U.S. and other countries. Use of the keyboard Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

Every effort has been made to ensure that the information in this manual is accurate. Apple Computer, Inc. is not responsible for printing or clerical errors.

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

Apple, the Apple logo, Final Cut, Final Cut Pro, FireWire, Mac, Macintosh, Nothing Real, QuickTime, Shake, Tremor, and TrueType are trademarks of Apple Computer, Inc., registered in the U.S. and other countries.

Adobe is a trademark of Adobe Systems Inc.

Cineon is a trademark of Eastman Kodak Company.

Maya, Alias, Alias/Wavefront, IRIX, and O2 are trademarks of SGI Inc.

3ds max is a trademark of Autodesk Inc.

Softimage and Matador are registered trademarks of Avid Technology, Inc.

Times is a registered trademark of Heidelberger Druckmaschinen AG, available from Linotype Library GmbH.

Other company and product names mentioned herein are trademarks of their respective companies. Mention of third-party products is for informational purposes only and constitutes neither an endorsement nor a recommendation. Apple assumes no responsibility with regard to the performance or use of these products.



Contents

1	Lesson One: Basic Tutorial	5
	Getting Started	5
	Tutorial Summary	5
	Mac OS X Notes	6
	The Basic Tutorial	6
	Loading Images	9
	Viewing Images, Parameters, and Channels	12
	Launching a Flipbook	16
	Basic Compositing	17
	Windowing Conventions	21
	Setting Resolution	21
	Tuning Parameters	25
	Transforming Images	29
	Fading an Element	31
	Rendering a Sequence to Disk	32
2	Lesson Two: Intermediate Tutorial	37
	Tutorial Summary	37
	Slipping Time Sync	37
	Optimization	40
	Motion Blur	43
	Color Correction With the ColorCorrect Node	49
	Animation Curves	53
	Color Correction Using Other Nodes	55
	Applying a Shadow Element	61

3	Lesson Three: Depth Compositing	71
	Tutorial Summary	71
	The City Composite	71
	A Few Viewer Tips	73
	Viewing the Z Channel	74
	Preparing the Z Channel	76
	Creating the ZCompose	77
	Matching the Blimp and Background Colors	79
	Adding Fog	82
4	Lesson Four: Using Local Variables With Expressions	85
	Tutorial Summary	85
	The Fan Composite	86
	Animating the RGrad	88
	Adjusting Clip Length in the Time View	89
	Using Local Variables With Expressions	92
	Using RBlur	96
	Concatenation of Color Corrections	105
	Adding Motion Blur to Pre-Animated Elements	108
5	Lesson Five: Using Keylight	115
	Tutorial Summary	115
	Part One: Using Keylight to Pull a Key	115
	Testing the Mask With a Viewer Script	117
	Adjusting the Mask With Parameters	119
	Masking	120
	Color Correcting the Foreground Image	123
	Part Two: Using Keylight	126
	Using fgBias to Remove Blue Spill	129
	Using a Holdout Matte	130
6	Lesson Six: Using Primatte	135
	Tutorial Summary	135
	The Basics of Pulling a Key in Primatte	135
	Inner Mechanics of Primatte	138
	Masking Primatte	142

Spill Suppression in Primatte 145

Alternatives 150

7 Lesson Seven: Tracking and Stabilization 151

Tutorial Summary 151

Tracking and Stabilizing Nodes 151

Stabilizing an Image Sequence 152

Converting Stabilization Data to MatchMove Data 156

Using the MatchMove Node 158

Position the Foreground Element 165

Color Correct the Foreground Element 168

8 Lesson Eight: Making a Macro 177

Tutorial Summary 177

What Is a Macro? 177

Creating a Handmade Macro 179

Saving and Testing the Macro 185

Adding a Button to the Interface 187

How to Set Slider Ranges 189

Creating Macros With MacroMaker 191

Creating Sliders in MacroMaker 194

9 Lesson Nine: Creating a Clean Plate 197

Tutorial Summary 197

Creating a Clean Plate 197

Paint With the Reveal and Clone Brushes in the QuickPaint Node 203

10 The Cookbook 205

Cookbook Summary 205

Good Habits 206

Bad Habits 208

Coloring Tips 210

Filtering Tips 213

Keying Tips 214

Layering Tips 220

Transform Tips 224

Depth Tips	226
Text Treatments	227
Expressions	232
Cookbook Macros	237
Command-Line Macros	237
Image Macros	239
Color Macros	242
Filter Macros	245
Key Macros	246
Transform Macros	247
Warp Macros	249
Other Macros	250
Using Environment Variables for Projects	254

Lesson One: Basic Tutorial

Getting Started

This first tutorial follows the tradition of “read in images—composite the images together—render the images” to give you an idea of what Shake is about. The Intermediate tutorial section discusses parameter animation, basic color correction, node manipulation, and other things that beat puttering around the house. The Advanced tutorial section discusses how to track, pull keys, work with Z channels in exceedingly clever ways, and how to work with expressions and macros.

In addition to the tutorials, it is highly suggested that you explore the *Shake 3 Reference Guide*. The reference guide contains information about color correction, keying and spill suppression, masking, transforms, caching, premultiplication, bit depth, logarithmic color space, and other Shake features. For example, Chapter 10, “Using Masks,” follows a tutorial format.

Tutorial Summary

- Mac OS X Notes
- The Basic Tutorial
- Loading Images
- Viewing Images, Parameters, and Channels
- Launching a Flipbook
- Basic Compositing
- Windowing Conventions
- Setting Resolution
- Tuning Parameters
- Transforming Images
- Fading an Element
- Rendering a Sequence to Disk

Mac OS X Notes

The following information relates to Shake on Mac OS X and its documentation.

Use a three-button mouse

A three-button mouse is required to run the software. Shake also supports the middle scroll wheel of a three-button mouse.

The **Delete** key (immediately below **F12**) on the Macintosh system keyboard is equivalent to the **Backspace** key on Linux/IRIX systems. The Macintosh **Del** (“del”) key (the small key in the group of six keys, near **Page Down**, **Page Up**, and **Home**) is equivalent to the Linux/IRIX **Delete** key.

If you are using a smaller Macintosh keyboard without the **Del** key, use **Opt+Delete** (again, the key below **F12**).

Platform keyboard differences

Keyboard commands may differ between the Macintosh platform and the IRIX or Linux platform. In most cases in the documentation, the Macintosh keyboard command is cited first, followed by the Linux and IRIX command. The two options are separated by a forward slash. For example, “To pan the Time Bar, press the middle-mouse button and drag, or press **Opt-click / Alt-click** and drag.”

Command versus Ctrl

You can use **Command** or **Ctrl** interchangeably for hot keys on the Macintosh platform. For example, use **Command+C** or **Ctrl+C** to copy an object.

Launching Shake in the Terminal

Macintosh OS X wraps up binaries and its contents into one icon in the Finder. Click the Shake icon in Macintosh OS X to launch Shake. You can right-click the Shake icon to obtain menu options. For example, right-click the Shake icon and select Show Package Contents to open the subdirectories. Alternatively, use the Terminal to navigate down inside *shake.app* to *shake.app/Contents/MacOS/* to find the actually binary.

The Basic Tutorial

In this tutorial, composite four rendered 3D images over a gradation.

Note: The elements are from a swell animation project called “Vanilla Pudding,” from Wild Brain, Inc. in San Francisco. These folks are great, not least because they were kind enough to pass on footage for this tutorial, but also because they like to mix 3D and 2D animation. So extend a Laurel and Hardy handshake to Karen Ansel and Nicolas Weigel of Wild Brain, Inc. Backslapping aside, this version pales in comparison to the actual “Vanilla Pudding” shot (oops, more backslapping). This project has been trimmed down in complexity.

For those of you into narrative motivation, Vanilla, our cubically cranial gal, does not take too kindly to some other kids who have torn the wings off a butterfly. In the shot prior to this scene, Vanilla pulls the gun out of her nethers. Here, she takes aim with the aforementioned gun and fires. The missile, by the way, is a miniature hippopotamus that drools.

Frame 1

Frame 28

Frame 56



Launching Shake

To launch Shake, do one of the following:

- On the Macintosh platform, double-click the Shake icon.

Note: Do not remove the Shake binary from its standard install directory. You may place it in the Dock, however.

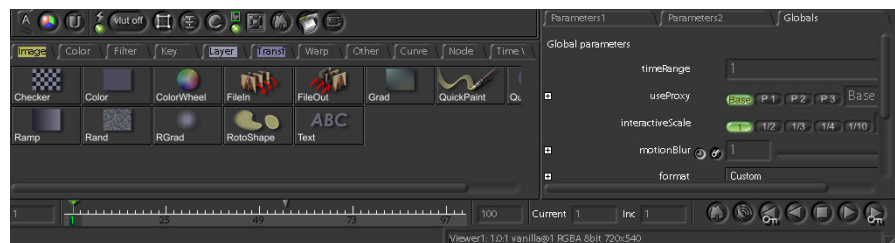
- On the Linux or IRIX platform, type the following in any shell:


shake

Note: On an OS X system, you must type the complete path to Shake (*/Applications/Shake3/shake.app/Contents/MacOS/shake*), or have environment variables set to run Shake from the Terminal. For more information, see “Setting Environment Variables for Shake” in Chapter 17 of the *Shake 3 Reference Guide*.

Contextual Help

An important feature to note is the Shake contextual help window. As you pass the cursor over buttons and parameters, a brief message appears that states their function, as well as the relevant hot key. The help window is located in the lower-right portion of the interface.



Right-click to view contextual menu functions. For example, right-click the Viewer Channel button  (the button that looks like the Martian eyeball from “War of the Worlds”) in the lower-left corner of the Viewer, and the menu appears. The letters that appear next to the contextual menu options are the hot keys for each choice. Therefore, the right-mouse button can also be considered as contextual help.

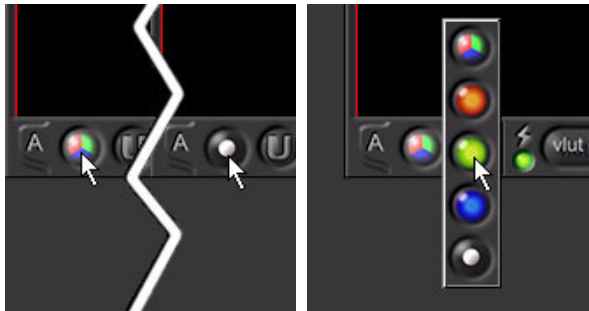


There are different click and hold button behaviors, for example:

- Click the Viewer Channel button in the Viewer to toggle between the RGB and the Alpha Views.
- Click and hold the Viewer Channel button to select a specific channel view.

Click

Hold



Overriding the Default Button Choices

To override the default choices, **Ctrl**-hold and select your next option. For example, the default button behavior for the Viewer Channel button is to toggle between RGB View and Alpha View. To modify the behavior to toggle from RGB View to Red View, then to Alpha View and back to RGB View, perform the following steps:

- Make sure the Viewer Channel button is set to RGB View.
- **Ctrl**-hold the button, select the Red View button, and release the mouse.
- **Ctrl**-hold the button, and select the Alpha View button. Because Alpha View already toggles to RGB View, you do not have to specify this behavior.

To save this behavior, choose File > Save Interface Settings.

Loading Images

Every image process in Shake is done by creating and linking nodes in the Node View. The nodes are located in the Tool Tabs (Image, Color, Filter, etc.).

To load images, select the *FileIn* node from the Image tool tab (the tutorials book and reference guide notation for creating nodes is “the Image–*FileIn* node”).




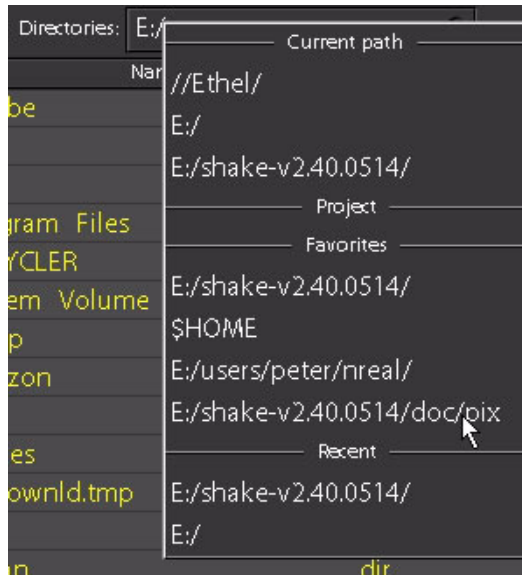
Do not confuse the *FileIn* node with the Load and Save buttons at the top of the software—the Load and Save buttons are for loading and saving scripts.



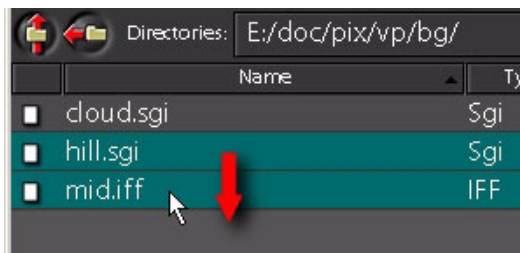
- 1 In the Image tab, click *FileIn* to launch the Browser. The “Load image or sequence browser” window appears.
- 2 Browse to the *Shake3/doc/pix/vp* directory.

Browsing Tips

- Use the Up Directory button  or press **Delete** (Macintosh) / **Backspace** (IRIX or Linux) to move up one level.
- Double-click a directory to move down one level.
- Click in the file list area, and press the **Up Arrow** and **Down Arrow** keys to scroll up or down, or type the first letter of the file you want.
- Middle-mouse or **Opt-click** / **Alt-click** to scroll the window—you do not have to use the scroll bar on the right.
- The tutorial directories are stored in the Directories pull-down menu, as well as recently visited directories and favorite project directories.



- 3 Browse to *doc/pix/vp/bg*, and drag-select *hill.sgi* and *mid.iff* (*cloud.sgi* is not needed for this exercise).



- 4 Click Next at the bottom of the Browser.

Both images are read in. The Next button allows you to continue selecting multiple image files from different directories without having to create individual *FileIn* nodes one at a time. After you click OK, each image and image sequence you selected will be represented by a separate FileIn node in the Node View.

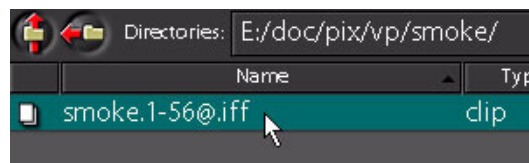
- 5 Press **Delete / Backspace** (or click the Up Directory button) and navigate to the *smoke* directory—ignore the *shadow* directory for now.

The *smoke* directory contains a sequence of images ranging from 1 to 56. The @ sign indicates these are non-padded frames. To see the files listed individually, turn off the Sequence Listing toggle at the bottom of the Browser.

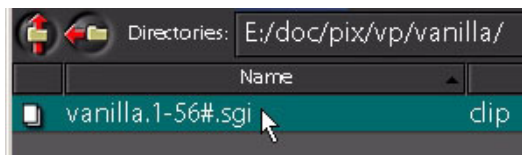
- 6 To load the sequence, activate Sequence Listing.



- 7 Select the files as a sequence. Press the **Space bar** (rather than the Next button), as the Next button shortcut.



- 8 Navigate to the *vanilla* directory, and read in the vanilla sequence. To finish loading images, either double-click the vanilla sequence (which also has 56 frames), or select the sequence and click OK at the bottom of the Browser.



If you double-clicked before reaching the vanilla images, click the Image-*FileIn* node to launch the Browser.

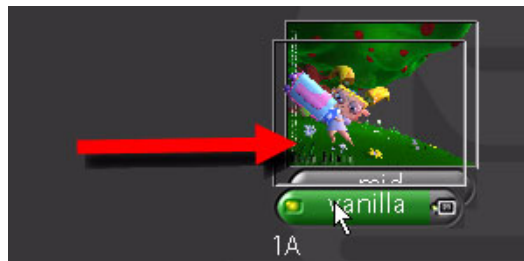
Viewing Images, Parameters, and Channels

Four nodes appear in the Node View. Each node is a function that can be viewed or modified. In this exercise, all of these are *FileIn* nodes that load images.



Image thumbnails appear on the nodes. As shown in the *mid*, *vanilla*, and *hill* images in the above illustration, the thumbnails also indicate transparency if there is an accompanying alpha channel. The *smoke* element is blank because frame 1 is black.

Here's a swell trick to impress your date with next Saturday: Drag the *vanilla* node over the *mid* node in the Node Tree—it does a mini-composite. Does this help you composite or change your tree at all? Nope, but it's cool, isn't it?



To do an actual composite, you must connect the nodes. This happens in a moment, so stop fidgeting.

A bit about thumbnails:

- The thumbnails represent the frame at the time of file loading.
- To refresh to the current frame, select the node and press **R** with the cursor in the Node View.

- To view the alpha channel, place the cursor over the thumbnail and press **A**. To return to RGB, place the cursor over the thumbnail and press **C**.



- Any node can have a thumbnail—select the node and press **T**.
Note: The thumbnails do not dynamically update (see below).
- To hide thumbnails, select the nodes and press **T**. Press **T** again to show the thumbnails.
- Additional controls for the thumbnails are located in the Globals tab.

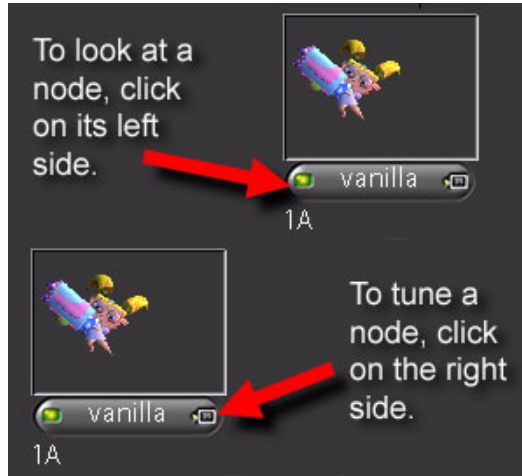
Go ahead, ask the question. It's obvious you want to ask it... "Why aren't the thumbnails always updating?" Shake does not dynamically update thumbnails because it is either inefficient or inaccurate. For example, if you are working on 6K plates, do you really want to spend all of your time resizing 6K plates down to tiny icons? (That's rhetorical, by the way.) As another example, in a script of 900 nodes, the drag on the script would be phenomenal. The most efficient and accurate way to check a node is to load it into the Viewer.

This nicely brings up the next subject, loading and viewing nodes.

Viewing a Node or Loading its Parameters

- To load a node into the Viewer, click on the left side of the node.
- To load the parameters into the Parameters tab, click on the right side of the node.

- While viewing one node, you can tune another node.



In the above illustration, the *vanilla* image is the currently viewed node, as indicated by the highlighted button on the left side of the node. The 1A that appears below the node indicates it is in Viewer 1, buffer A. The tiny text field on the right side indicates the vanilla parameters are also loaded. OK, true, there is only the one node in the illustration. Just try to get past that.

As mentioned above, use the help window to get information on nodes. As you pass the cursor over a node (no need to click), the resolution, bit depth, node name and type, and channels are displayed. For example, you can see that the *vanilla* image is a *FileIn* node named *vanilla*, is 8-bit, has RGBA channels, and is 720 x 540 pixels.

vanilla (FileIn) 8bit RGBA 720x540

Look at the Different Channels in the Viewer

Use the View Channels button  to toggle to different channels. For example, click View Channels to select the alpha channel .



However, nobody actually uses that button. Remember when you right-clicked on the View Channels button and it listed the hot keys? Use the hot keys instead when the cursor is in the Viewer to quickly view a channel (**C, R, G, B, A**). All the cool kids in the back row are doing it.



Setting the Frame Range

There are two places to set the frame range in Shake. The first and most important is in the Globals tab. The Globals tab lists all script settings—the frame range, proxy settings, default resolutions, GUI settings, and quality settings. The normal Parameters tab is a listing of parameters for a specific node.

There are two ways to show the Globals tab:

- Click the Globals tab. (Gee, how novel.)
- Double-click an empty area in the Node View.

The first parameter in the Globals tab is the *timeRange* parameter. This frame range determines what frames are rendered. Although it is saved in the script, you can override it in the command line. Click the Auto button to examine the *FileIns* and determine the range automatically.

To enter the time range:


- Click Auto in the timeRange parameter.



The timeRange parameter is extremely flexible and can be manually edited:

Entry	Calculates
1-56	Frames 1 to 56.
20-30	11 images from frames 20 to 30.
1-56x3	Frames 1, 4, 7, etc.
1,10,20-30x2	Frames 1, 10, 20, 22, 24, 26, 28, and 30.

Launching a Flipbook

On the Viewer toolbar, click Flipbook  to launch a flipbook render of any node. This launches a render into a floating window. (If you are testing a *FileOut* node, the actual *FileOut* is not executed. To render to disk, use the Render command (right-click in the Node View, or use the Render menu). Otherwise, the sequence is rendered into memory and you can play it back.

Flipbook Controls

The following table contains several Flipbook controls.

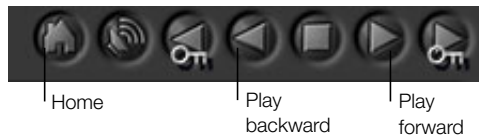
Entry	Calculates
. (Period; think of it as the > key.)	Play forward.
, (Comma; think of it as the < key.)	Play backward.
R, G, B, A, C	Show the red, green, blue, alpha, and color channels.
Shift -drag	Scrub through the animation.
Esc	Close the window.

Once closed, all links to the Flipbook are lost—you cannot save or use the images again. You can have as many Flipbooks as memory allows. You can stow the Flipbooks and retrieve them later, but they take up memory.

You can also use the Time Bar to indicate a frame range and obtain playback. The Time Bar displays the range that you want to concentrate on, and does not get saved into the script.

To set the playback range for the Time Bar:

- 1 Place the cursor over the Time Bar and press **Home** on the keyboard (or click the Home button on the Time Bar) to load the script's timeRange into the Time Bar.
- 2 Press Play on the Time Bar to play through the sequence. This differs from the Flipbook in that it is not real time, but does place the images into the memory cache, optimizing future calculations in the interface.



For more information on the Flipbook, see “The Flipbook” in Chapter 2, “The Shake User Interface,” in the *Shake 3 Reference Guide*.

Basic Compositing

Each tutorial element contains an alpha channel that makes the images ready (and easy) to composite.

Begin the composite:

- 1 Click the *smoke* node in the Node View to select the node.
- 2 In the Layer tab, click the *Layer* node.

The different layer nodes (such as *Atop*, *Over*, and *Outside*) are combined in the *Layer* node. There is no difference between a *Layer* node with the operation parameter set to *Over* and an *Over* node.

Because the *smoke* node was selected in the Node View when the *Layer* node was added, the *Layer1* node is attached to the *smoke* node.

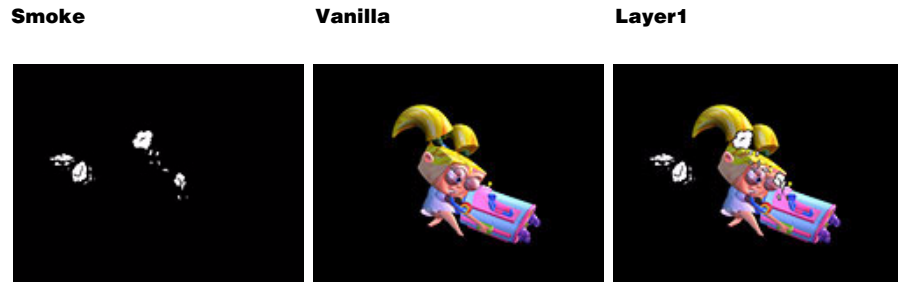


The *Layer1* node has two inputs on the top of the node. The *smoke* node is attached to the first input.

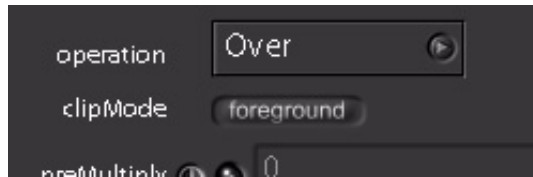
- 3 Drag the second input from the top of the *Layer1* node to the bottom of the *vanilla* node to connect the two nodes. You can also drag from the bottom of *vanilla* to the second input on *Layer1*.



Information flows downward in the Node View like a waterfall—the image information is passed from the *smoke* and *vanilla* nodes, and fed into the *Layer1* node to perform the composite.



The operation parameter is listed in the *Layer1* parameters. *Over* is the most frequently used operation, and assumes an alpha channel exists in the first image input. You can select a different compositing mode with the pull-down menu, but for this composite, stick with *Over*.



The node compositing logic is similar to language: “Input 1 is Over Input 2.” Therefore, if the inputs are switched, the compositing order is reversed.

- 4 Go to frame 18.
- Note:** To move frames, use the **Left Arrow** and **Right Arrow** keys, or enter a specific frame number in the Current text field on the Time Bar.
- 5 To switch inputs, grab the bottom portion of the connecting line (called a noodle). When the noodle is selected, it turns red. Drag the noodle to the other node input.



The node logic is switched from *smoke* over *vanilla* to *vanilla* over *smoke*.



This is bad.

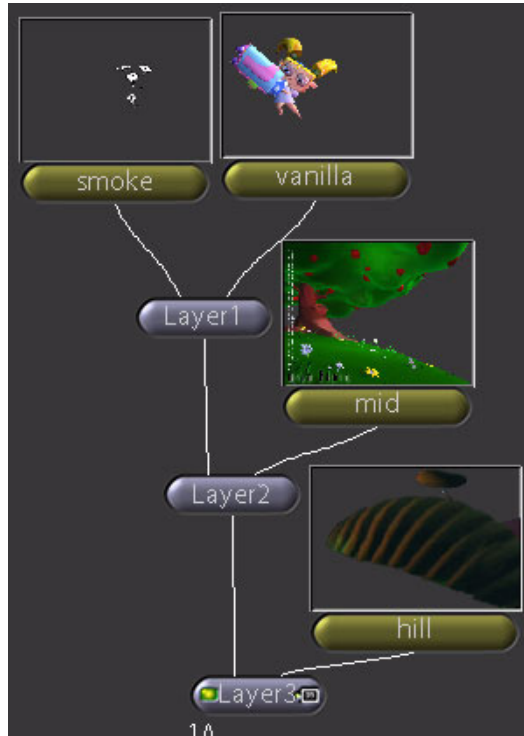
- 6 Switch the inputs again (or press **Command+Z** / **Ctrl+Z** to undo your previous operation). The *smoke* image is over *vanilla*.

To Break a Connection

Place the cursor over the noodle and when the noodle turns red, press **Del** (Macintosh; near the **Home** and **End** keys) / **Backspace** (IRIX and Linux). You can also **Ctrl**-click the noodle.



- In the Node View, select the *Layer* node and add two more *Layer* nodes. Use the following illustration as a guide to connect the background elements.



The composite result appears.



Windowing Conventions

The following table lists Shake's windowing conventions.

Function	Hot Key	Notes
Expand a window full screen	Space bar	Press the Space bar again to zoom back to normal view.
Pan a window	Middle-mouse button or Opt -click and drag / Alt -click and drag	Works in all windows, including the Browser.
Zoom a window	Command / Ctrl -middle-mouse button or Ctrl+Opt -click and drag / Ctrl+Alt -click and drag	Works in the Node View, the Viewer, the Time Bar, and the Curve Editor.
Zoom in on a Viewer	+/- (by the Delete / Backspace key)	Gives you an integer-based zoom so you have fewer round-off artifacts on your display. The zoom follows the cursor.
Reset a View	Home	Works in the Node View, the Viewer, the Time Bar, and the Time View.
Resize a pane		Grab the border between two panes and drag to resize the window.

Setting Resolution

Shake is not only resolution independent, but can also dynamically change resolution during the compositing process. For example, you can simultaneously output a HD image and a 601 video image. Won't that impress the family on your next holiday visit?

There are three ways to set your composite resolution:

- Composite elements over an image of the desired resolution.
- In the Transform tab, use the *Fit*, *Resize*, or *Zoom* node to scale your images.
- In the Transform tab, use the *Crop*, *Viewport*, or *Window* node to crop into or out of your window. These nodes do not resize the image, but instead reset the framing of the image.

In this example, use the first technique and composite elements over an image of the desired resolution.

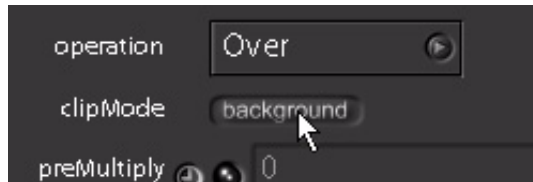
To set the resolution:

- 1 Click the left side of the *bill* node to load the image into the Viewer. The *bill* image is 370 x 250 pixels.

Since the *bill* node is connected to *Layer3* (step 7 above), the resolution changes to 720 x 540. Each *Layer* operation (that is, every node in the Layer tab, not just the *Layer* node itself) can set the output resolution.

- 2 Double-click the *Layer3* node in the Node View to simultaneously view the node and load its parameters.

One of the parameters is called *clipMode*. When *clipMode* is set to foreground, the resolution of the first image is used. When set to background, the resolution of the second image is used.



- 3 To ensure the resolution is 720 x 540 pixels, set *clipMode* to foreground.

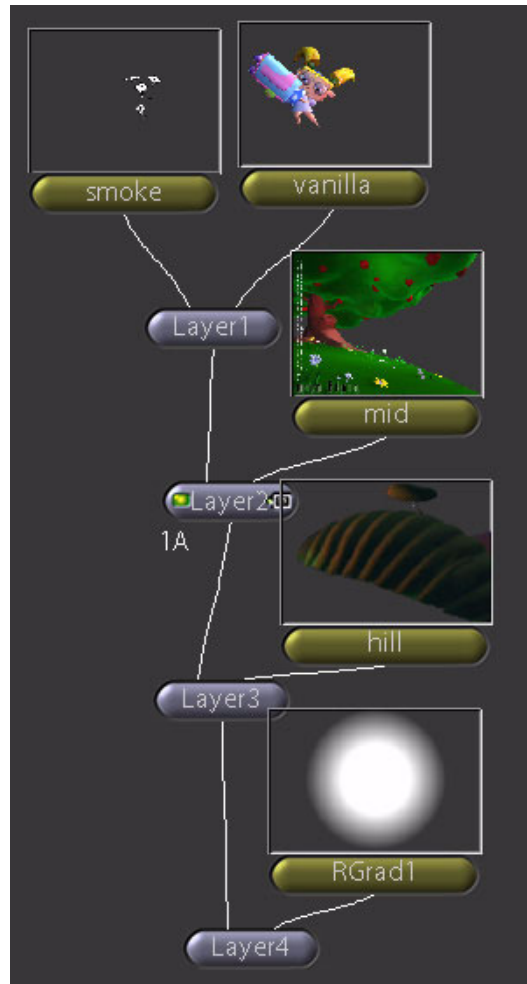


Next, create a sky element for the background with a radial gradation that is generated by an *RGrad* node. Nodes that generate images, such as *RGrad*, *Ramp*, *ColorWheel*, and *RotoShape*, contain width and height parameters to set resolution.

To create the radial gradation:

- 1 Create an Image-*RGrad* node.
- 2 Using the following illustration as a guide, attach the *RGrad* node to the node tree with a fourth *Layer* node.

Note: The node trees in this exercise are in a vertical layout for illustration purposes. Trees generally tend to migrate diagonally down.



Notice that the *RGrad1* width and height are set to 720 x 486, and *Layer3* has a height of 540 pixels.

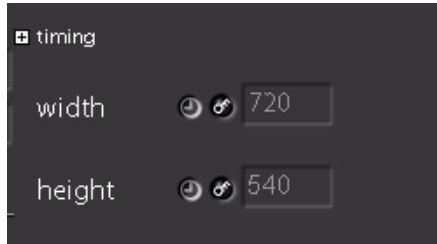
To match the height of *RGrad1* to the height of *Layer3*, you have several options:

■ **Option 1: Do nothing.**

A tempting and acceptable solution is to not worry about it at all. The Shake engine is designed to allow you to do this because of the Infinite Workspace. The Infinite Workspace ensures that elements are never clipped due to framing.

- **Option 2: Edit the RGrad parameters.**

Click in the height text field to manually edit the height parameter of the *RGrad*.



This is fine and dandy, but what about the next time you create an image node? Shake has a default width and height in the format pop-up menu in the Globals tab, so you do not have to manually change a node each time it is created. You can select from presets (Cinemascope, Academy, different flavors of NTSC, PAL, HD), or open the subtree to set your own.

- **Option 3: Set the defaultWidth and defaultHeight.**

In the Globals tab, open the format subtree. Set the *defaultHeight* to 540.

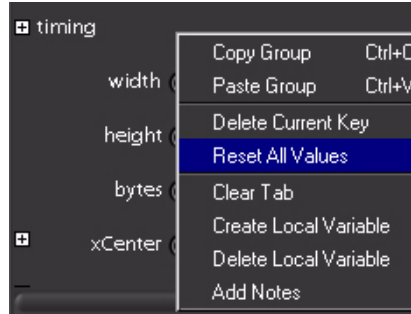


Setting the *defaultWidth* and *defaultHeight* in the Globals tab is the best solution, as it is very rare to have a seven-node composite. A normal composite typically requires multiple *RGrad*, *RotoShape*, and *QuickPaint* nodes to help with masking.

Changing the *defaultHeight* parameter does not change the height of the current *RGrad*, but the new setting is applied to any subsequent *RGrad* nodes.

To reset the *RGrad* parameters:

- In the Parameters tab, right-click and select *Reset All Values*.

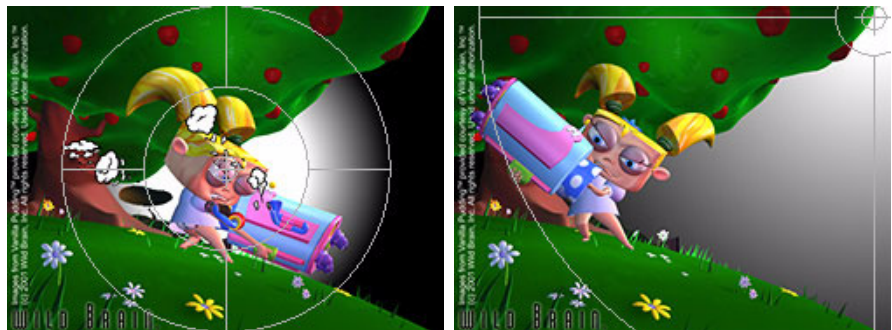


Tuning Parameters






This section discusses the different ways to tune the parameters of a node. Nodes can have text fields for numerical entry, widgets to help you select color, or onscreen controls for interactive placement. The *RGrad* node has all three types. Since the fine glowing ball in the background does not convey a feeling of the sky on a nice, warm, non-compositing workday, modify *RGrad1* in the following steps.

To position the circle using the onscreen controls:

- Grab the center and move it to the upper-right corner. Grab the rings to adjust the radius and drop-off.



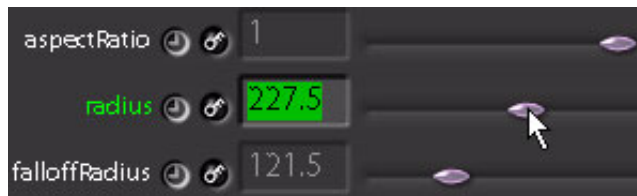
Controls for the onscreen controls are located in the Viewer toolbar. The following table lists some helpful buttons.

Button	Function	Notes
	Autokey	When enabled (highlighted green), a keyframe is created for parameters linked to onscreen controls. (The Autokey button is not used in this exercise.)
	Delete Keyframe	In case you enabled Autokey, there is also Delete Keyframe. Go to the frame that has the keyframe you want to delete, and click the Delete Keyframe button.
	OSC Color	Sets the color for the onscreen controls.
	OSC Display Toggle	Onscreen controls are always displayed.
	OSC Display Toggle	Onscreen controls are invisible when moved. Hold down the OSC Display toggle to access.
	OSC Display Toggle	Onscreen controls are turned off.

Notice that when the parameters are modified, the parameters adjust to return the numerical data.

You can modify a text field in several ways:

- To modify a number, drag the slider to modify the value.



- To select text, click once to select the text field. Double-click for a type insertion point.



- To use virtual sliders, **Ctrl**-drag in the text field to activate the virtual sliders. This gives you more precision and the ability to go beyond the slider limits.



- To nudge a value up or down, place the cursor over the text field and hold down **Shift**, **Ctrl**, or **Alt** and press the **Left Arrow** and **Right Arrow** keys. Press **Shift+Arrow** to raise or lower the value by a factor of 10. **Ctrl** is the normal increment, and **Opt / Alt** is .1x the normal increment.



Note: If you are using a stylus, click to the Globals tab. In the *guiSettings* subtree, enable *virtualSliderMode*. This assists you with the virtual slider. You should also assign one of the stylus buttons as the right-mouse button.

Animating parameters is explained in the next tutorial.

The *RGrad* node has two Color Picker buttons—one button to control the center color of the *RGrad*, and one button to control outside the center of the *RGrad*.



Using the Color Pickers, you can edit color in several ways:

- Enter your color value numerically into the three text fields.
- Ensure the Color Picker button is active (it has a yellow ring around it), and select the color with the Color Wheel.
- Ensure the Color Picker button is active, and drag in the Viewer to select the color.
- Use the virtual color pickers.

To use the virtual color pickers, hold down the key on the keyboard that corresponds to the channel you want to modify and drag the cursor left and right anywhere on that row. You can select:

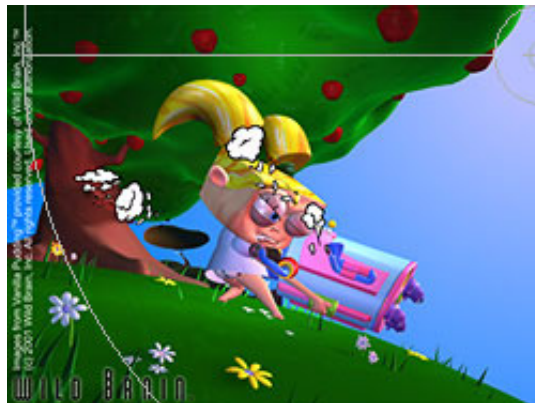
(R)ed, **(G)**reen, **(B)**lue, **(O)**ffset, **(H)**ue, **(S)**aturation, **(V)**alue, **(L)**uminance, **(C)**yan, **(M)**agenta, **(T)**emperature



With this mechanism, you can effectively group all three channels by holding the **O** (offset) key down and sliding left and right. (It's more fun than feeding peanut butter to a dog.)

Color correct the *RGrad*:

- Using one of the methods described above and the following illustration as a guide, change the *RGrad* to a more realistic sky-blue color.



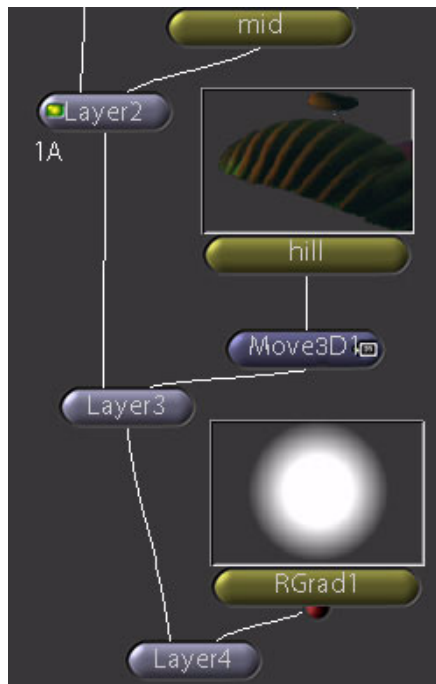
Transforming Images

Those of you who were the type to sit in the front row of school and give apples to the teacher (and no doubt incurring a great deal of nasty comments from the back of the class) may have noticed that the *hill* element—a smaller image resolution than *mid*—is tucked behind the *mid* element. As always, there are several ways to correct this problem. One solution is to use the Transform–*Move3D* node.

To transform the hill image:

- 1 In the Node View, select the *hill* node.
- 2 In the Transform tab, click *Move3D*.

A *Move3D* is inserted between *hill* and *Layer3*.



- 3 Click once on the left side of *Layer4* to ensure you are viewing *Layer4*. (Double-clicking loads the parameters and views the image, which you don't want, since you want to still tune the *Move3D* node.)

Note: You can view any node and adjust the parameters of any other node.

The *Move3D* onscreen controls appear around the frame of the image's resolution. Remember, *bill* and *Move3D1* are at 370 x 250 resolution. The controls match that resolution.



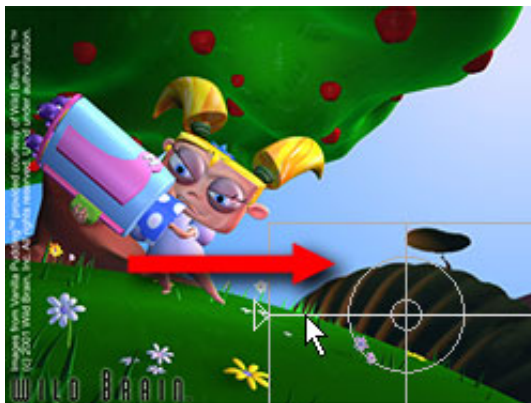
- To pan the element, grab the horizontal or vertical crossbars. Do not grab the center—it is the center of rotation control.
- To scale uniformly, grab a corner and drag.
- To scale in one direction only, grab an edge.
- To rotate, turn the larger, outside circle.

The smaller circle is the center-of-rotation/scale. You do not need to move it in this exercise.

When the cursor is over a section of the controls, the controls are slightly highlighted.

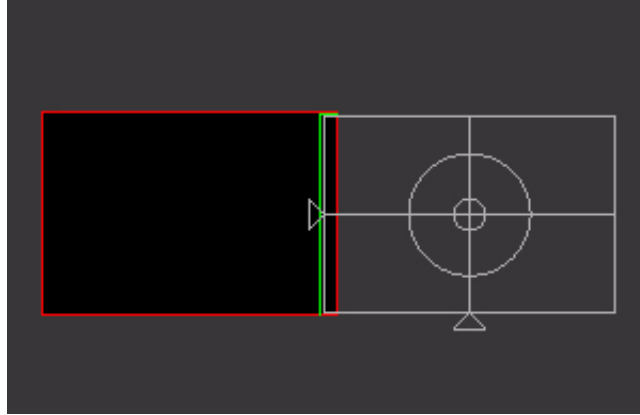
Note: To reset a node, right-click in the Parameters area and select Reset All Values.

- 4 Using one of the techniques mentioned above, pan the *bill* image to the right side of the image.



- 5 Click on the left side of the *Move3D1* node.

Because the controls have panned most of the image out of frame, the frame is mostly black.



Due to the Shake Infinite Workspace, with the exception of approximately four nodes, there is never any clipping of an image if it goes out of frame—you can always retrieve it later. View *Layer4* to confirm the entire hill image is still there.

The Infinite Workspace is your best friend. For more information, see “About the Infinite Workspace” in Chapter 4 of the *Shake 3 Reference Guide*.

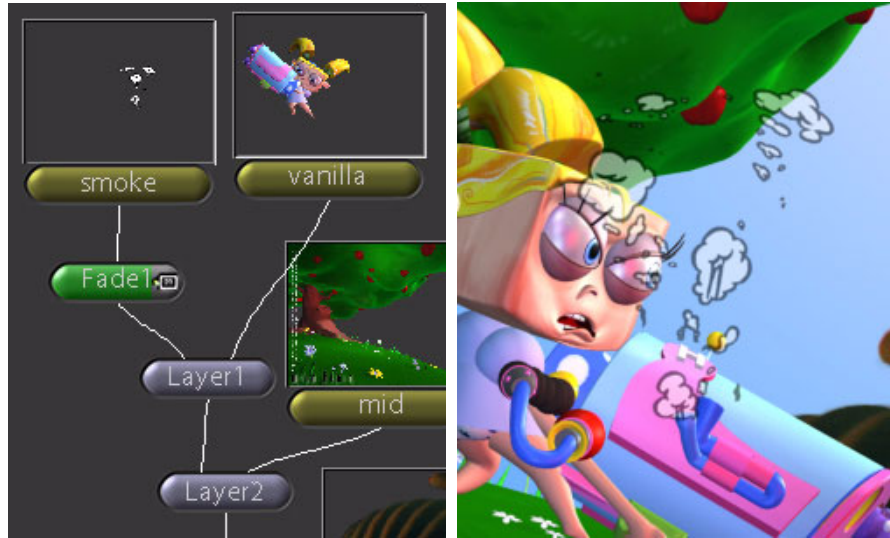
Fading an Element

The final step in this exercise (before rendering the animation) is to fade the smoke.

To apply a fade node to the smoke element:

- 1 Go to a frame that contains the smoke (after frame 7).
- 2 In the Node View, select the *smoke* element.
- 3 In the Color tab, click the *Fade* node.

- 4 Set the fade value to .5.



As demonstrated in this particular step, everything in Shake is done with a node.

Rendering a Sequence to Disk

If you are still awake, render and save the image sequence of this soul-shattering vision of humanity's existence.

To save the script:

- 1 Click the Save button (**Command+S** / **Ctrl+S**).

Because this is the first time you are saving the script, the Save Script window appears. Next time you click Save, the name you supplied is taken.

- 2 In the File Name text field, enter a name for the script, and click OK.

Note: To Save As, use **Command+Shift+S** / **Ctrl+Shift+S**, or choose File > Save As.

To render files to disk:

- 1 In the Globals tab, ensure that the Globals timeRange is set to 1-56 (click Auto).

- 2 Attach an Image-FileOut node to *Layer4*.

The Save Image or Sequence window appears.

- 3 In the Save Image or Sequence window, navigate to your desired directory.

Note: To create a new directory, click the Create Directory  button (a rather straightforward process).

- 4 Enter the image name into the File Name text field at the bottom of the window. As a courtesy to other software packages, include the following naming conventions:
 - The name of the file.
 - If you are not using the QuickTime format, supply the frame numbering symbol—either @ or #. A single @ is an unpadding number, that is, 1, 2, 3, 4. A single # is 4-place padding, that is, 0001, 0002, 0003. Either symbol used multiple times indicates that many places of padding, that is, @@@@ is 00001, 00002, 00003.
 - If you are using the QuickTime format, omit the frame numbering symbol and enter .mov at the end of the file. The format options appear in the *FileOut* parameters.
 - The file format extension, such as .cin, .sgi, .jpg, .iff, etc. For QuickTime, enter .mov.

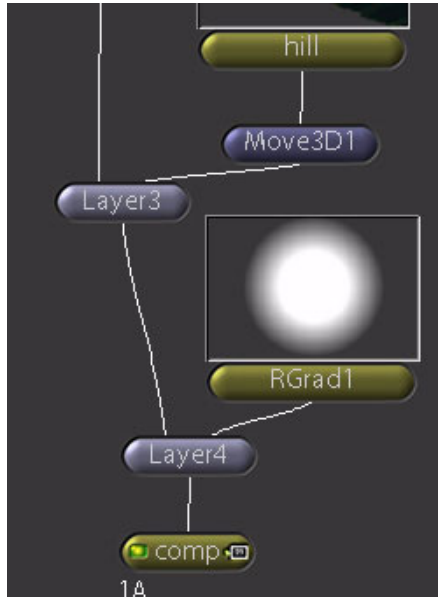
Example:

Enter *comp.#.sgi* as your filename. The images are written as *comp.0001.sgi*, *comp.0002.sgi*, and so on.



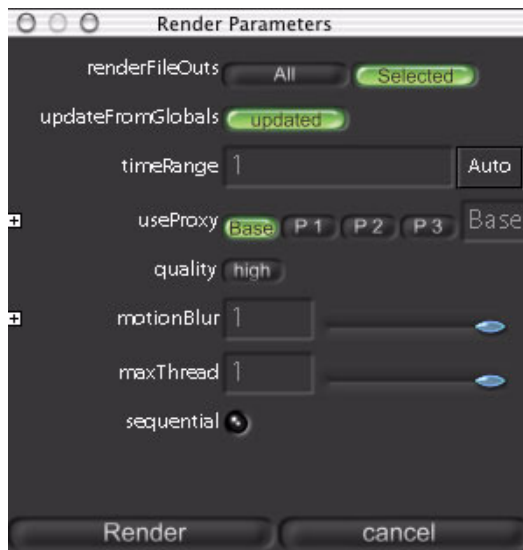
- 5 Once you have entered your filename settings in the Save Image or Sequence window, click OK.

A *FileOut* node that uses the given filename is added to the node tree.



- 6 Choose Render > Render FileOut Nodes.

Note: You can also right-click in the Node View and select Render > Render FileOut Nodes. The Render Parameters appear.



In the Render Parameters, you can enter a new time range, proxy scale, quality level, motion blur, and the number of CPUs to use for the render. To render all *FileOut* nodes, enable All in the renderFileOuts parameter. To render only active FileOuts, enable Selected. You can also select a *FileOut* node after the Render Parameters window is opened.

7 Click Render.

The images are rendered to disk, and a 320 x 243 snapshot view of the current frame is displayed. The snapshot is always 320 x 243, regardless of input resolution, and only shows the currently rendering frame.

Batch Rendering in a Shell

You can choose to render on the command line. Click the Save button at the top of the interface to save the script. If you have not yet saved the script, you are prompted for a script name, otherwise it writes over what you had before. (**Command+Shift+S** / **Ctrl+Shift+S** is Save As.)

Note: If you are working in a shell, you have usually set environment variables. For more information, see “Environment Variables for Shake” in Chapter 17 of the *Shake 3 Reference Guide*.

1 Save your script as *vanilla_test.sbk*.

2 Stow or quit Shake.

3 Open the Terminal and navigate to the directory where your script is saved.

4 Type:

```
sbake -exec vanilla_test.sbk -v
```

Note: In OS X, if you do not have environment variables set for Shake, you must type the full Shake path, for example:

```
/Applications/Shake3/sbake.app/Contents/MacOS/sbake -exec vanilla_test.sbk -v
```

Usually, you set your environment variables to use the *sbake* command with a lower-case “s.” For more information, see “Environment Variables for Shake” in Chapter 17 of the *Shake 3 Reference Guide*.

On a Linux or IRIX system, type the following (the Shake command begins with a lower-case “s”):

```
sbake -exec vanilla_test.sbk -v
```

The *-v* means “verbose,” so the render status is displayed. For more information on the command line, see Appendix B, “The Command-Line Manual,” in the *Shake 3 Reference Guide*.

5 To override the time range, use the *-t* option:

```
sbake -exec vanilla_test.sbk -v -t 10-20
```

or

```
sbake -exec vanilla_test.sbk -v -t 1-56x2
```

The above command renders every other frame.

Lesson Two: Intermediate Tutorial

This tutorial picks up at the spine-tinglingly riveting end of Lesson One, and discusses (OK, “discuss” means it blathers on for hours) color correction, premultiplication, optimization, motion blur, and the Time View.

Tutorial Summary

- Slipping Time Sync
- Optimization
- Motion Blur
- Color Correction With the *ColorCorrect* Node
- Animation Curves
- Color Correction Using Other Nodes
- Applying a Shadow Element

Slipping Time Sync

This section introduces you to the Time View, where clips are viewed in time. To start, you import some additional elements.

To import the elements:

- 1 Continue with the script from the Basic tutorial, or lazily drift up and click the Load button and load the script in *Shake3/doc/pix/vp/vanilla1.sbk*.

- 2 In the Image tab, click *FileIn* and read in *Shake3/doc/pix/vp/bg/cloud.sgi* and *vp/shadows/shadows.1-56#.iff*.



- 3 To view the composite so far (completed in the Basic tutorial), click on the left side of the *Layer4* node.
- 4 To go to frame 31, do one of the following:
 - Position the cursor at frame 31 in the time range display portion of the Time Bar (in the bottom-left of the interface) and click.
 - In the Current field, enter 31 and press **Enter**.
Note: You can use the **Left Arrow** and **Right Arrow** keys to move forward and backward one frame at a time.

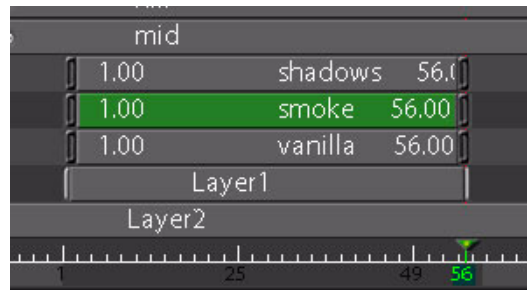
Notice that the *smoke* is out of sync with *vanilla*, and starts one frame too late. (Honestly, this wasn't set up to be cute, it was just one of those happy accidents...)



To sync the smoke clip with the animation:

- 1 In the Node View, select the *smoke* element, and click the Time View tab.

The following illustration shows that *shadows*, *smoke*, and *vanilla* are clips from frames 1 to 56. The *mid* image (single frame), *Layer1*, and *Layer2* (composited with *mid*) have infinite length.

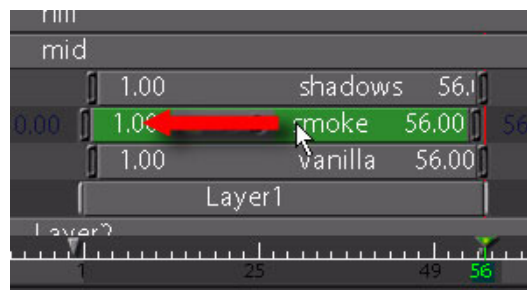


You can select nodes in the Node View or the Time View. You can view nodes, load parameters, or ignore nodes by clicking on the buttons on each clip in the Time View. Activate the Select Group toggle to load only the active nodes in the Time View.

- To scale the Time View, press **Command / Ctrl** and the middle-mouse button and drag left and right.
- To pan the Time View, press the middle-mouse button and drag.

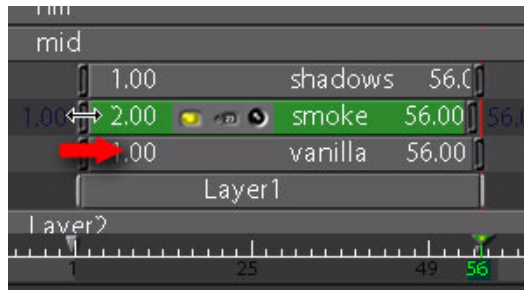
2 Click and hold on the middle of the *smoke* clip bar, and drag it one frame to the left.

The clip is slipped back one frame. The numbers inside the clip represent the clip range that is read from disk. When the clip is dragged, the numbers outside of the clip are the frames (exclusive) that the clip runs between in the scene.



The *smoke* clip now starts one frame early. Normally, this is not a problem, as the first frame of smoke is black. As a clean-up step, reset the clip to start at frame 1.

- 3 Drag the left end of the smoke clip to the right so that the clip starts at frame 1 (the outside number, displayed in blue, reads 1.00 and the clip number reads 2.00) and the clip aligns with the other clips. This indicates that only frames 2 to 56 are read from disk.



- 4 Go to frame 30 and verify that *smoke* is properly synced.

Out of Sync



In Sync



Optimization

In the Node View, position the cursor over the *shadows* node and look at the help window (in the lower right of the interface). Notice that the channel type is BW (Black and White)—there is only one channel in the image. This is significant in the later “Applying Vanilla’s Shadow” section, but for now, note that because it is only one channel, it uses less disk space and loads faster than a 3-channel RGB image. To optimize your projects, you can save your black-and-white images as 1-channel images.

To save a black-and-white image as a 1-channel image:

- Apply a Color–*Monochrome* node to the *FileIn*.
- Apply a *FileOut* as a new image name.

You do not need to save the tutorial images as 1-channel images. (It has already been done for you—fancy that.)

Note: Many compositing and paint packages do not support 1-channel images, so test your pipeline before throwing yourself into battle.

You can also easily make monochrome images with the command line:

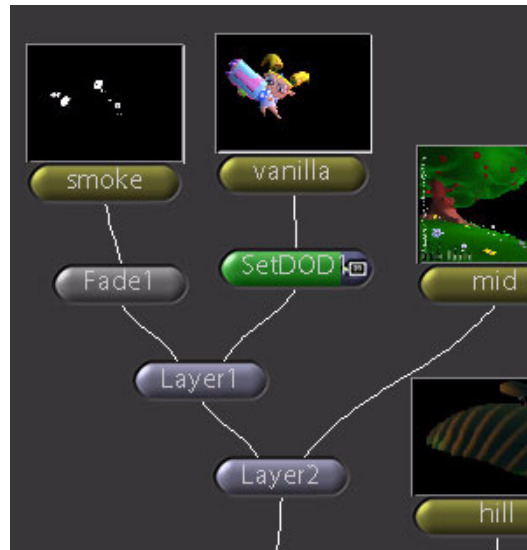
```
shake myRGBImage.#.iff -mono -fo myBWImage.#.iff -t startFrame-endFrame -v
```

For more information on the command line, see Appendix B, “The Command-Line Manual,” in the *Shake 3 Reference Guide*.

A powerful optimization step to take early in the compositing process is to use the glorious, enemy-crushing trick of designating a Domain of Definition (DOD). The DOD reduces rendering time, I/O activity, and memory use. The DOD is a rectangular region around an element that defines the important part of the image, that is, everything that is not pure black on the RGBA channels. For example, the *vanilla* image is mostly black—you are only interested in Vanilla herself. Therefore, limit the DOD to indicate the fun bits of the image.

To set the DOD:

- 1 In the Node View, select the *vanilla* node.
- 2 Insert a Transform–*SetDOD* node.



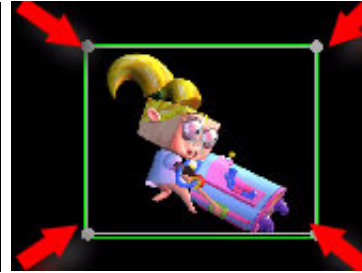
- 3 Make sure the Onscreen Controls are enabled .
- 4 Drag the DOD corners or edges in the Viewer to frame Vanilla.

This tells the software to only be concerned with this smaller portion of the image. The rest of the image is not loaded in. This reduces memory usage, I/O activity, and processing time, both upstream and downstream of the SetDOD. Isn't that swell?



Vanilla



Vanilla With DOD

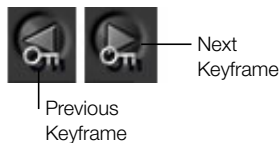


Because the clip is animated, you must animate the DOD as well.

- 5 In the Viewer toolbar, enable Autokey  in the Viewer. When enabled, it sets a keyframe when you modify the DOD. If you make a mistake, click Delete Keyframe  to delete the keyframe.
- 6 Go through the 56 frames, placing keyframes approximately every 7 frames to follow the animation. The following illustration has keyframes at frames 8, 15, 21, 28, 47, 51, 53, and 56. As you place keyframes, notches appear in the Time Bar that indicate a keyframe.



To quickly jump to keyframes, use the Next Keyframe or Previous Keyframe buttons on the Time Bar, or press the **Up Arrow** and **Down Arrow** on the keyboard.



If you want to practice, the *smoke* and *shadows* nodes are good candidates for *SetDOD* nodes. This step is not necessary for the tutorial, but is recommended for production. You can frequently drop rendering time by 50 percent with *SetDOD*.



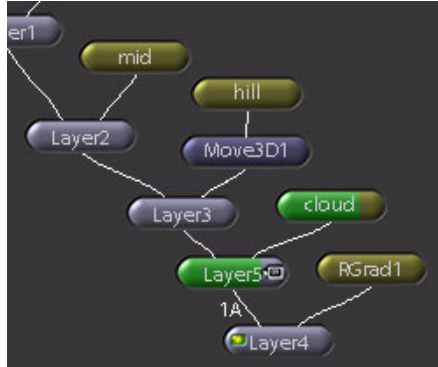
Motion Blur

To examine motion blur, you need some motion. In the real Vanilla Pudding shot, a dark cloud shoots quickly in as Vanilla amps up her anger. Wild Brain has a cool animation for the effect, but then you would not be able to examine motion blur, and would have to load another 10 MB of data. To compensate in this lesson, animate the position of the *cloud.sgi* still image.

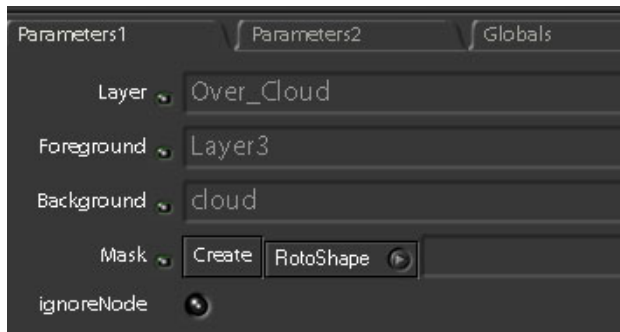
To animate the cloud image:

- 1 In the Node View, double-click the *Layer3* node and attach a *Layer-Layer* node. The fifth layer node is inserted between the *Layer3* and the *Layer4* nodes.

- 2 Connect the *cloud* node to the second input of *Layer5*.



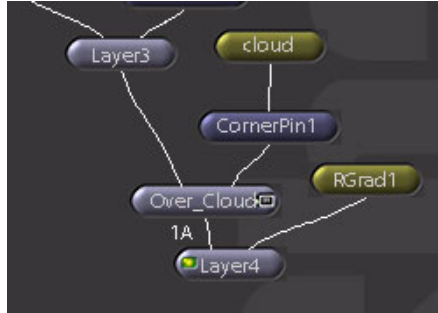
- 3 Because the *Layer* nodes are beginning to stack up, rename *Layer5* to *Over_Cloud*.
 - a Click the right side of the *Layer5* node to load its parameters.
 - b Click in the nodeName text field, type “Over_Cloud,” and press **Enter**.




Note: When naming nodes, do not use spaces, funny characters, or node types, such as “Blur,” since there is already a *Blur* node in the Shake toolset.

Next, move the cloud image. In the first tutorial, a *Move3D* node was used to pan the hill. To help modify the shape of the cloud and add a bit of perspective, use a *CornerPin* node.

- 4 Select the *cloud* node and insert a Transform–*CornerPin*.



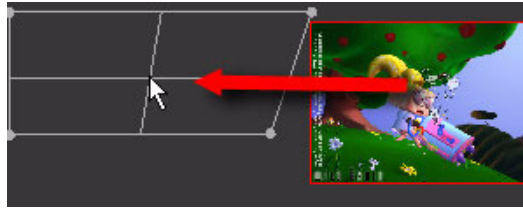
Vanilla's expression changes from curiosity to anger from frames 30 to 37. Move the dark clouds to support her animation.

- 5 Go to frame 37.
- 6 Ensure Autokey  is enabled.
- 7 Using the *CornerPin* node onscreen controls and the following illustration as a guide, position the corners of the *cloud* image. Since the cloud image is 500 x 400 pixels, the corners do not match the borders of the full frame. Pinch in the bottom of the image for some perspective.



- 8 Go to frame 30.
- 9 Using the following illustration as a guide, click the center crossbar (to move all four corners at once) and drag to the left.

Note: You can also grab an edge to move two corners at a time.



- 10 In the Time Bar, move the current frame marker along frames 30 to 37 to test the animation. The cloud is animated over 7 frames.

The Transform nodes *Move2D*, *Move3D*, *CornerPin*, *CameraShake*, *Pan*, *Rotate*, *Scale*, *Stabilize*, and *MatchMove* have motion blur parameters. There are three motion blur parameters: *motionBlur*, *shutterTiming*, and *shutterOffset*. The *shutterTiming* and *shutterOffset* parameters are located in the *motionBlur* subtree.



The motion blur in Shake is mighty swank—rather than render several in-between images and average the images together, Shake tracks each pixel through the transformation.

To activate the motion blur:

- 1 In the *CornerPin1* node parameters, set the *motionBlur* to 1.

motionBlur = 0

motionBlur = 1



The motionBlur parameter controls the quality level. When set to 0, the blur is turned off. When set to 1, the blur is high quality. For an approximation and to decrease your render time, lower the motionBlur value to .5 (for acceptable), or .1 (for test level).

- 2 Set motionBlur to .25.

motionBlur = .5



motionBlur = .1



The shutterTiming parameter specifies the fraction of the frame exposed for the motion blur. For example, the default .5 setting indicates that only half of the frame of movement is exposed. This is the default since a film camera has a shutter exposure of around 178 degrees out of 360, so it is rounded off to .5. (You can also enter 178/360 to be exact, but nobody likes a smart aleck.) If shutterTiming is set to 0, motion blur is disabled. If set to 1, the entire frame is calculated. It looks good, but is technically inaccurate. You can also set this number beyond 1 to calculate later movement into the current frame. Therefore, to attempt to match a video camera with a shutter setting of 1/300, enter 1/300 in the shutterTiming parameter, and it resolves the setting to .0033, or not much blur at all.

- 3 To test the effect of changing the shutterTiming value, set shutterTiming to 1.5 (in the motionBlur parameters subtree).

- 4 Set shutterTiming to .5.

shutterTiming = 1.5



shutterTiming = .5



The shutterOffset parameter controls the beginning point in time, relative to the start of the frame, that the motion is calculated. By default, shutterOffset calculates the motion from the start of the frame. When set to -.5, it calculates the motion from the previous half frame of movement. This can be useful when adjusting motion blur for *RotoShapes*.

- 5 To test the effect of changing the shutterOffset value, set shutterOffset to -.5 (in the motionBlur parameters subtree).
- 6 Set shutterOffset to 0.

shutterOffset = -.5



shutterOffset = 0



Each transform node with motion blur has settings that can be individually tuned. To turn off *all* motion blur settings, click the Globals tab and set the motionBlur parameters to 0. The first two parameters are multipliers of the node motion blur settings. Entering 0 in the Global motionBlur internally multiplies all other motionBlur settings by 0, and disables all motion blur. To return to the node's setting, set motionBlur to 1. Typically, you tune your motion blur and then set your Global motionBlur to 0 as you finish the rest of your script. When you are ready to render, reset your Global motionBlur value to 1.

- 7 To ensure fast evaluation of your script, click the Globals tab and set the motionBlur to 0.

Color Correction With the ColorCorrect Node

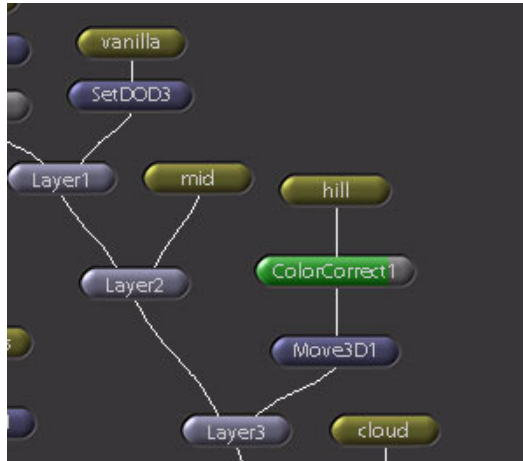
Now that you have added the cloud, you can begin to consider the coloring. Vanilla Pudding has strong saturated colors, so you do not want to modify her. However, you can boost the color of the hill in the background to suggest atmosphere (the meteorological phenomenon, not the romantic staging practice). You can then animate the color boost down when the cloud passes over.

Shake has many color correction nodes that are based mainly on its command-line heritage. Most of the color corrections are single-function nodes that apply a single mathematical operation—add, multiply, gamma function of color, etc. In the interface, you can also use a master color correct node that combines many of these nodes. The node is deviously named *ColorCorrect*.

To color correct the hill element:

- 1 Go to frame 30.
- 2 Click the left side of the *Layer4* node to view the composite.
- 3 In the Node View, select the *hill* node and insert a *Color–ColorCorrect* node.

Typically, the general rule is to perform color correction before you apply transformations.



Although there are many parameters in the *ColorCorrect* node, only the Master Add parameters are used in this lesson. With the *ColorCorrect* node, you can tune your RGB channels globally, or just in the dark areas, midtones, or highlights, do manual curve adjustment, color replacement, channel inversion, reordering, and hue shifting.

- 4 In the *ColorCorrect* parameters, click the Master Controls subtab (open by default).
- 5 In the Add parameter row, press **O** (offset), and hold the left-mouse button down and drag to the right until the value is approximately .3.
- 6 Press **B** (blue) and drag slightly to the right to add some blue.

Note: You can also click the Add color picker and select a color from the Color Wheel, or right-click on the picker box and select a color from the Pop-Up Color Palette.



Master Add = 0, 0, 0

Master Add = .29, .28, .34



The sky is incorrectly boosted in brightness with the hill image. This occurs because you are color correcting a premultiplied element from a CG render. When you use the Over operation mode in a *Layer* node, you declare that the black areas in the RGB channels match the black areas in the mask channel. When the Add parameter is modified, the black levels are boosted, and so do not match the black areas of the alpha channel. This occurs when a premultiplied element is color corrected. This is bad. The *preMultiply* parameter in the *ColorCorrect* node allows you to easily correct this premultiplication problem with a button click or two.

- 7 In the *ColorCorrect1* node parameters, click the Misc subtab.

- 8 In the preMultiplied parameter, click “yes.”



preMultiplied - Off




preMultiplied - On



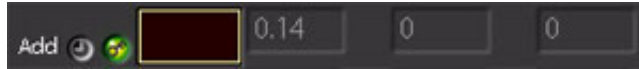
Only the *hill* element is color corrected (and not the sky behind it). This is good.

Note: Shake exposes control of premultiplication to ensure the user has control over premultiplication settings. Because premultiplication is at the heart of compositing, regardless of the software you use, you are strongly encouraged to read “About Premultiplication and Compositing” in Chapter 9, “Color Correction: Part II,” of the *Shake 3 Reference Guide*.

To animate the hill color correction:

- 1 Go to frame 31 (the cloud starts to pass overhead).
In the Parameters tab, there is a small Autokey button  next to every parameter. When enabled, you can animate parameters.
- 2 Click the Master Controls tab, and enable Autokey next to the Master Add color picker.
A keyframe is created at frame 31. When Autokey is enabled and you change a value, a keyframe is created.
- 3 Go to frame 37.

- 4 Lower the Add values to 0, and add a small amount of red.



Frame 31



Frame 37



Animation Curves

This section introduces editing animation curves. The following steps are not necessary for the tutorial, but demonstrate how to create an ease-in/ease-out on your curves.

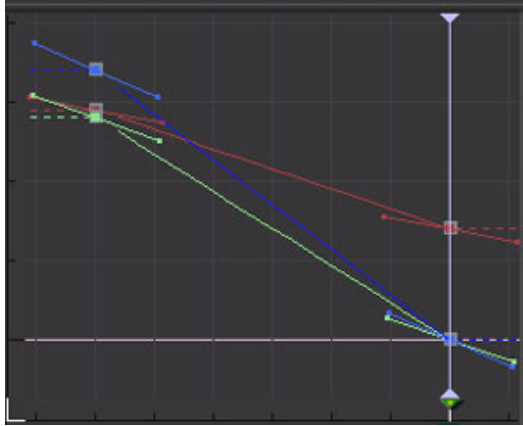
Loading and editing parameters in the Curve Editor:

- In the Add parameter row, click the button that looks like a small clock . (Clock equals Time equals Curves. Get it?)

A red checkmark appears to indicate the parameter is loaded in the Curve Editor. This is automatically activated when the Autokey button is enabled.

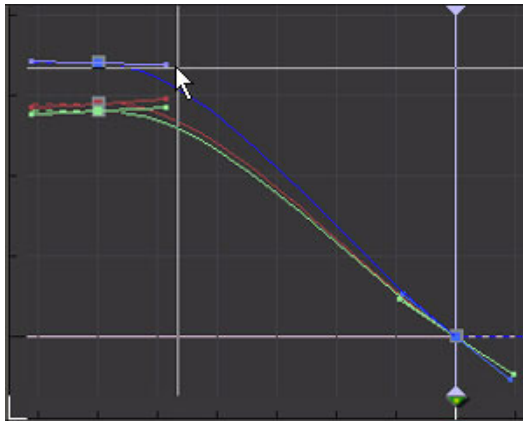


- In the Tool Tab, click the Curve Editor tab to view the animation curves.



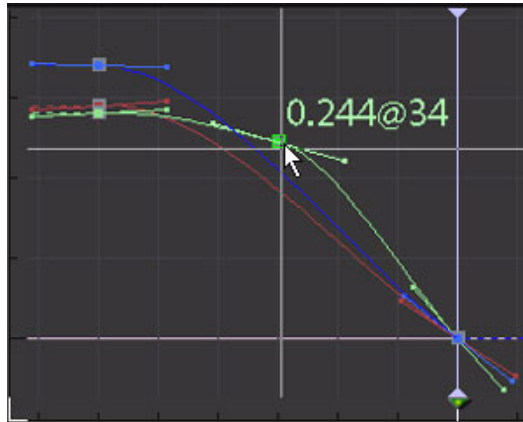
Note: To expand the Curve Editor window, position the cursor in the Curve Editor and press the **Space bar**. Press the **Space bar** again to reset the Curve Editor window.

- To adjust the tangents, click and drag on a tangent end.
The flatter the control point curve, the greater the ease-in or ease-out.



- To insert a keyframe, **Shift**-click on a segment.

Note: You can also enter a value at the desired frame in the Parameters tab to create a keyframe.



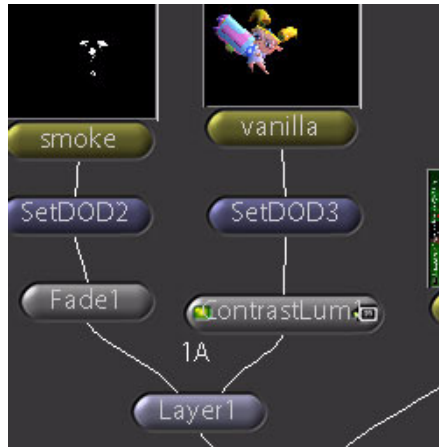
- To move a keyframe, click the keyframe and drag.
- To remove a keyframe, select the keyframe and press **Delete** (near the **Page Up/Down** buttons) / **Backspace**.
- To break the tangents of a curve, **Ctrl**-click a tangent end.
- To join broken tangents, **Shift**-click a tangent end.

Color Correction Using Other Nodes

This section discusses a practical solution to premultiplication problems and the concatenation of color corrections. To discuss color corrections without the *ColorCorrect* node, first do the following small test (that has nothing to do with the final composite).

To test for premultiplication:

- 1 In the Node View, select the *SetDOD* node that is attached to the *vanilla* node and insert a *Color-ContrastLum* node.



- 2 In the *ContrastLum1* parameters, set value to .4.

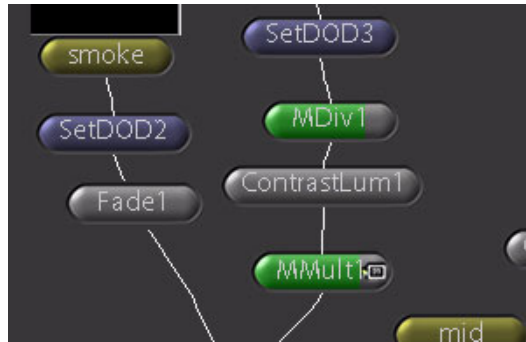
Noise appears along the edge of Vanilla's hair. (Press + or - by the **Delete / Backspace** key to quickly zoom in and out of the Viewer.)



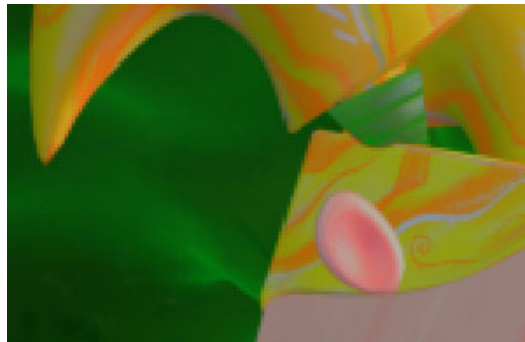
This problem occurs because, as with the hill image in the “Color Correction With the ColorCorrect Node” section above, you have color corrected a premultiplied element and revealed artifacts.

The quick solution is to unpremultiply the element before color correcting, and then premultiply the element again before you composite. This is done by bracketing the *ContrastLum* node between a *Color-MDiv* (“Mask Divide”) node and a *Color-MMult* (“Mask Multiply”) node. Use these nodes whenever you modify a premultiplied element (the black areas of the RGB equal the black areas of the alpha in a premultiplied image).

- 3 In the Node View, select the *SetDOD* node above *ContrastLum1* node and insert a *Color-MDiv* node.
- 4 Select *ContrastLum1* node and insert a *Color-MMult*. (Be sure you insert the *MMult* node, not a *Mult* node.)



The edges are clean.



To test the premultiplication solution:

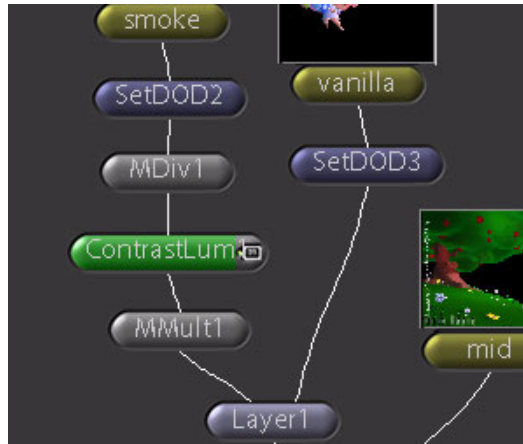
- 1 In the Node View, press **Shift** and select the *MDiv1* and *MMult1* nodes.
- 2 Press **I** to ignore the *MDiv1* and *MMult1* nodes.
A red line appears on the ignored nodes.
- 3 Press **I** again to activate the *MDiv1* and *MMult1* nodes.

For a discussion on how the above process solves the oh-so-simple-solution-and-oh-so-misunderstood premultiplication problem, see the mathematical elaboration in “About Premultiplication and Compositing” in Chapter 9, “Color Correction: Part II,” of the *Shake 3 Reference Guide*.

The *ContrastLum* node on Vanilla is used as an example. To put this into practice, color correct the *smoke* element so it is not so bright and has some transparency.

- 4 **Shift**-select the *MDiv1*, *ContrastLum1*, and *MMult1* nodes.
- 5 Press **E** to extract the nodes from the tree.

Note: To remember hot keys, right-click in the area—a contextual menu that lists all commands and their associated hot keys appears.
- 6 Connect the *SetDOD* node (that is attached the *smoke* node) to the *MDiv1* node, and reconnect *MMult1* to the *Layer1* node.



The next interesting aspect of color nodes is that the nodes concatenate. What in blazes does concatenate mean? Concatenation collapses multiple mathematical equations into one operation. Concatenation gives two primary benefits: Greater color integrity and faster processing. For example, if you apply a *Brightness* node, which multiplies color, with a value of 5, and then apply a second *Brightness* node of .2, they cancel each other out, since $RGB * 5 * .2 = RGB * 1$. Without concatenation, you end up with most of your image crunched down to a 20-percent value. Not all color nodes concatenate—only the nodes that have a small yellow “C” on the node icon in the Tool Tab. The nodes with a red “C” (*AdjustHSV* and *LookupHSV*) only concatenate with each other. So, for example, with the six nodes in the following illustration, *Brightness*, *Clamp*, *Compress*, and *ContrastRGB* can be placed in any order as many times as necessary and they internally resolve into one operation. *AdjustHSV* can only be lined up with itself or a *LookupHSV*.



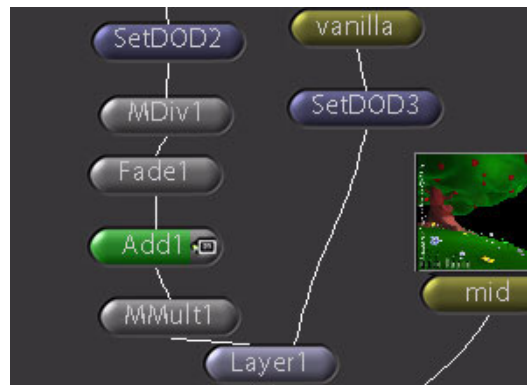
Note: When working in float space, the concatenation restrictions disappear, since all nodes effectively concatenate.

To fade the smoke element:

- 1 In the Node View, select only the *ContrastLum1* node.
- 2 In the Color tab, **Ctrl**-click *Fade*.

The *ContrastLum* node is replaced with a *Fade* node.

- 3 In the *Fade* parameters, set the value to approximately .6 to add transparency.
- 4 Insert a Color-*Add* node after *Fade1*.



Next, use the *Add* node to boost the black levels on the outline of the smoke.

- 5 Press **O** (Offset) and drag to the right over the *Add1* node's Color Picker to set the levels up to approximately .3.
- 6 Tint the smoke slightly blue—press **B** and drag to the right to boost the blue channel a bit.

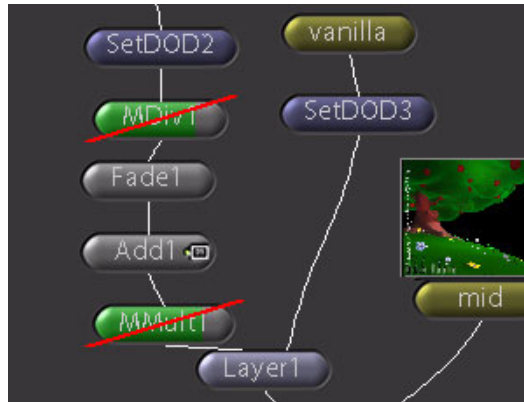


Because the *Fade* and *Add* nodes concatenate, the nodes are calculated in one pass rather than two.

Note: Inserting a side-input mask or non-concatenating node breaks concatenation, and results in inefficiencies and possible loss of image integrity.

Now, what about those pesky *MDiv* and *MMult* nodes? Are they really necessary?

- 7 For the answer to the above question, select the *MDiv* and *MMult* nodes and press **I** to ignore the nodes.



Because an *Add* node is used, the blacks are boosted, and the resulting image appears too bright.



- 8 Press **I** again to activate the *MDiv* and *MMult* nodes.

Note: The *Fade* node does not require an *MDiv*/*MMult* coupling, because it affects the RGB and alpha channels equally. However, if placed after *MMult1* or before *MDiv1*, it breaks concatenation. This example opts for efficiency.

Note: The more efficient way to compress your black and white levels is to use the Color–Compress node. It sets your new low and high colors. The *Fade/Add* combination is used to demonstrate concatenation.

Applying a Shadow Element

The last step discusses two different strategies for applying the *shadow* element to the *mid* plate. The simple approach is discussed first.

Shadow Method 1

The *shadows* element is a single channel black-and-white image with no alpha channel. Subtracting the *shadows* element from the *mid* element affects the RGB channels of *mid* but not the alpha. This is good.

To subtract the shadow element:

- 1 In the Node View, select the *mid* node and attach another Layer–Layer node.
- 2 In the *Layer* node parameters, name the *Layer* node “ShadowSubtract” (or some other clever name).

- 3 Connect the *SetDOD* node from *shadows* (if you created one earlier in the “Optimization” section for practice, otherwise connect the *shadows* element directly) to the second input of the *ShadowSubtract* (new *Layer*) node.



- 4 In the *ShadowSubtract* node parameters, set the operation of *ShadowSubtract* to *ISub*. (Select *ISub* from the “operation” pop-up menu.)

ISub means “Image Subtract.” *IAdd*, *IMult*, *IDiv*, *ISub*, and *ISubA* are also available as math operators. *ISubA* takes the absolute value of the subtraction, making it a good node for finding the difference between two images (but has nothing to do with this composite).



A percentage slider appears when the operation is switched to *ISub*.

- Using the slider, set the percentage to approximately 20 or 30 percent.



- To save the script, press **Command+Shift+S** / **Ctrl+Shift+S** (Save As).
The Save script window appears.
- Name the script *vanilla2.sbk* (or something artistic that expresses the inner magical you).

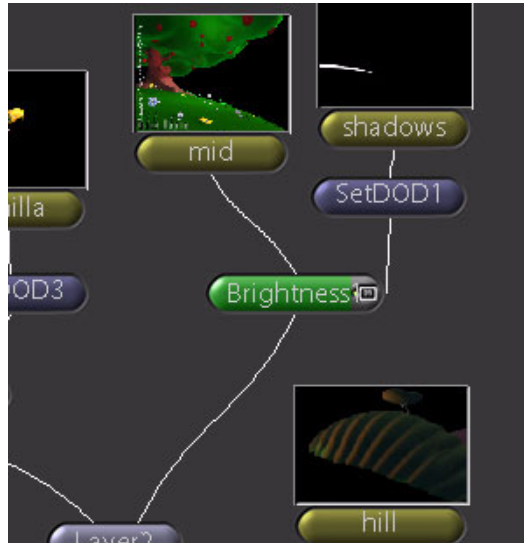
Shadow Method 2

The next way to apply the shadow is as a mask to a color correction. Replace the *ShadowSubtract* node with a *Brightness* node.

To create the shadow as a color-correction mask:

- In the Node View, select *ShadowSubtract*.
- In the Color tab, **Ctrl-click** *Brightness*.
The *ShadowSubtract* node is replaced with a *Brightness* node.

- 3 Connect the output of the *SetDOD1* node (that is attached to *shadows*) and connect it to the side input of the *Brightness1* node.



Note: Although all nodes have a side mask input, you only get practical results from one-input nodes such as *Add*, not multiple-input nodes such as *Layer*. The internal logic is the opposite of what you expect for multiple-input nodes.

- 4 In the *Brightness* parameters, set the value to approximately .3.

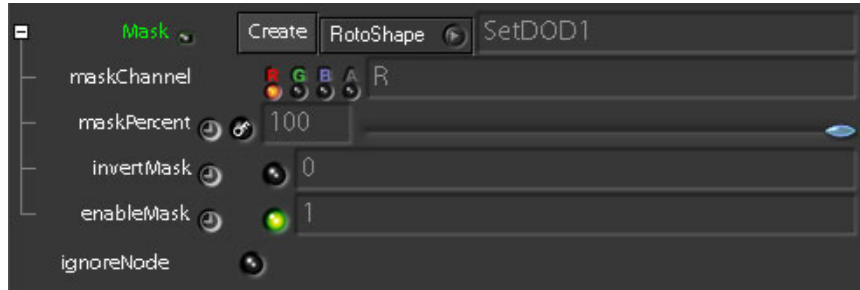
You are no doubt chagrined and humiliated to discover absolutely nothing happens. Because the mask takes the alpha channel of the mask input (*SetDOD1* in this example) as the masking channel, nothing happens. If you recall, *shadows* only has a single black-and-white channel. Therefore, the mask is considered to be black, so there is no influence as a mask.

- 5 To set a mask channel other than alpha, load the parameters of *Brightness1*. (Click on the right side of the node.)

In the upper portion of the *Brightness* parameter tab is a parameter called Mask. If a mask is attached, there is a subtree plus sign.

- 6 Click the Mask subtree plus sign to reveal parameters that pertain to the mask.

- 7 Set the maskChannel parameter to R, G, or B. Since *SetDOD1* is a black-and-white channel image, any channel (except alpha) works.



The shadow is created with the shadow element used as a mask to a color correction.

Use the side mask input in the following circumstances:

- The masked node is not a *Layer* node or any other node reading in more than one input. The logic is completely backward from what you want.
- You use the mask just once in a chain.
- You are not concerned about breaking color or transform concatenation.

If you use the same mask several times on connected nodes, you:

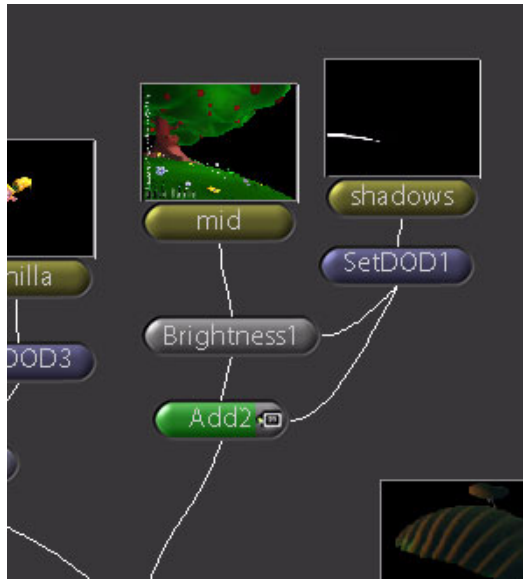
- Slow down your render.
- Break possible concatenation with transform or color nodes.
- Corrupt the boundary between the masked and the non-masked area.

(Optional) To add blue to the shadow area with the Add node:

- 1 In the Node View, select the *Brightness1* node.
- 2 Insert a *Color-Add* node.

- 3 Connect the *SetDOD1* as the mask input. Remember to open the Mask subtree and specify the proper channel.

This is an example. Don't do this. This is naughty.



The brightened area is mixed with the non-brightened area, and then mixed with the brightened blue. Two mixing operations are calculated. *Add* and *Brightness* usually concatenate, but not in this case because it is broken by the mask inputs.

The workaround is to use the *Layer-KeyMix* node. This mixes two images through a third mask.

To use a *Layer-KeyMix* node:

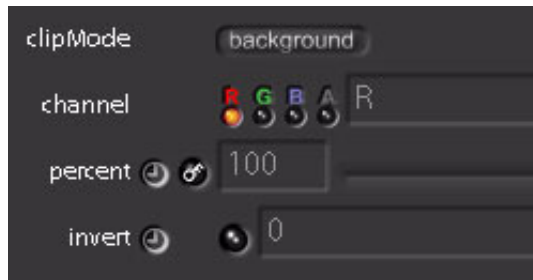
- 1 In the Node View, select the *mid* node.
- 2 In the Layer tab, **Shift-click** *KeyMix*. This creates a new branch instead of inserting *KeyMix1* between the *mid* and *Brightness1* nodes.
- 3 Connect the *Add2* node to the second input of the *KeyMix1* node, and connect the *SetDOD1* node to the third input of *KeyMix1*—not the side mask input.

- 4 To disconnect the noodles to the mask inputs of the *Add2* and *Brightness1* nodes, **Ctrl**-click on the input noodles.

Instead do this. This will bring glory and fame.



Like the mask inputs, you must specify the channel of the third input you want to use as a mask in the *KeyMix* parameters. You can also invert the channel.



You have now created the shadow with a different technique. You can pile as many operations as you want on the *Brightness1–Add1* chain and have them efficiently mix into just the shadow area.

To wrap this up, the two main compositing nodes are *Over* (or *Layer* with its operation parameter set to *Over*) and *KeyMix*.

- Use *Over* when the first input has the mask you want to use. The image can be pre-multiplied or unpre-multiplied. If unpre-multiplied, you must enable *preMultiply* in the *Over* parameters.

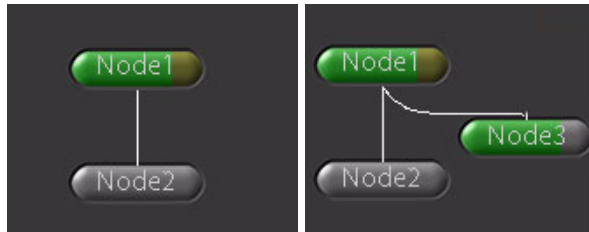
- Use *KeyMix* to mix two images through a mask, or when you are compositing foreground over background with a third image as the mask. The foreground must be unpremultiplied.

Inserting Nodes Into a Tree

To insert a node between two nodes, pick the parent node and click on the new node from the Tool Tab.



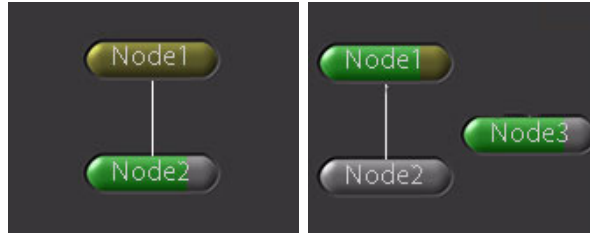
To create a new branch, select the parent, and **Shift**-click on the new node.



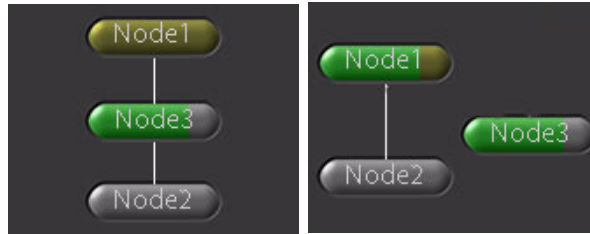
To replace a node, select the node to replace, and **Ctrl**-click on the new node.



To create a floating node, **Ctrl+Shift**-click on the node you want.



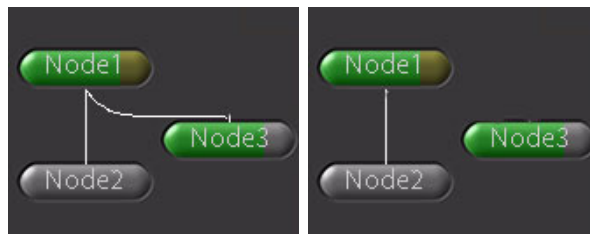
To extract a node from a tree, select the node and press **E** for Extract.



To delete nodes, select the nodes and press **Del / Delete**.



To detach a noodle, **Ctrl**-click on the noodle or put the cursor over the noodle (it turns red) and press **Del / Delete**.



Lesson Three: Depth Compositing

A Z channel is the distance of a pixel from the image plane (the camera). This information usually comes from a 3D render, and precisely indicates the distance of a pixel from the 3D camera. Pixels with higher Z values are further away from the camera than pixels with lower Z values. Unlike normal color values that range from 0 to 1 (in 8 and 16 bits), Z values can be any number. This tutorial discusses the use of the Z channel in Shake, as well as other fundamental compositing tools, including the Pixel Analyzer node and color correction.

Tutorial Summary

- The City Composite
- A Few Viewer Tips
- Viewing the Z Channel
- Preparing the Z Channel
- Creating the ZCompose
- Matching the Blimp and Background Colors
- Adding Fog

The City Composite

Most composite nodes depend on the RGB and alpha channels to calculate the final composite. Depth compositing uses a fifth channel, the Z channel, that orders pixels according to a certain depth from the pixel plane. Generally, Z channels are generated with a 3D renderer. Z channels are simple to generate in a 3D render, since every pixel has a certain distance from the camera. Z compositing takes the pixel with the lower Z value when it compares two images. A Z channel can be rendered into the same file as the RGBA, or saved into a separate file and later loaded with a *Copy* node using Z as your channel. In this tutorial, use both techniques to composite 3D-rendered blimps into a (really bad) scan of New York City.

The following two images each contain Z information. The blimps have a rendered Z channel; the city has a hand-painted channel. By using the Z, you can mix the blimps among the buildings.

RGB Images



Depth Images



Blimp Mask and Final Composite



Why use *ZCompose*? With nearly one hundred ships in the air, you have to recreate the cityscape in 3D and create a 3D mask in the render. The most practical solution in 2D is to fake the Z with a hand-painted mask of the New York City image. By using Z, the dirigibles appear in front of and behind the buildings, as shown in the following close-up image.



To start the Z composite, read in the tutorial images using one of the following:

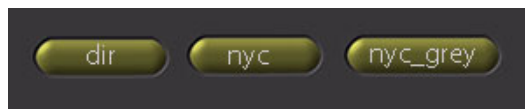
- Start Shake, and create an Image-FileIn node. In the Browser, navigate to *doc/pix/nyc*. **Shift**-click *dir.iff*, *nyc.jpg*, and *nyc_grey.jpg*, and click OK.

Note: **Shift**-click allows you to select multiple files within the same directory. You can also drag-select.

- Be a geek and use the Terminal to start Shake. Navigate to *doc/pix/nyc*, and type:
`shake dir.iff nyc.jpg nyc_grey.jpg -gui`

Note: This assumes you have Shake in your Terminal path.

You have a simple (but oddly soothing) tree that looks like the following:



A Few Viewer Tips

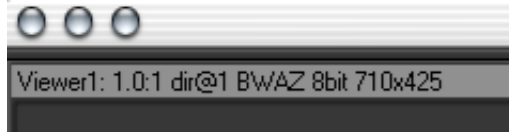
This section provides some helpful tips for using the Viewer.

First, a bit of image framing:

- 1 In the Node View, click the left side of the *dir* node to view the *dir* image.
- 2 In the Viewer, press the **F** key to fit the image to the Viewer. The “fit” icon is illuminated.

Note: “Fitting” the image to the Viewer may result in a non-integer zoom, and cause visual artifacts. This is especially noticeable with interlaced footage.

- 3 Press **Home** to reset the image to a 1:1 view ratio.
Any visual artifacts caused by the non-integer zoom are removed.
- 4 Press **+ / -** (by the **Delete / Backspace** key) to zoom the image. The location of the cursor determines the point of zoom.
Shift+F ensures that the Viewer title bar is visible. The Viewer title bar displays information about the image. For example, the *dir* image is a BWAZ (black-and-white image, with alpha and Z channels) 8-bit image.



Now you are probably thinking, “Great, I just press Z and see the Z channel.” Well, that would be dandy and obvious, but sadly it isn’t so. Shake does not know what to show you that accurately displays your image sequence, as your Z values may zoom toward or away from the camera over the course of the sequence. For example, a plane could be flying toward the camera, so Z values change every frame.

Viewing the Z Channel

To view the Z channel of an image:

- 1 Click and hold the Viewer Scripts button, and select the Z View button.



The Z values for your image are determined and displayed. This is an old Alias-rendered image, so the actual Z values go from 0 to approximately 270.



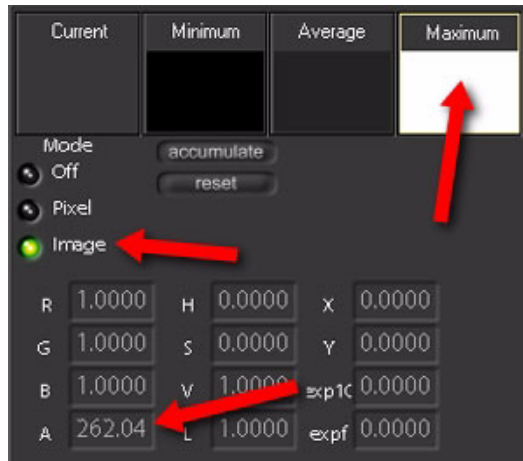
- 2 To see the numerical Z values, right-click the Z View button and select Load Viewer Script Controls into Parameter2 Tab.

- 3 In the Parameters2 tab, set the floatZinA parameter to Original.

The real Z values are placed into the alpha channel for analysis purposes. Scrub the cursor on the Viewer to print the distance on the top of the Viewer, except it's in the alpha channel position. (Wasn't that easy? How do you say "hack" in your language?)

- 4 Use the Pixel Analyzer tab when the Z View Script is applied to get the min/max values. In the Pixel Analyzer tab, set Mode to Image and click Maximum.

The maximum value is printed in the alpha slot. This only works when the Z View floatZinA is set to Original or Distance.



Not the most elegant method, but it works. Here, Original and Distance return the same value. However, if you use a 3ds max or a Maya-rendered image, you get negative values and $-1/\text{distance}$ values respectively, which are not obvious. Setting the Z View ViewerScript to Distance returns the actual distance from the camera.

Note: The Z View is non-intuitive in that it displays low values (objects close to the camera) as white. Objects which have greater Z values (objects far from the camera) are given darker values. This is the opposite of how you associate high numbers and image values, but it allows Shake to display distant objects as fading into the infinite distance properly.

- 5 Click the Z View button to disable the Z View ViewerScript.

Preparing the Z Channel

You must prepare the false (hand-painted) Z channel for the New York City image to be copied over as a proper Z channel. Whereas the dirigible element has a specific Z channel with ranges from 0 to approximately 262, in this case you must copy the painted black-and-white image as the simulated Z channel. The only difference is that the values of this channel, since it is 8-bit, range from 0 to 1.

Apply an *Invert* node and a *Reorder* node to the NYC depth image:

- 1 In the Node View, select the *nyc_grey* node.
- 2 Apply a *Color-Invert* node.
- 3 With the *Invert* node selected, add a *Color-Reorder* node.



The *Reorder* node is used to copy the red channel of the painted image into the Z channel.

- 4 In the *Reorder* parameters, enter *rgbar* in the channels text field (in the lower portion of the parameters). This string specifies that red goes to the red channel, green to the green channel, and so on. The fifth value, Z, also takes the red channel.

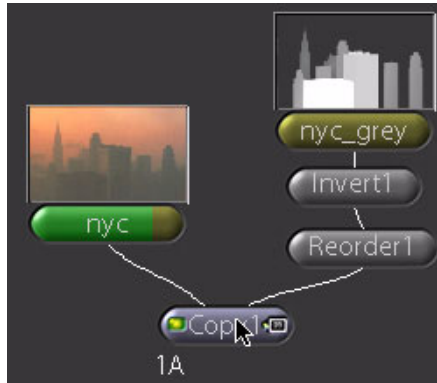
The city Z image fades to black because there are low values in the distance. Either the *nyc_grey* image or the *dir* image must be inverted, because in one image white represents the distance, and in the other, it is black. You cannot flip the blimps (*dir* image) because its Z range goes beyond 1, and *Invert* only properly inverts the Z channel if it is normalized (between 0 and 1). Therefore, the *Invert* node is placed on the city Z image (*nyc_grey*). Yup, this is confusing.

You have now created a snazzy Z channel. Next, use a *Layer-Copy* node to copy the Z channel to the NYC image.

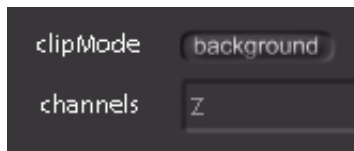
Copy the Z channel to the RGB NYC image:

- 1 In the Node View, select the *nyc* node.
- 2 Attach a *Layer-Copy* node.

- 3 Connect the *Reorder1* node to the second input of *Copy1*.



- 4 In the *Copy1* parameters, enter *z* into the channels text field. This copies the Z channel from the *Reorder1* node to the *nyc* tree.



There is one last step that is small and easily forgotten. The *ZCompose* node needs alpha channels in the images, and the NYC image does not have an alpha channel.

- 5 Load the *nyc* node parameters and enable *autoAlpha*.

A solid white alpha channel is supplied to the image.

Note: As an alternative, you can also attach a *Color-SetAlpha* after the *nyc FileIn* node.

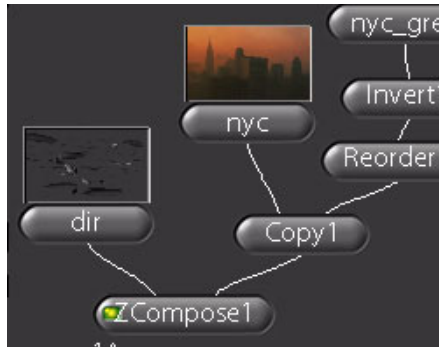


Creating the ZCompose

Since the Z channels are in place, you can now create the composite with a *ZCompose* node. The *ZCompose* function takes two images, and promotes the RGBA pixels by comparing which image has the lower Z value. Therefore, order does not matter with a *ZCompose*.

- 1 In the Node View, select the *dir* node.

- 2 Attach a *Layer-ZCompose* node.
- 3 Connect the *Copy1* node to the second input of *ZCompose1*.



And... Nothing happens!

OK, after all of that buildup, you are crushed and manically disappointed to see that the composite is only of the city—no airships in sight. Great. What a fine tutorial.

If you recall, the dirigible (*dir* image) Z values range from 0 to approximately 262, as shown earlier with the Pixel Analyzer. The *nyc* range is from 0 to 1. That essentially guarantees that the *nyc* image is always in front of the blimps, since 1 is less than, say, 131, the midpoint of the dirigible image. So, you must multiply the Z value of the city to scale it to match the Z range of the blimps.

- 4 In the Node View, select the *Copy1* node.
- 5 Attach a *Color-Mult* node (not *MMult*).



The *Mult* node has a parameter called depth. It is used to multiply, um, the depth. In the *Mult* parameters, there are three ways to modify the depth parameter:

- Drag the depth slider. Although the slider stops at 3, you can enter higher values (such as 50) manually in the depth text field.
- **Ctrl**-drag in the text field to activate the Virtual Sliders that allow you to go beyond the slider limits.
- Place the cursor over the depth text field. Hold down **Shift** and press the **Right Arrow** key. The value advances by increments of 10 (10 times the normal increment). Hold **Ctrl** for increments of 1. Hold **Alt** for increments of .1.

Things start to look mighty keen once you get past 70, and triumphantly swell around 190.

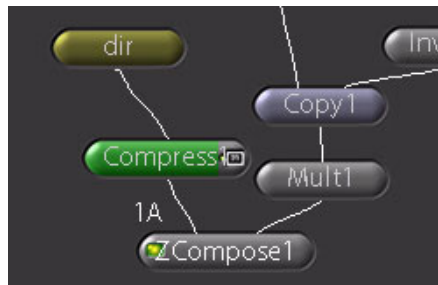


Matching the Blimp and Background Colors

You may have noticed that, gee, the blimps are black and white in the middle of a fine smoggy red sunset. In that way, at least, *nyc* resembles Los Angeles.

Use a *Compress* node to blend the blimps into the scene. This is a good node for squeezing images with a wide luminance range to match an image with a low luminance range.

- 1 In the Node View, select *dir*.
- 2 Insert a Color-*Compress* node.



The *Compress* node squeezes the blacks and whites into a colored range.

- 3 In the *Compress* node parameters, click the Low Color Picker.

The Color Picker opens and displays the ColorWheel.

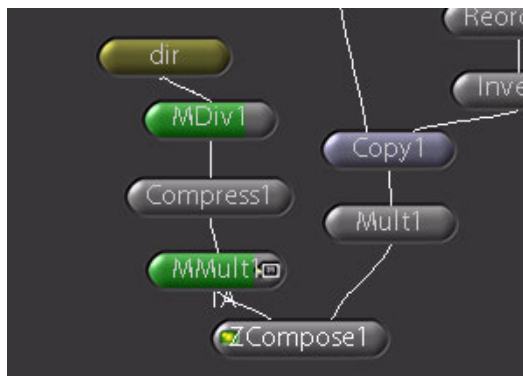
- 4 Click Min above the ColorWheel to select the darkest colors on the *nyc* image.
- 5 In the Viewer, scrub along the dark windows of the buildings.
Hey! Everything goes bright! This tutorial blows!



The entire image is brightened because you are color correcting a premultiplied image (similar to the artifacts on Vanilla in the Intermediate tutorial when her contrast was lowered). By raising the black levels, the entire image, including the space between the blimps, is raised.

As discussed in the Intermediate tutorial, use the following solution:

- 1 In the Node View, select the *dir* node.
- 2 Insert a Color-*MDiv* node.
- 3 Select the *Compress1* node.
- 4 Insert a Color-*MMult* (not *Mult*) node.

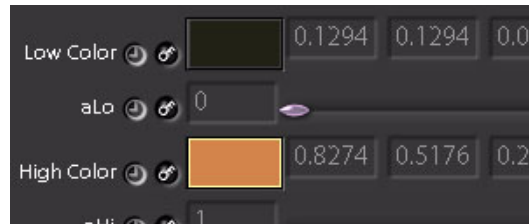


The color correction appears only on the blimps.



Next, set the higher colors.

- 5 Load the *Compress1* parameters (click the right side of the node).
- 6 Click the High Color Picker.
- 7 In the Color Picker tab, click Max.
- 8 Scrub along the sky to sample the brightest orange color.



The blimps blend with the New York City image.

Note: To gang the three values in the Color Picker together, hold down the **O** key (for offset) and click-drag left and right over the numbers. You can also use **S** (saturation), **H** (hue), **T** (temperature), and other similar channel keys, such as **L**, **R**, **G**, and **B**.

Adding Fog

In this section, add depth cueing by color correcting the blimps in the foreground differently than the blimps in the background. There are about three hundred ways to do this, but here, pull a depth-based key from the *dir* image and use it as a mask for the *Compress1* node.

Without Fog

With Fog

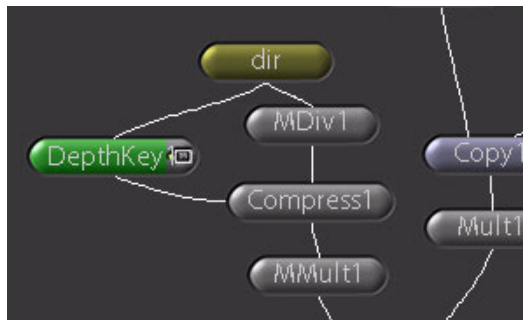


- 1 In the Node View, select the *dir* node.
- 2 In the Key tab, **Shift**-click *DepthKey*.

A new branch is created with the *DepthKey* node.

- 3 Connect the *DepthKey* output to the right side of the *Compress1* node (the mask input).

Note: The mask input automatically jumps to the left side after it is connected if the masking node (*DepthKey1*) is to the left of the masked node (*Compress1*).



- 4 In the *DepthKey1* parameters, set *hiVal* to approximately 130.

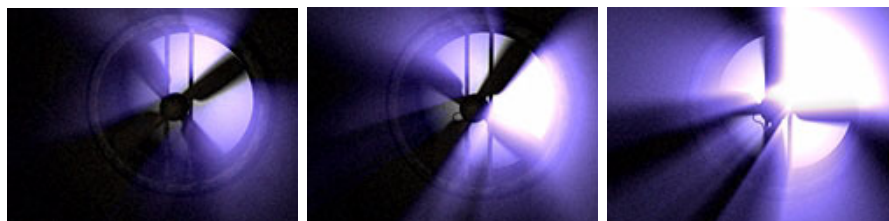
The blimps halfway to the far distance have fog at 100 percent. A key is pulled from the Z channel of the *dir* image. To verify the key, view the *DepthKey1* node alpha channel.

- 5 Tune the *hiVal* and *loVal* parameters to adjust where the fog appears. To increase the fog effect, raise the Low Color in the *Compress1* parameters closer to the bright orange of High Color.

Note: As an alternative, connect the *MMult1* node to a *Layer-KeyMix* node. The *DepthKey* is the third image input, and whatever else you want (a colored ramp, a differently color-corrected blimp, etc.) is the second input in the *KeyMix* node.

Lesson Four: Using Local Variables With Expressions

Using an image of a ceiling and five images of a rotating fan as source material, this tutorial uses local variables with expressions, radial blur, and motion blur to create the ol’ “gee, those fan blades sure have some air of mystery with that strange flickering light behind them, and it sure is dusty in here, even though there is a ventilation device” effect. You know the one. Here are some example images:



Tutorial Summary

- The Fan Composite
- Animating the *RGrad*
- Adjusting Clip Length in the Time View
- Using Local Variables With Expressions
- Using *RBlur*
- Concatenation of Color Corrections
- Adding Motion Blur to Pre-Animated Elements

The Fan Composite

The fan composite is broken down into four steps:

- Set up the basic composition, both in the frame and in time. Since you only have five images of the fan, loop the animation using the Time View. Also, add an animated *RGrad* behind the ceiling image to act as a light.
- Add a flickering effect to the *RGrad* node. This uses expressions, and explains how to use local variables to better control the expressions.
- Add the Radial Blur (*RBlur*) effect for the volume-rendering feel.
- Although it is already animated, add motion blur to the fan. It's cool.

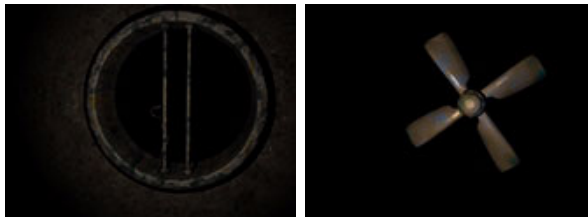
To read in the lesson files:

- 1 In the Image tab, click *FileIn* and load *Shake3/doc/pix/fan/ceiling.iff* and *fan.1-5@.iff*.

In the Load image or sequence browser window, use one of the following methods to read in the files:

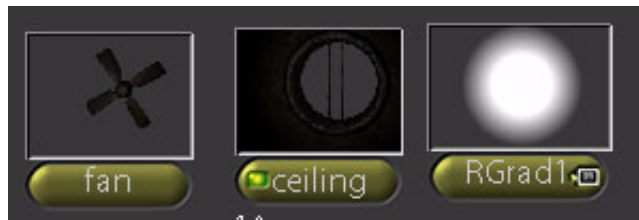
- **Shift**-click both files.
- Drag-select both files and click OK.
- Select *ceiling.iff* and press the **Space bar**, and then select *fan.1-5@.iff* and click OK. Pressing the **Space bar** is the equivalent to using the “next” button in the Load image or sequence browser window.

The two *FileIn* nodes, *ceiling* and *fan*, are added.



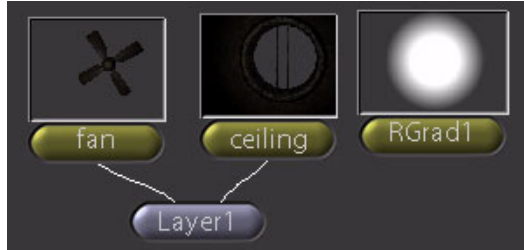
- 2 Add an Image-*RGrad* node.

You now have a mind-bogglingly complex node tree.

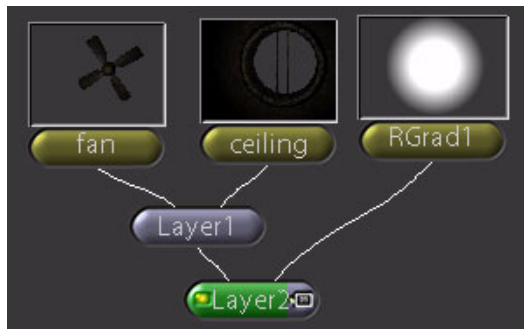


To composite the images together:

- 1 In Node View, select the *fan* node and add a *Layer–Layer* node.
- 2 Attach the *ceiling* node to the second input (background) of the *Layer1* node.
The *fan* image is layered over the *ceiling* image. By default, the *Layer1* node operation is set to Over (the layer operation can be changed to another type of operation).



- 3 Select the *Layer1* node, add a second *Layer* node, and attach *RGrad1* as the background.



- 4 Because *RGrad1* is created at the default resolution (D1), ensure the clipMode parameter of *Layer2* is set to take the foreground resolution.



The resulting image should look like the following:




Animating the RGrad

Since you only have five frames of material, you can add a little kick to the composite by animating the position of the *RGrad* so it approaches the hole in the ceiling.

To animate the RGrad:

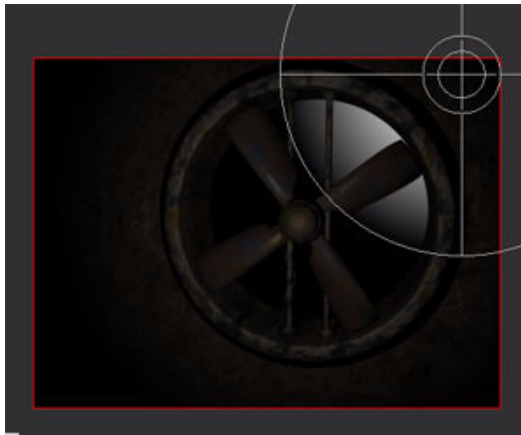
- 1 Go to frame 1.
- 2 Click the left side of the *Layer2* node (to load it into the Viewer), and click the right side of the *RGrad1* node (to load its parameters).

The *RGrad1* onscreen controls appear.

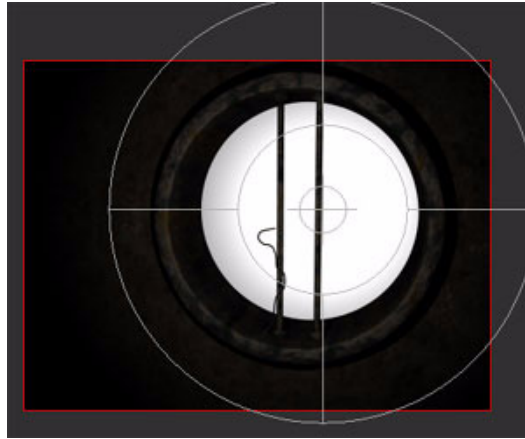
- 3 In the Viewer, enable Autokey  .

A position keyframe for the *RGrad* is created.

- 4 Using the onscreen controls, position the *RGrad* in the upper-right corner and reduce the size of the *RGrad*.



- 5 Go to frame 75.
- 6 Using the following illustration as a guide, enter a second position keyframe.



Drag the current frame on the Time Bar to see the dramatic animation. Those of you not sound asleep by now will note the fan blades disappear after frame 5, somewhat reducing the realism of the scene. Doh! You can fix this in the Time View.

Adjusting Clip Length in the Time View

Click the Time View tab to view the clip length of your nodes.



Only image generators such as *FileIn*, *Grad*, etc., and nodes that combine two branches together (*Over*, *IDisplace*, etc.) are listed in the Time View. Since the nodes you have created in this tutorial are these types of nodes, they are displayed in the Time View.

The infinity symbols that appear on the left and right edges of *ceiling* and *RGrad1* indicate that the clips (single frames) have no start or end points. The short fan clip, however, has a finite length of five frames. At frame 6, the clip disappears.

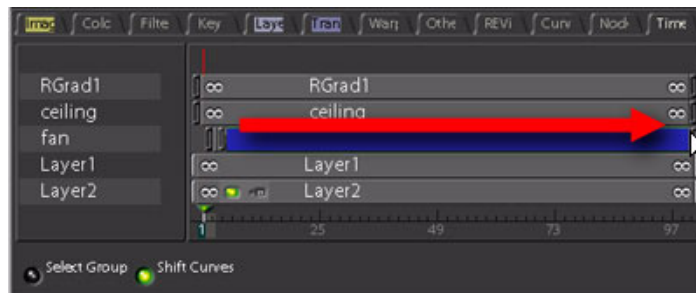


To extend the length of a clip:

- 1 Press **Ctrl** and drag the right edge of the clip to frame 75.

The clip length is extended.

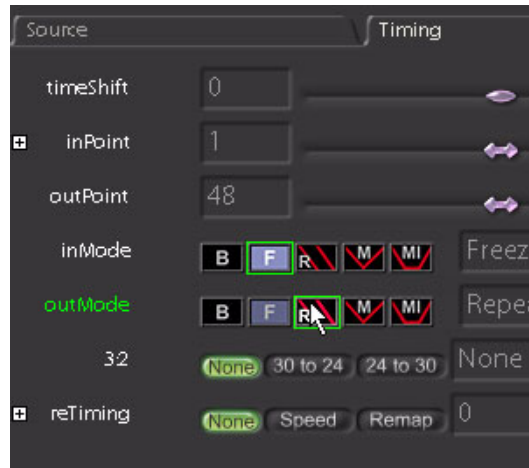
By holding **Ctrl** down, you break the relationship between the clip from the disk and the actual duration of the clip in the composite. If you did not hold **Ctrl**, the clip is dragged the from-disk range out, suggesting that there are in fact 75 frames on disk. That would be bad.








By default, the time bar is blue to indicate that the repeat behavior is frozen—the last frame of the clip is repeated to the out point.

- 2 In the *fan* node parameters, click the Timing tab.

- 3 Set the outMode to Repeat .



The inMode and outMode parameters determine the behavior of a clip that is extended beyond its normal range, as with this example.

Icon	Mode	Notes
	Black	No repeat.
	Freeze	Last (or first) frame is repeated.
	Repeat	Entire clip is repeated.
	Mirror	Clip is repeated backward, but the end frames are not repeated.
	Mirror Increment	Clip is repeated backward literally, so the end frame is displayed twice in a row.

Also, startFrame and endFrame control where in time the clip is located, and firstFrame and lastFrame control what part of the clip is read from disk. Both can be controlled in the parameters or in the Time View.

The Time View indicates a repeat of the clip, with markers that indicate the looping point.



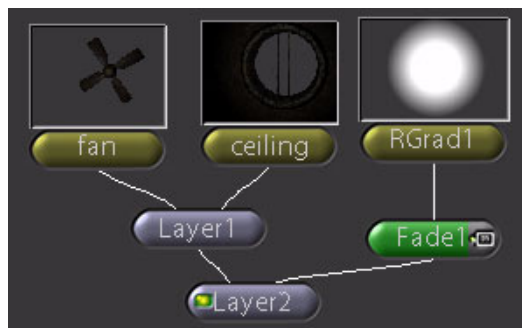
Go to the Time Bar, and click the Playback button . This differs from the normal Viewer Flipbook button in that the images are not loaded into RAM—it just renders and plays through as fast as it can. The fan blades loop.

Using Local Variables With Expressions

This step involves creating several new variables, and then combining the variables with expressions to create a controlled flickering effect on the background light (or whatever it is). This can be accomplished with manual animation, but it is much more interesting to use expressions.

Creating Flicker on the Light

- In the Node View, select the *RGrad1* node and insert a *Color-Fade* between the *RGrad1* and *Layer2* nodes.



You can, of course, enable Autokey for the *Fade1* value, and manually animate the fade for a flicker effect on the light, but you really want to show the people working next to you how superior you are in all ways moral, physical, and technical. That is, after all, the point of CG. Instead, use an expression.

To insert a function into the value text field:

- 1 In the *Fade1* parameters, double-click the value text field (to select the 1).

- 2 Type the following simple expression:

time



A plus sign appears next to the value parameter. Click the plus sign to open the expression beneath the parameter in a text field.



The keyword *time* tells Shake to take the current frame number and use it for value. Move the Time Bar slider to check the values—value increases as you increase the frame due to the time expression. This of course brightens the image too much as you get to higher frames, since at frame 70 it creates a fade value of 70. Therefore, use a different expression.

You can put any expression in the value expression text field, so do something crafty like using a random function.

- 3 Type the following in the value expression field:

rnd(time)




This returns a random value between 0 and 1, with *time* as your seed. The *rnd()* function is not truly random. Use the same seed each time, and you get the same result. By using *time* as your seed, you guarantee that the values differ from frame to frame.

Additional functions can be found in “Variables, Linking, and Expressions” in Chapter 18, “Macros, Expressions, and Scripting,” in the *Shake 3 Reference Guide*.

- 4 Click and drag the value slider.

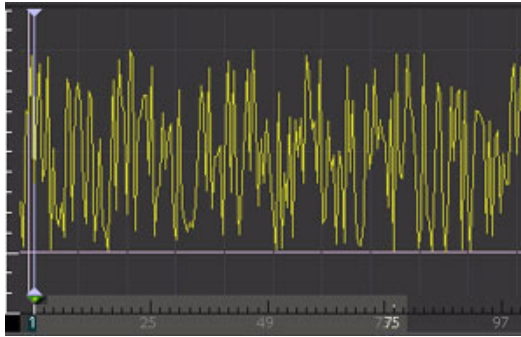
The expression is removed.

- 5 Type the expression again, or click Undo .

Note: You can also press **Command+Z** / **Ctrl+Z**.

The only exception to this behavior is if your expression is an animated curve that you create with keyframes, since curves are actually expressions. In that case, modifying the slider enters or modifies a value.

- 6 Click the Load Expression button  to view the parameter in the Curve Editor.



Your dreams of dominating your coworkers are no doubt crushed when you click on the yellow curve—nothing happens, because you are not using keyframe animation. Each value is generated by the expression.

To help control the expression:

- 1 In the *Fade1* parameters, right-click and select Create Local Variable from the pop-up menu. The Local Variable Parameters window appears. You can name the local variable and declare the variable type:

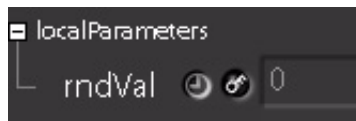
float: A number with a decimal place (0.0, .1, .5024, 1.00, etc.)

int: A rounded number (0,1, 2, etc.)

string: Characters (“Hey, what’s up?”)

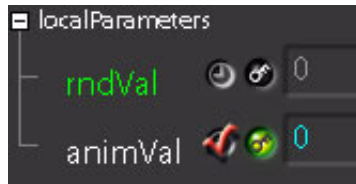
An extra parameter is added to *Fade1* (that you can use how you want).

- 2 Enter *rndVal* as the name, set the variable to float, and click OK. A new subtree named *Fade1.localParameters* appears at the end of the *Fade1* parameters list.
- 3 Open the *Fade1.localParameters* subtree to access the new *rndVal* parameter.



- 4 Adjust the *rndVal* slider—absolutely nothing happens. How exciting!

- 5 Create another local variable (right-click in the *Fade1* parameters) called *animVal*, also set to float.
- 6 Enable Autokey for the *animVal*.



- 7 Enter three keyframes at frames 1, 40, and 75. Make sure that at least one of the keyframes is set to a value of 1. The values really do not matter right now. And still, nothing happens! Oh, rapture!

The only parameter that matters to the *Fade* node is the value parameter. The others are not hooked into the value parameter, so they have no effect. To make things a little easier to work with, swap the *rnd* function from *value* to the *rndVal* parameter.

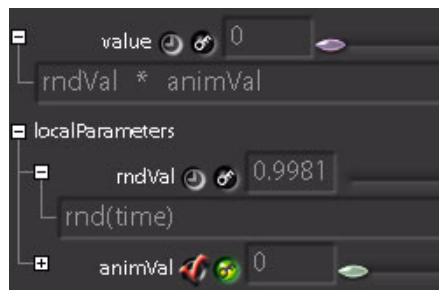
- 8 Double-click the value expression text field that reads *rnd(time)*, copy the expression (**Command+C / Ctrl+C**), and then paste the expression (**Command+V / Ctrl+V**) in the *rndVal* text field.
- 9 In the value expression box, change the expression to:

*rndVal * animVal*

This calls the two values from *rndVal* and *animVal* and multiplies the values.

Therefore:

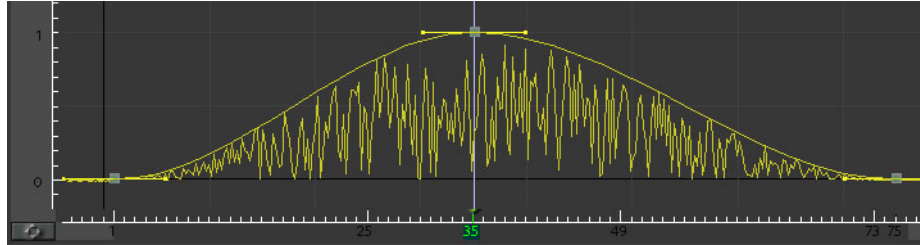
- value is set to *rndVal * animVal*.
- *rndVal* is set to *rnd(time)*.
- *animVal* is set to an animated curve.



- 10 Make sure *value* and *animVal*, but not *rndVal*, are loaded in the Curve Editor. (Click the small clock button next to the parameter name.)

- 11 The curve for *animVal* modifies the shape of the otherwise random curve. Select and move the keyframes, or insert a new keyframe (**Shift**-click on a segment), and you change the form of the random shape. If you do not see the shapes, make sure at least one of the *animVal* keyframes equals 1.

Note: To frame the curves, position the cursor in the Curve Editor and press **Home**.



This trick makes everybody warm and happy.

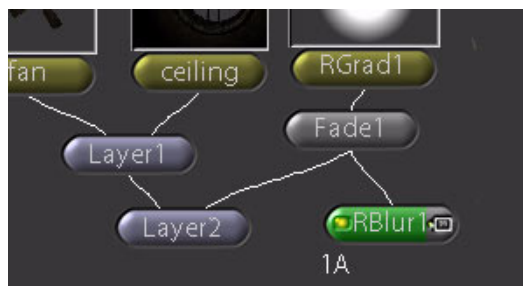
If you are using the Playback button, close the Curve Editor by switching to a different tab—when displayed, the Curve Editor and the Color Picker slow down playback speed.

Using RBlur

The following steps build the atmosphere with the Filter-*RBlur* function. *RBlur* stands for Radial Blur.

To insert the RBlur:

- 1 In the Node View, select the *Fade1* node.
- 2 **Shift**-click on Filter-*RBlur*. This creates a new branch off of the *Fade1* node.

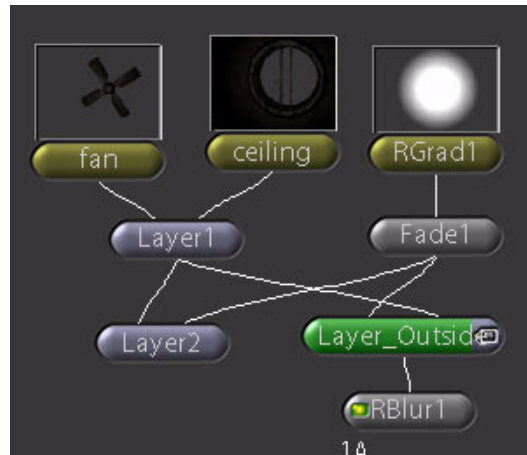


The default values do not greatly modify the *RGrad*.

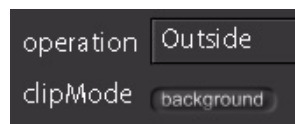


To give the RBlur more definition:

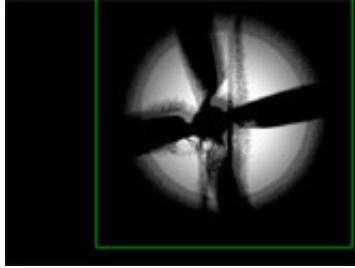
- 1 In the Node View, select the *Fade1* node, click the Layer tab, and **Shift**-click *Layer* to create another *Layer* node.
- 2 Rename the *Layer3* node to *Layer_Outside*. (Enter the new name in the *Layer3* node Layer parameters.)
- 3 Connect the output of the *Layer1* node to the second input of the *Layer_Outside* node.
- 4 Rewire the *RBlur1* node to the *Layer_Outside* node.



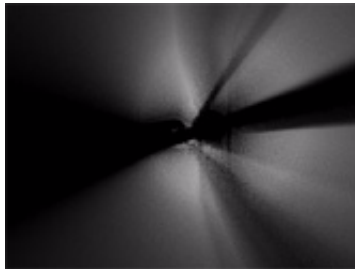
- 5 In the *Layer_Outside* parameters, set the operation to Outside.
- 6 Set the *Layer_Outside* clipMode to background.



As a result, the *RGrad* appears only where there is no alpha in the ceiling/fan composite. *RBlur1* now has a slightly more interesting effect.



- 7 In the *RBlur1* parameters, set the following:
- Set *iRadius* to 0. (There is no area of the center that is non-blurred.)
 - Set *oRadius* to 300. (This sets the blur amount.)
 - Set *amplitude* to 1. (This is a multiplier on *oRadius*—the greater the number, the more blur; also affects the rendering speed.)
 - Set *quality* to .1. (The quality is lowered to speed rendering.)



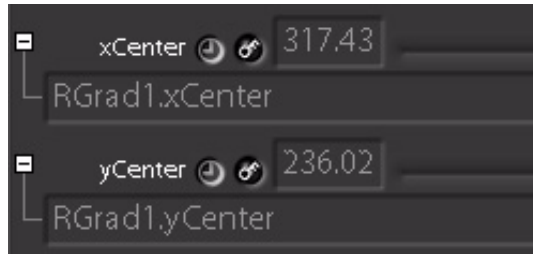
Linking the *RBlur* to the *RGrad*

Although the light is moving, the rays are emanating from the same static position. To improve the effect, animate the center of the *RBlur*, and match it to the center of *RGrad1*. The best way to do this is to link the *xCenter* and *yCenter* parameters of *RBlur1* to the *xCenter* and *yCenter* parameters of *RGrad1*. Then, if *RGrad1* is modified, *RBlur1* is automatically adjusted.

To link a parameter to another parameter within the same node, type the name of the parameter. To link to a parameter in a different node, type the name of the node, a period, and the parameter name (*NodeName.parameterName*). Both are case-sensitive.

To link the node parameters:

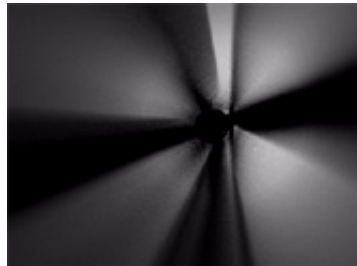
- 1 In the *RBlur1* parameters, type the following in the *xCenter* parameter:
RGrad1.xCenter
- 2 Type the following in the *RBlur1* *yCenter*:
RGrad1.yCenter



These expressions are the same as *rnd(time)* or *animVal * rndVal*, except that they check the *xCenter* and *yCenter* in the *RGrad1* node for their values.

Note: Moving the onscreen control for the *RBlur* breaks the link to *RGrad1*. To retrieve the link, click Undo.

The origin point of the rays is now animated to match the center of the light.



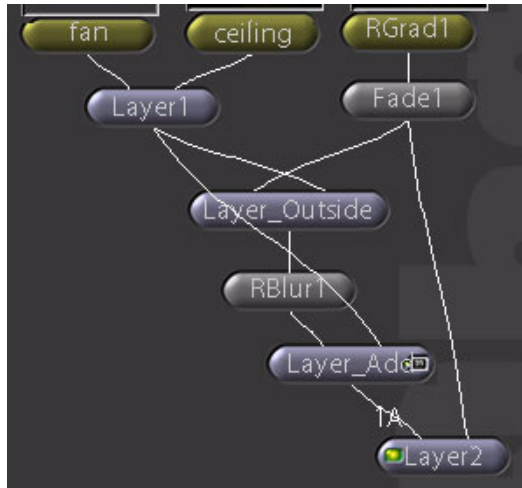
Hooking the RBlur Into the Composite

The following steps show you how to make all the node connections for the RBlur.

To connect the RBlur:

- 1 In the Node View, click the *RBlur1* node, and click *Layer-Layer*.
- 2 Rename the new *Layer* node *Layer_Add*.
- 3 In the *Layer_Add* parameters, set the operation parameter to *IAdd*.
- 4 Connect the output of the *Layer_Add* node back to the first input of *Layer2*.

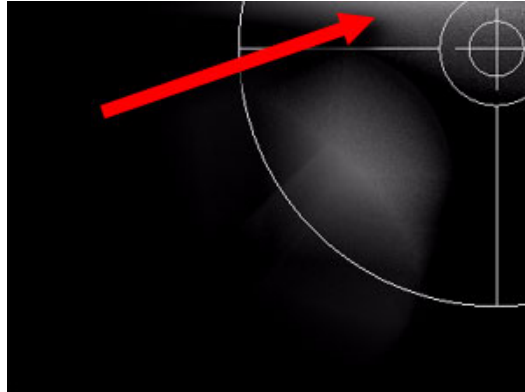
- 5 Connect the output of *Layer1* to the second input of *Layer_Add*.



Setting the operation to *IAdd* adds the two images together. *Layer2* places the atmosphere and the fan/ceiling elements over the *RGrad*.



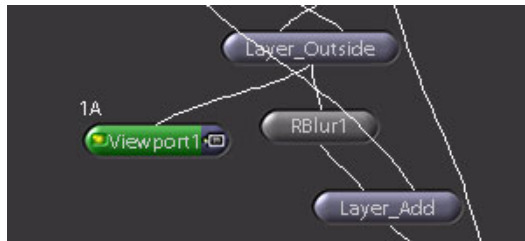
This looks keen, and you are surely very proud of yourself and the tremendous amount of labor that you have performed as you secretly congratulate yourself for not ending up slogging coal from the depths of some forgotten mine. However, if you view a frame where the *RGrad* is mostly outside of the frame (probably an early frame), light improperly bleeds in from the edges. Where does this come from?



The Shake engine has a normally divine feature called the Infinite Workspace. With the Infinite Workspace, nothing is cropped if it extends outside the frame. *Layer_Outside* should cut off the *RGrad*, except as viewed through the hole in the ceiling element. However, the *RGrad* is larger than the ceiling element, so it is not completely obscured by the ceiling element.

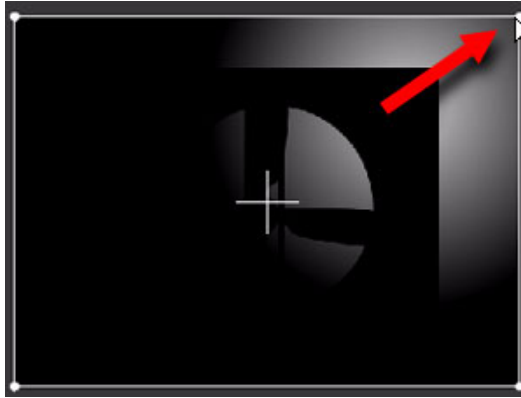
To check the extended canvas:

- 1 In the Node View, select *Layer_Outside*, click the Transform tab, and **Shift**-click *Viewport*.



Viewport is a good tool to view the extended canvas.

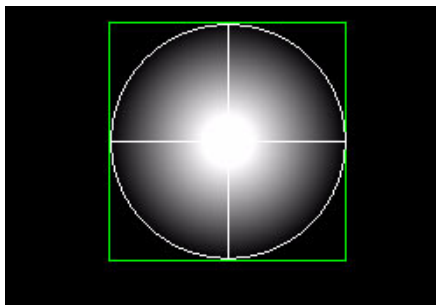
- 2 Drag the *Viewport* onscreen controls to enlarge the canvas to reveal that the *RGrad* exists outside of the frame boundary.



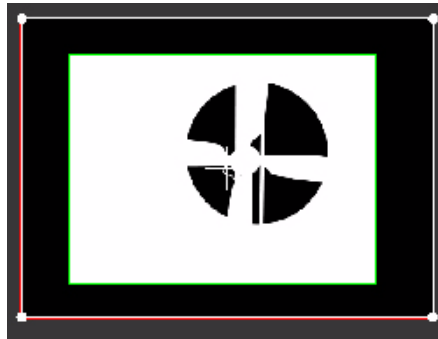
To correct this, use one of the following (but not both) solutions:

Solution 1—Control the Background Color

Background Color is considered everything outside of the Domain of Definition (DOD) of the image. The DOD is normally the frame of the image, but could be smaller. For example, when you view *RGrad1*, the DOD is described by the *RGrad* itself, and is the area inside of the green box. The green box is the DOD.



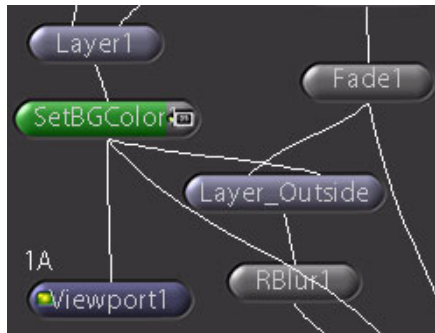
The DOD of the ceiling element is the frame border itself. If you rewire the *Viewport* (created earlier) to the *Layer1* element, you see something like the following when you view the alpha channel.



This area outside of the DOD is called the Background Color, and its color can be controlled with *Color-SetBGColor*.

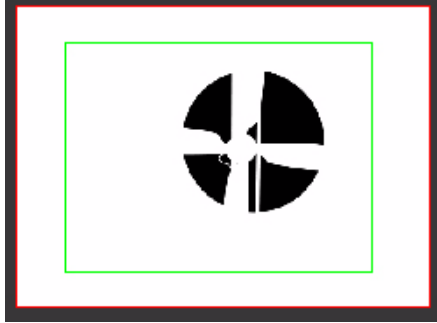
To add a *SetBGColor* node:

- 1 In the Node View, select *Layer1*.
- 2 Insert a *Color-SetBGColor*.

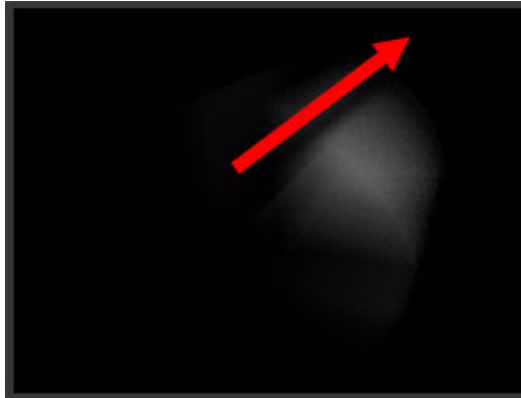


- 3 In the *SetBGColor1* parameters, set the alpha value to 1.

The solid alpha value is extended infinitely, and masks the *RGrad* outside of the ceiling frame.



- 4 View *RBlur1* (click the left side of the node).
The bleeding from the edges disappears.



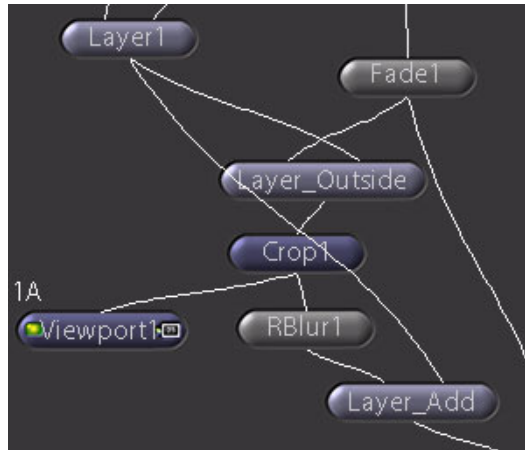
Solution 2—Clip the Infinite Workspace

The second solution is to clip the Infinite Workspace with a *Crop* node. This removes everything outside of the frame, and returns the frame to black.

To add a Crop node:

- 1 If you already followed Solution 1, delete the *SetBGColor1* node. (Select the node and press **Del / Delete**.)
- 2 Select the *Layer_Outside* node and insert a *Transform—Crop* node.

You are finished with this stage.



Compare the *Viewport1* node when it is attached to the *Layer_Outside* node with the *Crop1* node (with *Viewport1* attached) to view the difference.

Layer_Outside

Crop1



- 3 Regardless of the solution you followed, delete the *Viewport* node.

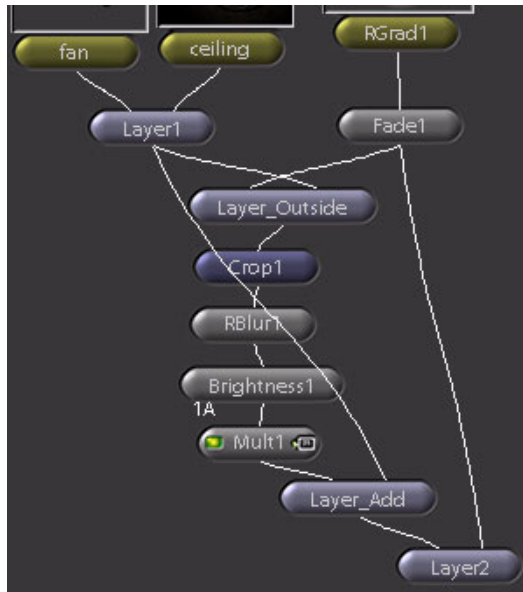
Concatenation of Color Corrections

Use the following steps to punch up the image, add a bit of color to the flaring, and brighten the image.

To concatenate the color corrections:

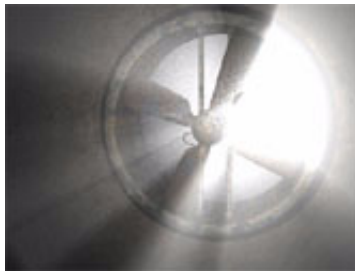
- 1 In the Node View, select the *RBlur1* node.
- 2 Attach a *Color-Brightness* node.

- 3 *Brightness1* should be active. Insert a *Color-Mult*. (Both nodes should be between *RBlur1* and *Layer_Add*.)



- 4 For the brightness, use the same animation as in *Fade1*, but double its value. You can do this with a link from the *Brightness1* value parameter to the *Fade1* value parameter. You can freely sprinkle parameter links into your expressions. In *Brightness1*, type:

*Fade1.value * 2*



- 5 In the *Mult1* parameters, set the color to blue with the Color Picker, or by using the Virtual Color Picker. To boost the blue color, hold down B and drag to the right.



You may not have picked up on this, because it is rather subtle, but *Brightness1* and *Mult1* are concatenating their color corrections. The *Mult* should be turning the entire *RBlur* blue, but you can see that the center remains white. The *Brightness* has internally raised the values of the center to around 1.5, 1.5, 1.5 (numbers are a theoretical example). When *Mult1* multiplies the color by .7, .7, 1 in the RGB channels, it lowers the color to 1.05, 1.05, 1.5. At 8 bits, this rounds down to 1, 1, 1, which is white.

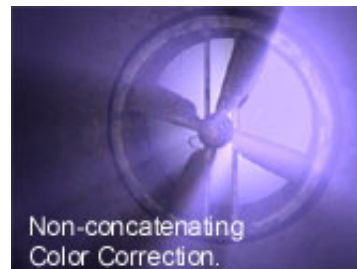
To view what occurs with no concatenation:

- 1 Insert a Filter-*Blur* node between *Brightness1* and *Mult1*.

With the *Blur* node between the *Brightness1* and *Mult1* nodes, everything turns a flat, disagreeable blue.

Concatenating

Non-Concatenating



- 2 Delete the *Blur* node to preserve the concatenation.

Adding Motion Blur to Pre-Animated Elements

By this stage, the composite may be too slow to use the Playback button, so use the Flipbook button in the Viewer to see your 75-frame animation. If only one frame appears, go to Globals, and set your timeRange to 1-75, and launch the Flipbook again. As you view the animation, notice that the fan movement appears choppy. To help smooth the animation, add motion blur to the fan without actually moving it.

Match the Fan Movement

The trick is to animate a temporary element (in this example, the letter “X”) to match the movement of the fan using a *Move3D* node. Then, the temporary element’s node (the “X”) is deleted, and the *Move3D*—with motion blur enabled—is applied to the moving element. Normally, when an animated *Move3D* node is applied to an element, the element moves. However, in the *Move3D* node parameters, you can enable the useReference parameter. The useReference parameter allows motion blur to be applied to the element without actually moving the element.

The *Text* node is good to generate a temporary element.

If reading this lesson in the printed tutorials book:

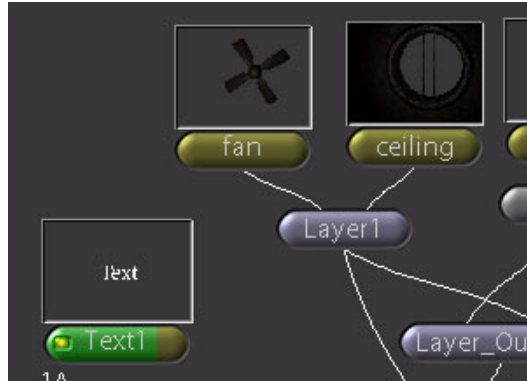
- 1 Create an Image-*Text* node.
- 2 In the *Text* node parameters, set width and height to 150.
- 3 Click the font pop-up menu and select Courier.
- 4 In the text field, type *X* as your text.
- 5 Set the xFontScale to 307 and set the yFontScale to 270.

If reading this in electronic form (pdf):

- 1 Copy the following line from your Browser (**Command+C** / **Ctrl+C**):

```
Text1 = Text(150, 150, 1, "X", "Courier", 307.74, 270.414, 1, 74.5, 134, 0, 2, 2, 1, 1, 1, 1, 0, 0, 0, 45);
```
- 2 Click in the Shake window.
- 3 Position the cursor in the Node View, and press **Command+V** / **Ctrl+V** to paste the *Text* node.

The *Text1* node appears under your cursor. Cool, isn't it?



In both cases, your result is similar to the following image.

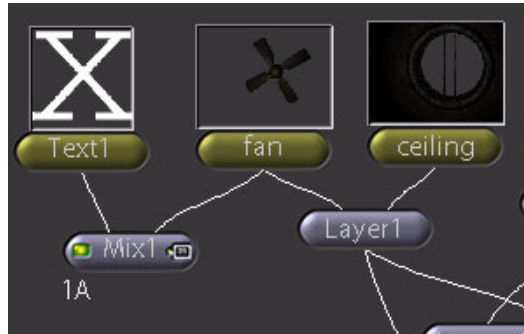


Note: To update the *Text* thumbnail in the Node View to reflect your changes, select the node and press **R**.

To mix the Text node with the fan node:

- 1 In the Node View, select the *Text1* node.
- 2 Attach a *Layer-Mix* node.
- 3 Connect the second input of the *Mix1* node to the *fan* node.

Because the *fan* is dark, boost the *fan* brightness (with a *Brightness* node), or view the alpha channel. Since this subtree is temporary, do not insert it into the rest of the tree.



Next, match the *fan* animation. The first step is to match the initial position.

If reading this lesson in the printed tutorials book:

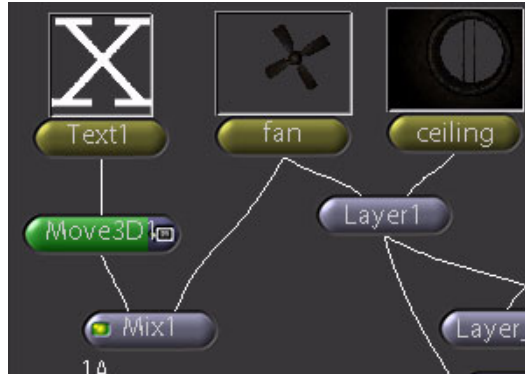
- 1 Go to frame 1.
- 2 In the Node View, select the *Text1* node and insert a Transform—*Move3D* node (see above illustration).
- 3 Set the following *Move3D* parameters:
 - Set *fieldOfView* to 15. (The *fieldOfView* value is from the 3D software's camera.)
 - Set *xPan* to 153.
 - Set *yPan* to 87.
 - Set *xAngle* to -7.6.
 - Set *yAngle* to -22.9.
 - Set *zAngle* to -196.
 - Leave *x/yCenter* at 75.

If reading this in the PDF version:

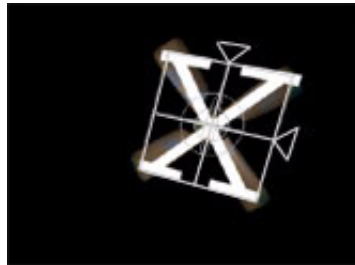
- 1 Go to frame 1.
- 2 Copy and paste the following *Move3D* node into the Node View:

```
Move3D1 = Move3D(0, 153.7649, 86.88863, 0, -7.659575, -22.97875, -196.597, 1, 0.9259, 0.9259, yScale, 75, 75, 0, 15, "default", xFilter, "trs", 0, 0, 0.5, 0, 0, time);
```


- 3 Link the pasted *Move3D* node between *Text1* and *Mix1*.

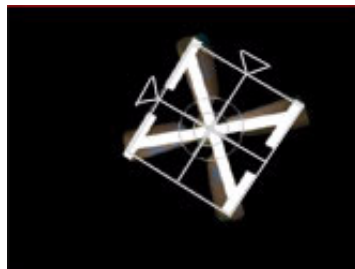


This gives you something like the following result.



To animate the X element:

- 1 In the *Move3D* parameters, click the zAngle Autokey button  to set a keyframe.
- 2 Go to frame 5.
- 3 Set the zAngle parameter to approximately -122.

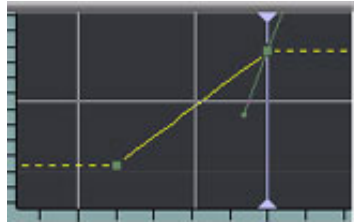


- 4 Use the **Left Arrow** and **Right Arrow** keys to step through the animation.
The animation matches between frames 1 and 5, but stops at frame 5. Use the Curve Editor to correct the problem.

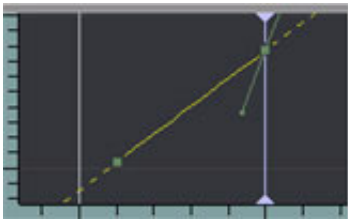
Note: When Autokey is enabled, the Load Curve button is activated, so the zAngle curve is already loaded into the Curve Editor.

- 5 Click the Curve Editor tab.
- 6 To select the curve, drag across the curve or select the curve in the curve list.
- 7 In the Curve Editor, change the curve cycle behavior from KeepValue (default) to KeepSlope. Instead of a flat value before and after the two keyframes, the zAngle parameter keeps the same slope into infinity.

KeepValue Mode



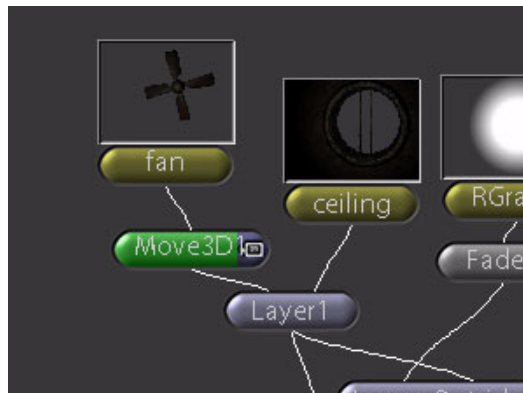
KeepSlope Mode



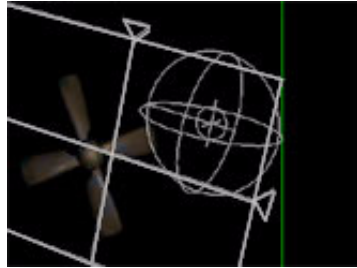
- 8 Test the animation again with the **Left Arrow** and **Right Arrow** keys. The animation continues and is matched beyond frame 5. You are almost done. Honest.

To delete the Text node and apply the animation to the fan:

- 1 In the Node View, select the *Move3D1* node and press **E** to extract the node. The node is pulled from the tree, but not deleted.
- 2 Attach *Move3D1* between the *fan* node and the *Layer1* node again.
- 3 Delete *Text1* and *Mix1*.

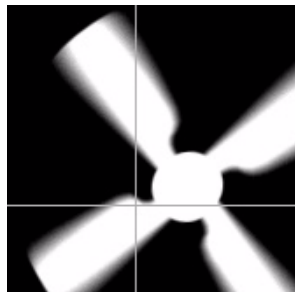


When you view *Move3D1*, the fan does not look quite right. The transformation is applied to the *fan*, which is not exactly what you want.



To apply fake motion blur to the fan:

- 1 In the *Move3D1* parameters, set *motionBlur* to *.5*.
The result still is not correct—the *fan* is transformed.
- 2 At the bottom of the *Move3D1* parameters, enable *useReference*.
The *fan* snaps back to its original position, and the motion blur is applied to the *fan*.



Note: Reel Pixel Interpolation: There is a plug-in called ReelSmart Motion Blur for motion analysis on your images. Visit the Revision Fx Web site at www.revisionfx.com.

Lesson Five: Using Keylight

This two-part lesson demonstrates pulling keys with the *Keylight* plug-in. The first section includes testing the key with a Viewer Script, masking, and color correction. The second section also discusses working with anamorphic images, removing blue spill, and using a holdout matte.

Tutorial Summary

- Part One: Using *Keylight* to Pull a Key
- Testing the Mask With a Viewer Script
- Adjusting the Mask With Parameters
- Masking
- Color Correcting the Foreground Image
- Part Two: Using *Keylight*
- Using *fgBias* to Remove Blue Spill
- Using a Holdout Matte

This tutorial discusses how to best take advantage of the *Keylight* plug-in and shows you a few masking and color correcting tricks that can be applied to other nodes as well.

Part One: Using Keylight to Pull a Key

The composite used in the first section is from the television special “Merlin,” and demonstrates the basic steps for using the *Keylight* plug-in (developed by Framestore CFC).

To pull the key on the merlin image:

- 1 Add an Image-*FileIn* node and browse to *Shake3/doc/pix/keylight*.
- 2 Read in the *merlin_fg.jpg* and *merlin_bg.jpg* images.

Note: To read in both files simultaneously, drag-select the files and click OK in the Load image or sequence browser window.

merlin_fg.jpg



merlin_bg.jpg



You are correct—that's not Merlin.

- 3 In the Node View, select the *merlin_fg* node and add a *Key-Keypress* node.
- 4 Connect the *merlin_bg* node to the second input (the background input) on the *Keypress* node.



Keypress has several color pickers. In this lesson, the important color picker is *screenColour*. (Guess in what country Framestore CFC is based.)

- 5 In the *Keypress* parameters, click the *screenColour* picker (the blue-colored box) and drag the cursor across the lower portion of the green screen color in the Viewer.

The color of the green (or blue) screen is selected. (*Keypress* does not work well with secondary colors such as cyan, magenta, or yellow.)

The result of your initial color pick is a fairly good key.



- 6 To view the alpha channel, position the cursor in the Viewer and press **A**.



Testing the Mask With a Viewer Script

The quality of your mask cannot always be gauged visually. Depending on your monitor gamma, the results of this tutorial differ. The default settings differ from system to system, or if you are in a large film facility, you may be using custom lookups to hammer the display to look like film output. (If it appears this issue is being brushed by quickly, you're right. It's far too complicated to treat in a tutorial.) Shake allows you to apply a custom gamma lookup on the Viewer to examine your images.

To test the mask:

- 1 Click and hold on the VLUT (Viewer Lookup) button.



- 2 Select the Gamma/Offset/LogLin (g/o/log) VLUT.

Note: There is only one viewer lookup installed by default; you can add your own. For more information, see “Viewer Lookups, Viewer Scripts, and the Viewer DOD” in Chapter 2, “The Shake User Interface,” in the *Shake 3 Reference Guide*.



Nothing happens, right? None of the three conversions—gamma, offset, or loglin—are activated by default.

- 3 Right-click `g/o/log` and select Load Viewer Lookup Controls into Parameter 2 tab.
- 4 In the Parameters2 tab, boost the `viewerGamma` value to approximately 2.



viewerGamma = 1



viewerGamma = 2



The heightened gamma reveals noise in the image. If you lower the `viewerGamma`, holes are revealed in the mask. You probably don't see much until you drop below .5, so don't worry about the holes for this composite. However, try to correct the background noise. (Keep in mind this image is going to be projected on a very large screen in front of millions of science fiction geeks who later take your film and advance frame-by-frame on their DVD player to reveal the multitude of errors. Perfectionism pays. You have been warned.)

Primatte allows you to do several picks to allocate foreground and background color. *Keylight*, however, relies on sliders and masking to tune the relationship between foreground and background. In this example, there are two parameters to help you. (The possibility exists your mask is fine, by the way.)

Adjusting the Mask With Parameters

The first parameter to adjust the mask is named `screenRange`. Raising the `screenRange` value increases the contrast in the mask. This slider is typically used for garbage masks that are later fed into other masks, as it tends to remove fine detail.

To test the effect of adjusting the `screenRange` values:

- 1 In the Viewer, click `g/o/log` to deactivate the VLUT.
- 2 Ensure the *Keylight* node is selected in the Node View.
- 3 In the *Keylight* Parameters1 tab, set `screenRange` to `.3`.
The mask is filled, but the edge quality disappears.
- 4 Set `screenRange` to `.03`.
The mask is filled in, but some of the hair detail drops off.
- 5 Return the `screenRange` value to 0.

You can see that this parameter can be clumsy. The second set of mask opacity controls are the Gain parameters found in the `fineControl` subtree. Raise or lower the Gain parameters to increase or decrease the mask in the shadow, midtone, or highlight areas.

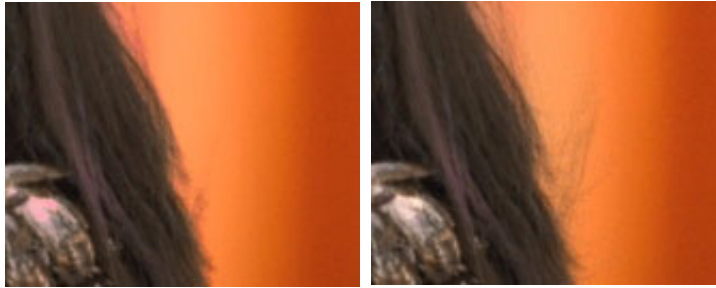
- 6 In the Parameters1 tab, click the plus sign to expand the `fineControl` subtree.
- 7 Set the `highlightGain` to approximately `.4`.

The noise disappears. However, the Viewer reveals that hair detail has been removed. This is bad.

- 8 To restore the hair detail, set `highlightGain` back to 0.

highlightGain = .4

highlightGain = 0



Note: To toggle the Viewer between alpha and color, position the cursor in the Viewer and press **C** or **A**.

So right about now, you may be thinking these parameters are useless and to never touch them and maybe there's something on TV worth watching at the moment that is far more interesting than dreary tutorials. These parameters are often useful, but this example reveals problems you must consider when keying—noise and edge detail. In this case, another approach should be taken, which is the use of masks. Hair is one of the hardest keys to pull. Filling in the inside of the mask is one of the easiest things to do. Therefore, it is better to use masks to fill in holes instead of risking losing hair detail.

Masking

In this step, draw a shape to get rid of most of the noise and still maintain the hair detail. This mask is then also used to get rid of the line on the left side of the image.

To add a rotoshape:

- 1 Display the alpha view in the Viewer.
- 2 With nothing selected in the Node View, add an *Image-RotoShape*.
- 3 Load the *Keylight_1v41 (Keylight)* node in the Viewer (click the left side of the node), and load the *RotoShape1* parameters (click the right side of the *RotoShape* node).

Note: The *Keylight* name is based on its version number.

The *RotoShape* toolbar appears in the Viewer, and the Add Shape tool is enabled by default.

- 4 Draw a loose rotoshape around the figure.

- 5 Click on the first point to close the shape.



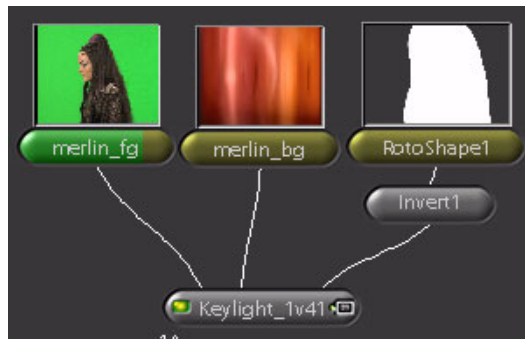
- 6 Connect the *RotoShape1* node to the fourth input (GarbageMatte) of the *Keylight* node to remove the gunk on the side of the image.

Doh! The matte is reversed and must be inverted.

- 7 In the Node View, select the *RotoShape1* node.

- 8 Apply a *Color-Invert* to *RotoShape1*.

The mask is reversed.

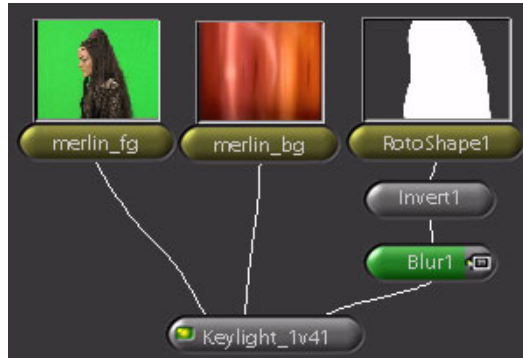


- 9 Click VLUT to activate the VLUT. You do not have to modify the values; they are saved for the session.

There is a rather suspicious line at the edge of the mask when the VLUT is enabled.



- To soften the edge, apply a Filter-*Blur* after *Invert1*. The *Blur* node is faster than using the *RotoShape* feathering.



- In the *Blur* parameters, adjust the values of the xPixels and yPixels.

Hey! As soon as the blur is activated, the top of the head gets clipped off. What's up with that? The *RotoShape1* is 486 pixels high by default and the other images are 540 pixels. So, right about now, you are screaming at us because we lied about that whole Infinite Workspace thing. Before you jump to the phone, be aware that *Blur* is the only function that gives you the option to work with the Infinite Workspace on or off. By default, it is deactivated so you can blur full-frame images without a black edge on the border.

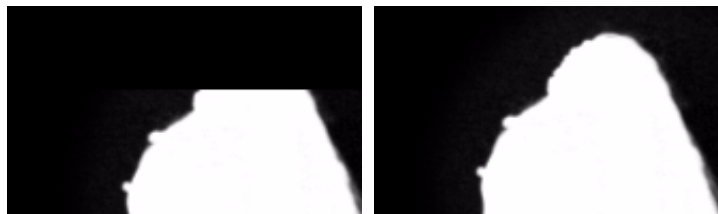
- Click the In Frame Only spread parameter to toggle the *Blur* spread parameter to Outside Frame.

The Infinite Workspace is enabled and the clipping is removed.

Note: You can also adjust the *RotoShape1* height parameter to 540 (in the *Rotoshape* parameters). As always, there are many ways to correct things in Shake.

spread = In Frame Only

spread = Outside Frame



Color Correcting the Foreground Image

To venture away from keying for a moment, this section discusses color correcting the foreground. Since the woman is composited into a red room, cast the woman with a red hue. You do not want to color correct after the *Keylight* composite, because that would change the already Amazingly Perfect Background. You cannot color correct before the *Keylight* node, because that would change the Amazingly Perfect Green Screen. You seem to be in a pickle. (If you don't speak English as your first language, feel free to drop a line for translations of any of the completely obsolete slang used in these tutorials.) Fortunately, the kids at Framestore CFC have taken this problem into account in two ways.

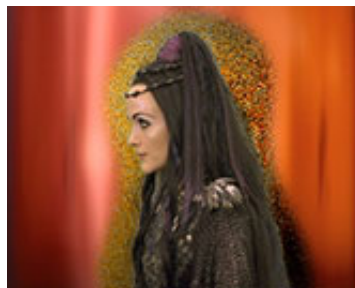
First, *Keylight* contains built-in color correction tools, for exposure (the equivalent of the Shake *Mult* node), gamma, and saturation. Feel free to use these tools if they are sufficient for your color correction. An advantage of the built-in tools is that they are calculated into the keying lookup table, so you have a slight speed advantage over using extra nodes. However, to do other operations, such as transformations or color-correction curves, you need to do something different.

It is very rare that you read in your elements, key, and composite with only one node. To apply other effects to the foreground, you must break the compositing out of the keying operations and use *Over* nodes instead. In the *Keylight* parameters, notice several unimaginative buttons for the output.



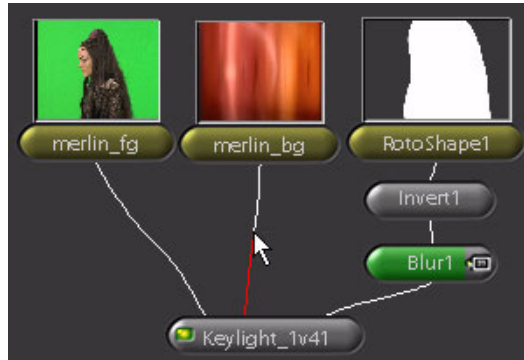
The fourth button from the left is called “unpremult”—an unpremultiplied image. The key is knocked out of the foreground, but is then divided by its mask. This setting allows you to apply color corrections before compositing over the background with an *Over* node. If you are only applying filters or transforms, set it to “on Black.”

- 1 In the *Keylight* parameters, set output to unpremult.



When the output is switched to *unpremult*, the composite looks terrible. This unpremultiplied state is temporary until you composite with an *Over* node.

- To delete the background image input into *Keylight*, **Command**-click / **Ctrl**-click the noodle.



This image still looks bad, but the background is not in the composite, so you are good so far.

- In the Node View, select the *Keylight* node.
- Attach a *Color-Compress* node.
- The *Compress* node is used for the color correction.
- Attach a *Layer-Over* node to the *Compress1* node.
- Connect the *merlin_bg* image to the second input of *Over1*.



Here is the crucial trick:

- In the *Over1* parameters, activate *preMultiply*.

The nasty gunk around the character is removed.



For the color correction:

Once again, you are matching a foreground image to a limited background image.

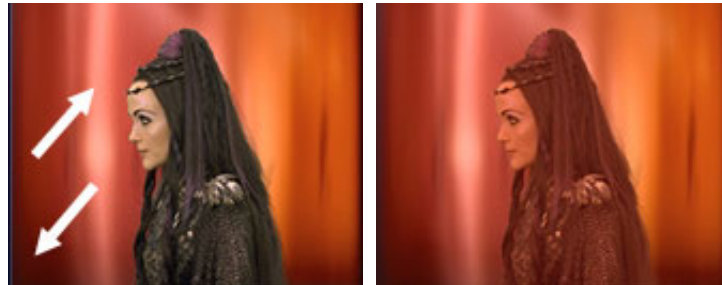
- 8 Load the *Compress1* node parameters.



- 9 In the *Compress* node parameters, click the Low Color box and scrub in the dark corner of the background.

The low color is selected.

- 10 To select the high color, click the High Color box and scrub on a bright portion of the background.



Note: In the Color Picker tab, Use Source Buffer is off. This is good, since you want to scrub color from the composite rather than from the unpremultiplied image coming out of *Keylight*.

Although this is an extreme color-correction used for illustration purposes, the principal concept here is that you spit out an unpremultiplied image, do your color corrections or transformations, and then composite with preMultiply enabled. You can alternatively use On Black for your output from *Keylight*, apply a Color-*MDiv*, do your color corrections, and then apply a Color-*MMult* to get the same result, but this complicates the tree.

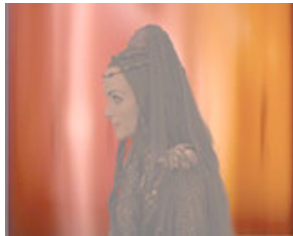
Testing a Color Correction

The following is a handy trick. Use the VLUT to gauge the color correction. If it is a good correction, the foreground blacks should be somewhat similar to the background blacks when the gamma is boosted, and the brights should be similar when the gamma is lowered below 1.

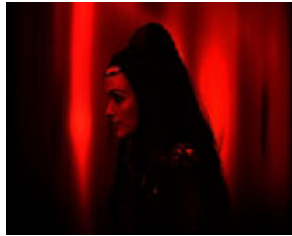
- In the Viewer toolbar, ensure the *g/o/log* VLUT is active and adjust the viewerGamma slider from .1 to 4 to evaluate the correction.

Note: If your VLUT parameters are not loaded, activate the VLUT in the Viewer (toggle VLUT to *g/o/log*), right-click *g/o/log*, and select Load Viewer Lookup Controls into Parameter 2 Tab.

A Bad Correction



Highlights Are Good



Lows Are Good



If things look good, deactivate the VLUT.

You have now pulled a basic key using *Keylight* and tested some of its parameters to adjust the mask. Since the parameters were not sufficient to remove the grain of the image but keep the hair detail, a garbage mask was created, thereby removing most of the noise. Afterward, a color correction was applied. You used a Viewer Lookup to test both the key and the color correction.

Part Two: Using Keylight

The composite used in the following section was created by Framestore CFC for the film “The Saint” (ahem, not using Shake, but Cineon, it is believed). For this lesson, use the five foreground and background images (at video resolution) in the *doc/pix/keylight* directory. “The Saint” images are provided courtesy of Framestore CFC and Paramount British Pictures Ltd.

This part of the tutorial focuses on more *Keylight* parameters and also discusses the use of anamorphic images.

To begin the composite:

- 1 Add an Image-FileIn node, and load the *saint_fg.1-5#.jpg* and *saint_bg.1-5#.jpg* image sequences.

The Joy of JPEG

Never use JPEG images for any footage you have to key. The compression causes major problems, as you will find in this tutorial. Also, do not use JPEG images for video interlaced footage, as the fields become corrupted.

The first thing you notice is that these are anamorphic frames, so everything is squeezed. Additionally, the element is originally a scanned Cineon plate, so it is in logarithmic color space. First, focus on the aspect ratio issue.

- 2 In the Globals tab, open the useProxy subtree and set the proxyRatio to .5.

Note: You can also set the proxy in the useProxy text field as 1, .5.



The images are squeezed by half in Y.

proxyRatio = 1



proxyRatio = .5



This squeezing can be turned on and off very quickly with the proxy button at the top of the interface. “Other” (highlighted in green) indicates you are using a custom proxy setting rather than a preset (P1, P2, P3). The scaling is just temporary while you work with the script. When finished with the composite, set the proxyRatio to 1.

- 3 In the Node View, select the *saint_fg* node and apply a Key–*Keylight* node.
- 4 Connect the *saint_bg* node to the second input (background) of the *Keylight* node.



- 5 In the *Keylight* parameters, click the screenColour picker box and scrub the back window of the car. This is a good place to scrub because there are no reflections (like the side window).



The lesson images are logarithmic (that is, filmic, rather than linear digital or video) color space images. Don't fret—you can specify the color space the node is working in the lower portion of the *Keylight* parameters.

- 6 Set colourspace to log for this composite.



Note: All masking data is dependent on your monitor gamma. Use the VLUT to ensure you have an accurate key as you proceed with the tutorial.



Using fgBias to Remove Blue Spill

Some blue spill appears on her hair on the right side.



Since the blue spill is subtle, you can linearize the log plates to return it to something resembling the real world.

To linearize the log plates:

- 1 In the Viewer toolbar, set  to .
- 2 Right-click the g/o/log button and select Load Viewer Lookup Controls into Parameter 2 Tab.
- 3 In the Parameters2 tab, set viewerLogLin to LogToLin.
- 4 Ensure viewerGamma is set to 1.

To help get rid of the blue spill:

- 1 In the *Keylight* parameters, click the fgBias Color Picker, and drag along her hair (the blond part, not the blue spill part).

Scrub Blue Spill on the Hair

With Spill Suppression



The blue is removed. (A ringing effect may occur due to the JPEG compression on the foreground element. Nothing you can do about that for this tutorial. Sorry.)

To minimize any yellowness that may occur in the image, you can decrease the fgBias saturation. The stronger the saturation of the fgBias color, the stronger the yellow tint.

- 2 Position the cursor over the fgBias Color Picker, press and hold **S**, and drag left.
The saturation of the color is decreased.



You can also set the balance parameters to 1 to reduce the yellow tint. It does not reduce all of the yellow—her scarf and map may still retain some yellow—but it helps.

The yellow color correction is further adjusted in the “Using the replaceColour” section, below.

Using a Holdout Matte

Show the alpha channel in the Viewer (position the cursor in the Viewer and press **A**), and you are no doubt appalled at what fgBias has done to the matte. Return to the composite view (press **C**), and notice the snow on the road is visible behind the seat belt.

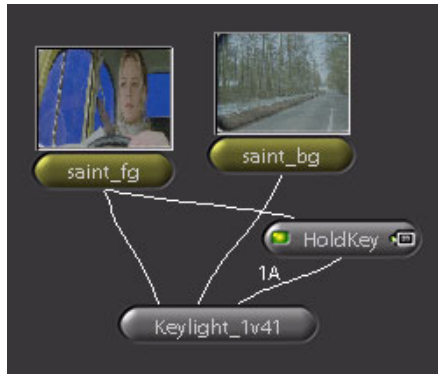


A common technique is to copy your first *Keylight* node, boost its contrast, reduce the size of the mask by chewing into it, and then feed it back into the first *Keylight* node.

To create a holdout matte:

- 1 Copy the *Keylight* node. (Select the node, press **Command+C** / **Ctrl+C** to copy and **Command+V** / **Ctrl+V** to paste.)
- 2 In the new *Keylight* node parameters, rename the node *HoldKey*.
- 3 Connect the *saint_fg* node to the first input of the *HoldKey* node.

- 4 Connect the *HoldKey* node to the third input of the *Keylight* node.



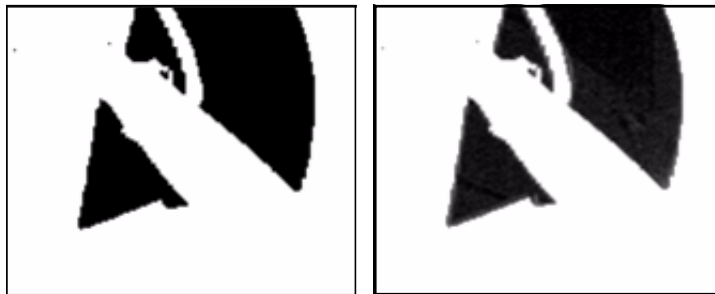
- 5 In the *HoldKey* parameters, boost the screenRange to .3.

The mask is solid.

When the holdout matte is plugged in, it acts as a mask for the foreground. You are, of course, not obliged to use *Keylight*. You can also use rotoscoping, *Primatte*, *Chroma/LumaKeys*, etc. The following illustrations show the *HoldKey* alpha and the *Keylight* alpha. *HoldKey* contains no grey tones, *Keylight* retains the window reflections.

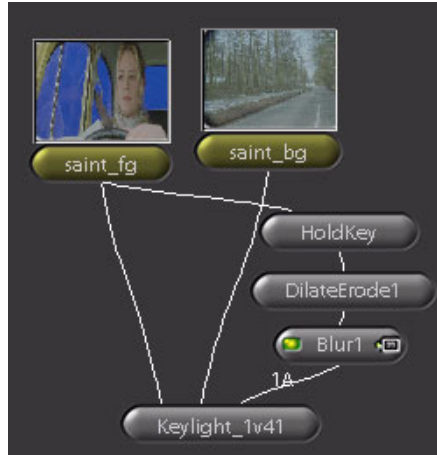
HoldKey

Keylight



The holdout matte has destroyed the edge quality of *Keylight*. To correct the edge quality, chew into the edge of the holdout matte.

- 6 Insert a Filter–*DilateErode* node and a Filter–*Blur* node between the two *Keylight* nodes.



- 7 In the *DilateErode1* parameters, set x/yPixels to approximately -3 to chew into the matte.
- 8 In the *Blur1* parameters, set x/yPixels to approximately 6 to soften the matte.

Blur1



Keylight



The holdout matte strengthens the normal key pulled by *Keylight*. Because the *DilateErode* node reduced the size of *HoldKey*, the matte does not interfere with the edges.

Test the effect of the holdout matte:

- In the lower portion of the *Keylight* parameters, expand the plumbing subtree, and enable `useHoldOutMatte`.

Without Holdout Matte

With Holdout Matte



Using the `replaceColour`

You can also use the holdout matte with the *Keylight* `replaceColour` parameter to help eliminate the yellowish tint. This is for areas that are keyed out by mistake because they are lit with more blue light than white light, and are also included in the holdout matte area.

To help correct the lighting with `replaceColour`:

- 1 In the Viewer, show the composite (press **C** with the cursor in the Viewer).
- 2 In the *Keylight* parameters, click the `replaceColour` Color Picker.
- 3 In the Viewer, select a color that is the opposite of the warm yellow—a mid-saturation purple.

In the following split illustration, the uncorrected yellowish image is on the left. The image with the replace color is on the right.



As an end note, like all keys, *Keylight* is not a magic bullet—a key rarely works with a single pixel scrub. Feel free to use *Keylight* in conjunction with other mask techniques and keys, and combine the keys through the use of the *KeyMix* function, or through the addition or subtraction of mattes with tools such as *IMult*, *IAdd*, *Outside*, *Inside*, *KeyMix*, *Max*, etc.

Lesson Six: Using Primatte

This lesson describes the basic use and mechanics of the Photron *Primatte* keying plug-in, as well as masking and spill suppression.

Tutorial Summary

- The Basics of Pulling a Key in *Primatte*
- Inner Mechanics of *Primatte*
- Masking *Primatte*
- Spill Suppression in *Primatte*
- Alternatives

The Basics of Pulling a Key in Primatte

The example images for this lesson are located in the *doc/pix/primatte* directory.

To pull the Primatte key:

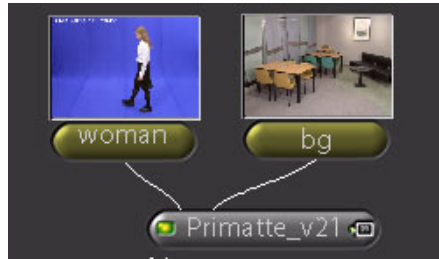
- 1 Add an Image-*FileIn* node and read in the *woman.iff* and *bg.jpg* images.



Note: If you are still in .5 proxyRatio (as set in Lesson Five), click the Globals tab and set proxyRatio to 1.

- 2 In the Node View, select the *woman* node.

- 3 Add a *Key-Primatte* node, and connect the *bg* node to the second input (background) of the *Primatte* node.



In *Primatte*, you select a “center” value that is the average color of the key you want to pull. The concept of center is explained in “Inner Mechanics of *Primatte*” on page 138.

Unlike *Keylight*, *Primatte* is ready to scrub your background color as soon as you add the node.

In the *Primatte* parameters, ensure the center Color Picker is selected (a yellow line appears around the Color Picker box).



- 4 In the Viewer, drag across the blue screen in the background.



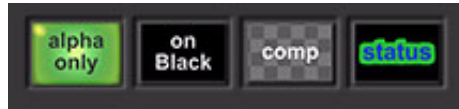
Nothing appears to change because only the alpha channel is initially modified. The output of the node is determined in the output parameters.

- 5 Position the cursor in the Viewer and press **A** to view the alpha channel.

The alpha channel shows a basic key is pulled. The majority of the background should appear black in the alpha channel.

Note: If most of the background is medium grey, the Color Picker was inactive when the blue screen color was selected—make sure the picker is active, and scrub the blue screen again. To select a different blue screen color, make sure the picker is active and scrub in the Viewer.

Output, one of the first parameters in *Primatte*, determines what is passed out of the node. The output is set to *alpha only* by default.



Primatte contains the following output parameters:

- *alpha only*—Only the alpha channel is modified; no spill suppression is performed.
- *on Black*—Foreground is composited over black and spill suppression is performed.
- *comp*—Foreground is composited over the background image with spill suppression.
- *status*— Different zones are color coded:
 - Black is pure background.
 - Red is pure foreground.
 - Blue is transparent areas.
 - Green is transparent with spill suppression.

To set the output mode to view the composite:

- In the *Primatte* output parameters, enable *comp*.

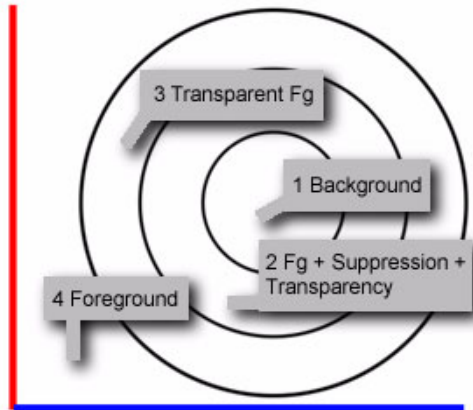


The composite appears in the Viewer.



Inner Mechanics of Primatte

So, what is this puppy doing all over your images? Whereas *Keylight* is based on a model of light and its properties, *Primatte* works by assigning color to one of four different zones through a series of color scrubs. Each zone has its own qualities. These zones are constructed in a 3D space created with the three RGB axes. The following illustration is a 2D representation of this 3D space.



Based on this model, you expect most of the pure blue to fall inside zone 1 (Background); colors near blue in zone 3 (Transparent Fg); and the rest of the colors in zone 4 (Foreground). Zone 2 is not yet used, because you have not assigned any color to the zone.

- In the *Primatte* output parameters, enable status.

The black pixels are zone 1 (bg), the red pixels are zone 4 (fg), and the blue pixels are zone 3 (transparent fg).

- Return the output parameters to comp.

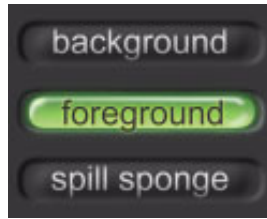
Do these zones affect your life in such a way that they inhibit or aid your ability to make frequent purchases of Cool Ranch Corn Chips? Absolutely not. It's just a bit of information so you can better understand the Inner Beautiful Primatte.

To assign color to different zones, pick one of the operator buttons and scrub color in the the foreground image. For example, her shirt is transparent in some places.

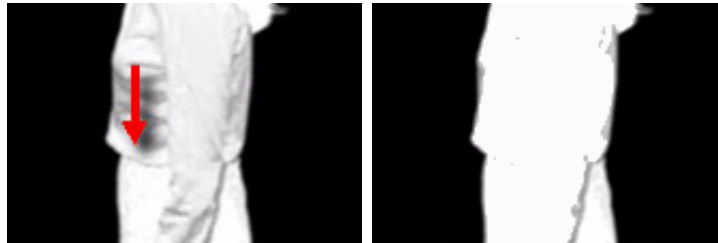
To add more of the shirt to the foreground using the foreground operator:

- 1 In the Viewer, press **A** to view the alpha channel.
- 2 In the *Primatte* operator parameters, click the foreground operator button.

The foreground operator is assigned and a new Color Picker is automatically activated.



- 3 In the Viewer, scrub the shirt area.



Note: The Undo function and *Primatte* are not best friends. To remove an operator, use the delete op button in the *Primatte* parameters. To pick a different color, click the Color Picker and scrub.

- 4 To sample more foreground area, click the foreground button again and scrub.
- 5 To re-scrub, ensure the Color Picker is active and scrub again.

When using *Primatte*, knowing when to stop is important. The more scrubs you perform with the foreground and background operators, the crunchier the matte edge becomes (this is bad).

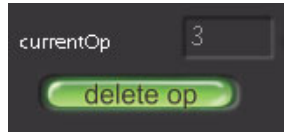
For example, you cannot get the dark bit under her arm. Kind of like that nasty corner in your shower, you can scrub on it all day but it doesn't disappear. This is a clue that this particular color is buried way down in zone 1 (pure background). If you try to assert this color as foreground, you are going to upset the entire keying model (that is also bad).



If you tried to get the dark area under her arm with a foreground operator and failed miserably, delete the operator.

To delete an operator:

- 1 In the *Primatte* parameters, click the delete op button below the currentOp parameter.



Instead, use a holdout matte to mask problem areas such as the dark portion under her arm. Knowing when to cut your losses and start doing a roto is a sign of maturity and true inner spiritual growth. This is covered in the “Masking Primatte” section below, so take your shoes off and put on your bunny slippers. You’ll get there in a moment.

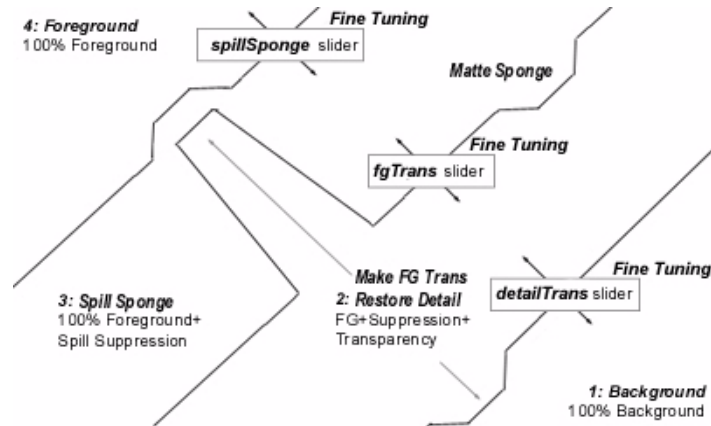
The background of the key, especially on the left side, needs work.

- 2 In the *Primatte* parameters, click the background operator and scrub the left side of the image.



By using the foreground and background operators, you have assigned pixels to either zone 1 or 4. Picking any of the eight operators assigns color to a different zone. The following image is a chart of the operator/zone assignments. Note that “decolor all” scales the entire 3D space—the shells are scaled toward or away from the Center point—and therefore does not involve picking a color (only moving a slider). Also, to make fine tuning effective, pick a color and then adjust the sliders. These operators are explained in the next section.

This illustration demonstrates (roughly) how each operator dimples the surface of the shells through the use of successive color picks. (You may have seen implementations of *Primatte* in other software using a real 3D representation of the color space using polygons. While cool, only about three people on the planet seem to understand it.)



Managing Operators

One swell thing about Shake's implementation of *Primatte* is that you can review, modify, or delete previous operators at any time. You do not have to start over from scratch if you are not happy with the key. Use the parameters shown in the following illustration to review, modify, or delete previous operators.



The currentOp slider scrolls from 0 (the Center pick) to the number of applied operators. Each operation can be accessed by using the slider. The Center pick is always 0, and cannot be deleted.

To modify previous operators:

- Use the currentOp slider to select the operation you want to modify. Operations are numbered in the order applied. For example, if you applied a foreground operation, a second foreground operation, and then a background operation, the first foreground operator is 1, the second foreground operator is 2, and the background operator is 3.

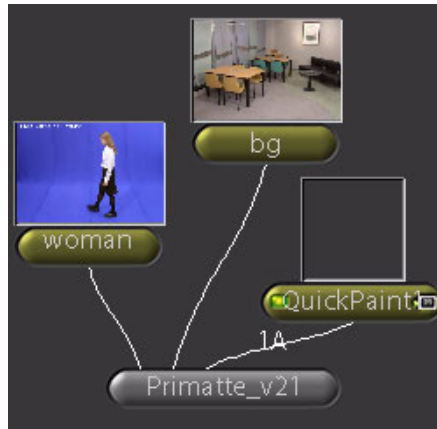
- Enable `evalToEnd` to view the effect of all applied operators. When disabled, only the operations up to the currentOp are displayed. For example, if you have 10 operators and your currentOp is set to 3, only operations 0 through 3 are calculated by *Primatte*.
- Use the “active” parameter to quickly toggle an operation on and off to test its effect.
- To rescrub a color, click the Color Picker and scrub in the Viewer.
- To remove an operator, press the delete op button.

Masking Primatte

In this section, use a holdout matte to clean up the key under her arm, and a garbage matte to eliminate the “image courtesy of Photron” text (which your client probably doesn’t want, unless your client is Photron). Therefore, holdout mattes add to opacity, garbage masks add to transparency.

To attach a garbage matte:

- 1 Click the Image tab, press **Command+Shift / Ctrl+Shift**, and click *QuickPaint*.
An unattached *QuickPaint* node is created.
- 2 Connect the *QuickPaint* node to the third input (garbageMatte) of the *Primatte* node.



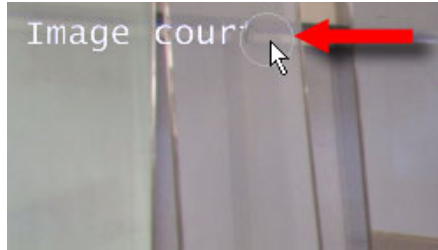
- 3 Show the composite (*Primatte* node) in the Viewer. (Click the left side of the node.)
- 4 Make sure the *QuickPaint* node parameters are loaded. (Click the right side of the node.)
- 5 In the *QuickPaint* toolbar, click the soft-edged main brush (default) to toggle the selection to the hard-edged brush.



Note: With the cursor in the Viewer, you can press **F3** to toggle brush types.

- 6 Drag the brush across the text.

The text disappears.



Note: As an alternative, you can connect the *QuickPaint* node between the *woman* node and *Primatte* node. Paint over the text using a color picked off of the blue screen. *Primatte* then automatically makes that area invisible as it keys out the blue color.

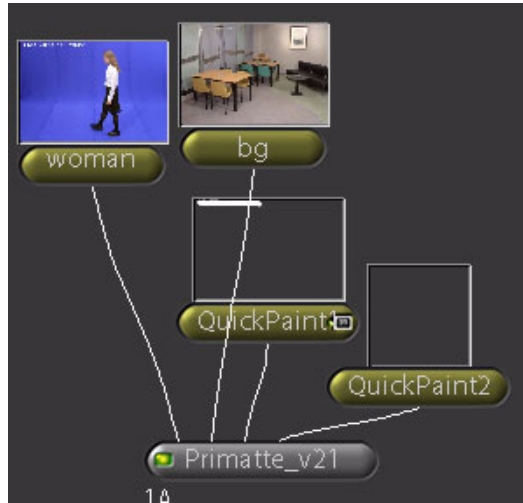
To select the channel for the garbage matte, use the *gMatteChannel* settings in the *Primatte* parameters.

Attach a Holdout Matte

Use a holdout matte to promote areas that should be visible, but are keyed out because the color is close to the blue screen or green screen Center color. For example, use a holdout matte to eliminate the line under her arm (since the foreground operators could not clean it up earlier).

To create a holdout matte:

- 1 Add a second *QuickPaint* node (or *RotoShape* node), and connect the node to the fourth input (holdout matte) of the *Primatte* node.



Note: To identify inputs on the top of a node, position the cursor over the input and read its name in the Help box at the bottom-right portion of the interface.

- 2 Using the brush in the *QuickPaint* node, paint a holdout matte along the line under her arm.



To select the masking channel of the holdout matte, use the *hMatteChannel* settings in the *Primatte* parameters.

A final parameter that helps with multiple masks is the arithmetic parameter. The arithmetic parameter indicates how the *Primatte* key interacts with the alpha channel of the foreground. By default, it replaces the alpha channel, but you can also add, subtract, or multiply the two masks. With this technique, you can pull several masks and combine the masks within *Primatte*. Since there is no incoming mask in this example, the arithmetic parameter has no effect.

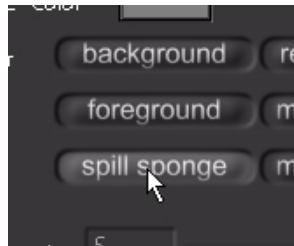
Spill Suppression in Primatte

Primatte has spill suppression tools. Spill suppression is the removal of reflected light (blue or green spill) on the foreground material that comes from the blue screen itself. In this example, blue spill is immediately evident on her white shirt. Although the suppression tools in *Primatte* are interesting and useful, as a general principle, you are encouraged to perform the suppression outside of the *Primatte* node with nodes such as *HueCurves*, *SpillSuppress*, and *ColorReplace*. As one client put it, “Why would you want to tackle two difficult tasks at once?” With that chilling little preamble, continue.

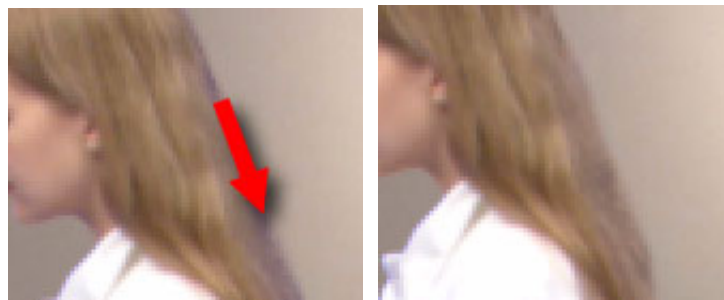
In this example, there are three primary areas where the blue spill is apparent: The edge of her hair, the shadows on her shirt, and possibly the little transparent bit of her skirt at her knees. The more foreground operators you apply, the more areas you must spill suppress. On the flip side, the fewer foreground operators you use, the more rotoscoping or paint touch up you must do. (Isn't keying fun?) A third option is to pull two keys with two separate nodes, and feed one key into the other.

To use the spill sponge operator:

- 1 In the *Primatte* parameters, click the spill sponge operator.



- 2 In the Viewer, drag along the purplish edge of her hair.

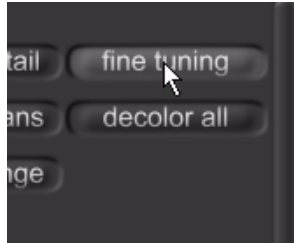


- 3 You may need to apply two or three spill sponge operators. The spill sponge operator applies a precalculated amount of spill suppression, and replaces the blue color with an alternative color.

For the spill on the shirt, you can use additional spill sponge operators, but instead use the fine tuning operator. The fine tuning operator is the same as using spill sponge, restore detail, and make fg trans—the last two either bring back fine details such as individual hair strands, or push foreground material into the slightly transparent zone. The fine tuning operator is much easier to use than the spill sponge, restore detail, and make fg trans operators, as it has sliders to tune how much of those particular effects you want.

Use the fine tuning operator:

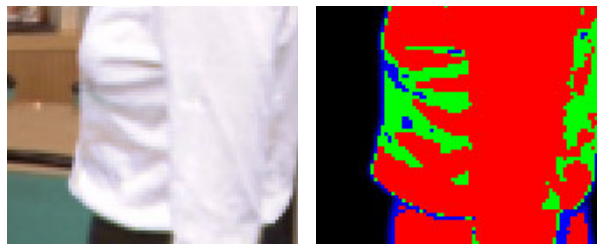
- 1 In the *Primatte* parameters, click the fine tuning operator.



- 2 In the Viewer, drag along the bluish part of her shirt.



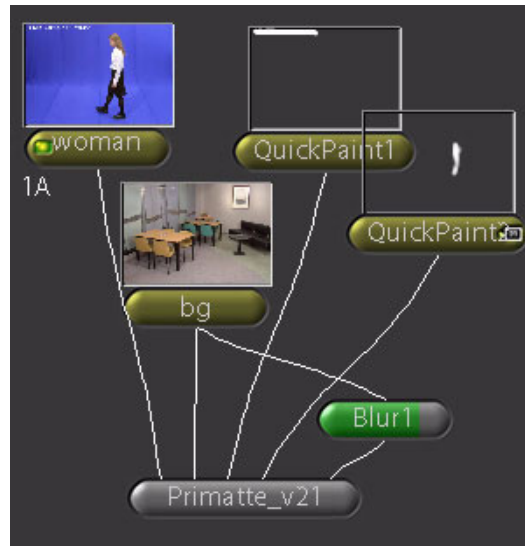
- 3 Adjust the spillSponge slider (under the Color Picker box). The more you slide, the greater the range of suppressed area.
- 4 To view the affected areas, toggle output to status (in the *Primatte* parameters) and move the spillSponge slider up and down. The green zone (spill suppression) increases and decreases in size.



The spill-suppressed image appears slightly transparent when you return the output operator to comp. This is because *Primatte* pulls in background color to replace the blue color. Here is the logic: If you stand in front of a blue screen and you have blue spill, then conceivably if you stand in front of a red wall, you have red spill. Therefore, *Primatte* sucks in the background image color to replace the removed blue screen color. This is sometimes sufficient, but often needs a little help. If the definition of the background is too sharp (for example, the black top of the chair appears through her shirt), you must modify the image from which *Primatte* pulls the replaced color. By default, it pulls it from the background image. This can be replaced by inserting an image into the sixth input on the *Primatte* node.

To use the replaceImage input:

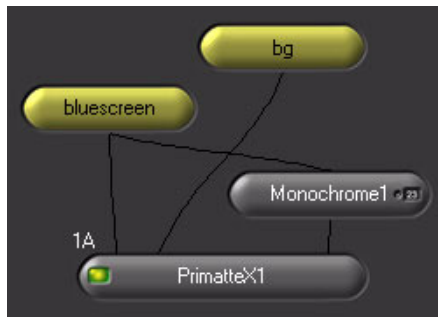
- 1 In the *Primatte* parameters, set the output to comp.
- 2 In the Node View, select the *bg* node.
- 3 **Shift-click** Filter-*Blur*.
- 4 In the *Blur* parameters, set the xPixels and yPixels to 200 (an arbitrary value in this case).
- 5 Connect the *Blur* node to the replaceImage input (the last input on the node) of the *Primatte* node.



Since the black chair backing has lost its definition, the shirt appears less transparent. The alpha channel remains solid.



Note: Another popular trick with the kids these days is to place a Color–*Monochrome* node on the foreground image and connect it as the *replaceImage* instead of a *Blur* node. Other useful nodes are *Key–SpillSuppress*, *Color–AdjustHSV*, *Blur*, and *Monochrome*, all on the foreground image. The following is a generic example.



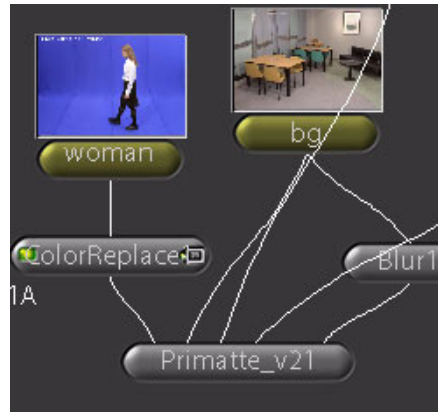
As another alternative for the spill suppression color, toggling *replaceMode* to use color uses the *Replace Color*. This fills a solid color to replace the removed blue spill color.

- 6 Make sure the composite is loaded in the Viewer.
- 7 In the *Primate* parameters, toggle *replaceMode* to use color.
- 8 Click the *Replace Color* Picker box, and scrub the color of the wall.

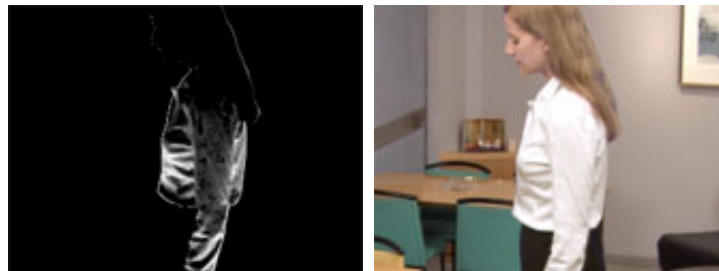
This can be helpful at times, but can also create a glowing edge around your image, which probably is not what you want.

- 9 Set the *replaceMode* parameter back to use image.

Another trick is to isolate the blue screen color with the *Color-ColorReplace* node and select a different (non-blue) color before it is keyed. For example, pick the blue spill areas on her shirt and use the *ColorReplace* node to replace it with a medium grey color.



The following example shows the blue on her shirt as the Source Color that is then replaced with color from the background image. The range parameters are all set to 0, with the falloff parameters all set to .2. Toggle the *ColorReplace* affectAlpha parameter and view the alpha channel of the *ColorReplace* node to test the area you are modifying. The blue screen should still be pure black.

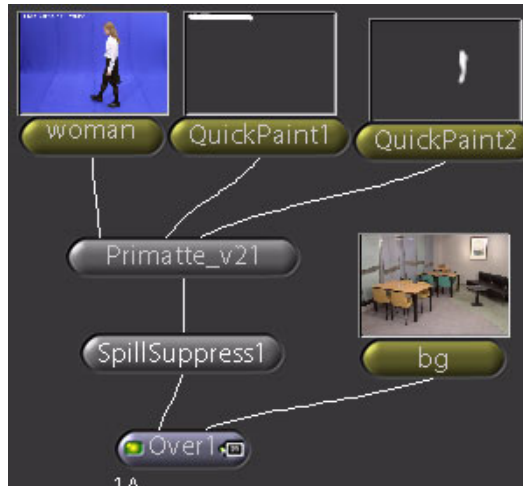


After using *ColorReplace*, delete some of the fine tuning operations in the *Primatte* parameters.

As you can see, there is no one way to use *Primatte*, or of pulling keys in general. We have explored several techniques to handle spill suppression here: Using spill sponge, changing the replaceImage with a blurred background image, changing the replaceImage with a *Monochrome* foreground image, and by changing the spill color before keying. Each technique has its advantages and disadvantages, which can only be appreciated with practice, as each key is unique. Isn't this fun?

Alternatives

Primatte has a great deal of flexibility—within the node itself, or by preceding the node with other nodes. You have the most flexibility, however, when you do not composite inside of the *Primatte* node. You can apply effects to the foreground element such as transforms or blurs, as well as spill suppression. In the following process tree, the *Primatte* output is set to alpha only, and preMultiply is enabled in the *Over* node. Because *Primatte* does not process the edge with color suppression operators, the edge stays much cleaner.



Note: Always supply the background image unless you are setting output to alpha only.

Although you can output *Primatte* with a pre-multiplied foreground without a background image, you should supply the background input if possible, as *Primatte* still calculates some of this information. If you do not supply the background image, black ringing occurs around the edges. If this is impractical, toggle over the *replaceMode* to use *Color* and supply an appropriate *Replace Color*.

This has a corollary: Save yourself the trouble and set your output to alpha only. No background image is needed; do your spill suppression with *HueCurves* or *SpillSuppress*. It also ensures your image is unpre-multiplied, making it ready to color correct.

Lesson Seven: Tracking and Stabilization

Tracking allows you to do two things. First, you can eliminate unwanted movement in a plate, such as the movement caused by film gate weave. This is known as stabilizing. Second, you can match a stable element to a moving element, such as swapping the heads of two actors. This is called matchmoving. You often use the two together. For example, to do a head replacement, you typically first stabilize the head you want to grab and then matchmove it to the body you want to place it on. Both involve tracking patterns in an image sequence from frame to frame. This can be done manually, but tends to get really tired, really fast, which is why you have tracking nodes to do it for you.

In this tutorial, use a *Stabilize* node to stabilize an image sequence and then convert the stabilization data to matchmove data. Next, track a sequence that changes perspective with a *MatchMove* node.

Tutorial Summary

- Tracking and Stabilizing Nodes
- Stabilizing an Image Sequence
- Converting Stabilization Data to MatchMove Data
- Using the *MatchMove* Node (with a four-point track)
- Position the Foreground Element
- Color Correct the Foreground Element

Tracking and Stabilizing Nodes

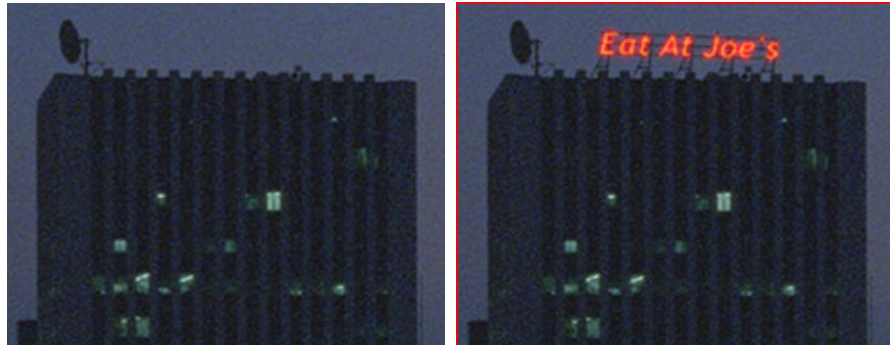
In Shake, three nodes track sequences of images:

- Transform–*Tracker*: This node generates unlimited tracking points, but does not apply movement to the element.
- Transform–*Stabilize*: This node generates one, two, or four tracking points, or uses points generated by other tracking nodes. It performs either stabilization or matchmoving.

- **Transform–MatchMove:** This node performs matchmoves and composites. It can also create its own one-, two-, or four-point tracks.

Stabilizing an Image Sequence

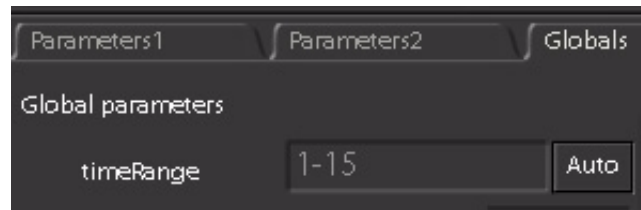
In the first section, stabilize a moving clip. Later, track the “Eat At Joe’s” sign onto the image.



To stabilize the building clip:

- 1 Add an **Image–FileIn** node and read in `doc/pix/stabilize/stab.@.jpg` (a 15-frame clip).
- 2 In the Global parameters, click **Auto** on the `timeRange` parameter.

Your script length is automatically set to the number of frames in the clip, 15 in this case.



- 3 Click **Home** on the Time Bar.



The Time Bar range is set from 1 to 15.



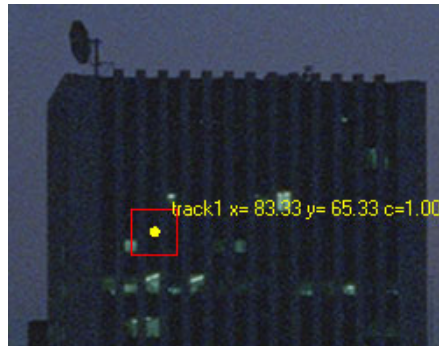
This does not affect your rendered frames, but helps you adjust your track points.

You have by now no doubt discovered the plate drifts to the left. This is a prime candidate for stabilization, which locks the image down.

- 4 In the Node View, select the *stab* node and add a Transform—*Stabilize* node.



A tracker appears in the Viewer.

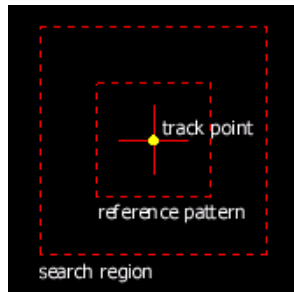




When the cursor is positioned over the tracker box, the tracker is highlighted. You can move a highlighted tracker box in the Viewer.

- 5 Position the tracker box on a high-contrast area, for example, the large window in the middle of the image.



The tracker is composed of the track point (the center), the reference pattern (the inner square), and the search region (the outer square). A pattern is isolated within the inner box, the reference pattern. The tracker looks for that pattern within the larger box called the search region. When a match is found, a track point (keyframe) is created.



- 6 Go to frame 1.
- 7 Click the Forward Track button  on the Viewer toolbar.
The 15 frames are tracked.
Note: You can also go to frame 15 and click the Backward Track button .
- 8 When the tracking is finished, activate applyTransform in the Parameters tab.

Note: When tracking large plates, enable `limitProcessing` in the *Stabilize* parameters to decrease image load times.



- 9 Press Play  or launch a flipbook  to test the stabilized plate.

The window stays in the same position for the 15 frames.

When the plate drifts, the *Stabilize* node moves it back to the original point. This eventually causes a black border to appear around the image. The simple (*lazy*) way to fix this is with a *Scale* node to adjust the plate size. This concatenates with the *Stabilize* node. This is good.

- 10 In the Node View, select the *Stabilize1* node and connect a Transform–*Scale* node.
- 11 Using the onscreen controls or the *Scale* node parameters, scale the image up slightly. Examine frame 15 to gauge the correct scale.



As this is the “Lazy Man’s Method,” and therefore appealing to composers across the world, it is not the ideal method. Instead, a clean plate should be constructed and placed behind the image. See “Lesson Nine: Creating a Clean Plate” on page 197.

Converting Stabilization Data to MatchMove Data

Use the same building clip to build a simple matchmove. In this case, remove the stabilization data from the building clip, invert the stabilization data, and apply the data to a second image.

To convert stabilization data to MatchMove data:

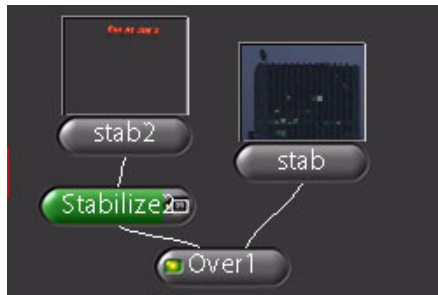
- 1 Add another *FileIn* node (do not delete your process tree from the previous section) and read in *doc/pix/stabilize/eatatjoes.iff*.

Note that because the image is saved as an .iff, the DOD data is retained.



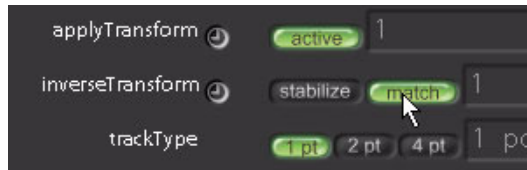
Since you have already tracked the background, you can attach the *Stabilize1* node to the *EatAtJoes* node, and then use a *Layer-Over* node to composite the two images together.

- 2 Extract the *Stabilize* node (select the node in the Node View and press **E**) from the *stab* (building) node.
- 3 Connect the extracted *Stabilize* node to the *eatatjoes* node.
- 4 Add a *Layer-Over* node to the *Stabilize* node.
- 5 Connect the *stab* (building) node to the second input (background) of the *Over* node.



The *eatatjoes* image drifts away from the background image. The *Stabilize* node is trying to stabilize the image—the opposite of what you want. To match the *eatatjoes* image to the movement of the building clip, invert the transform by setting the shrewdly-named *invertTransform* button to “match.”

- 6 In the *Stabilize* parameters, set *inverseTransform* to “match.”



The *eatatjoes* sign moves with the background plate.



As shown, you can use *Stabilize* for stabilization and matchmoving. Others lurking around the office prefer to use the *MatchMove* node, described in the following section. There is one gotcha with using *Stabilize* for matchmoving: You must turn off the transform and connect the *Stabilize* node to the background (tracked clip) in order to re-track.

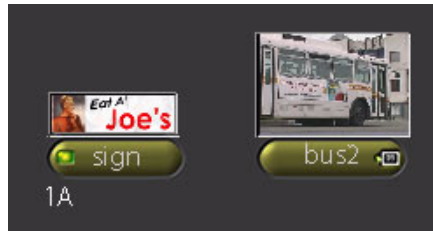
Note: When destabilizing, the *inverseTransform* toggle can also be used to reapply the same jitter that was removed. You may have a slight jitter due to gate weave. *Stabilize* the shot, apply all of your elements to the stabilized plate, and then copy and paste the original *Stabilize* node to the end of the tree. Then, set *inverseTransform* to *match*, and you arrive with the same natural-feeling jittery movement.

Using the MatchMove Node

The next example uses more complicated tracking information to match a background clip that changes perspective. The *MatchMove* node is used to match the movement of the background. In this example, read in a clip of a moving bus and an advertisement image, and apply the image to the side of the moving bus. This tutorial also discusses some solutions for common tracking problems.

To start the MatchMove script:

- 1 Choose File > New Script, or press **Command+N** / **Ctrl+N** to clear the existing nodes and begin a new script.
- 2 Add an Image–*FileIn* node and read in the 39-frame clip *bus2* from the *doc/pix/bus* directory, and read in *sign.jpg* from the *doc/pix/* directory. The sign image is 720 x 210 pixels, and the bus clip is 720 x 486 pixels.



(The original office of the Shake team appears in the clip. The building, not the bus.)

The following illustration shows the result of this lesson.

Before

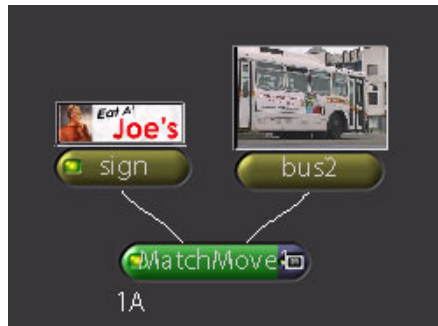
After



- 3 Examine the clip to identify good search patterns. (This should always be the first step.)
Luckily, the sign of the bus is visible during the entire clip, although there is some distortion due to perspective shifts.
- 4 In the Node View, select the *sign* node and add a Transform–*MatchMove* node.

Unlike other transform nodes, *MatchMove* allows you to composite within the node. The tracking is done on the second input (the background).

- 5 Connect the *bus2* node to the second input of the *MatchMove* node.



Although this project calls for a 4-point track (one for each corner), begin with only one tracker to keep things simple. Several intentional mistakes occur in this tutorial to help you better recognize problems with tracking your own footage. As they always do.

- 6 Go to frame 1.
- 7 Position the tracker in the lower-left corner of the sign on the side of the bus. Leave the tracker at the default size, and precisely position the tracker on the corner. Use the + / - keys to zoom in, with the cursor as an aiming device.

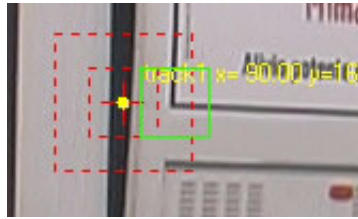
Note: Press **Home** to return the Viewer to the normal size.



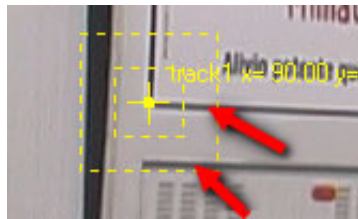
- 8 In the Viewer toolbar, click the Track Forward button.

Doh! The tracker advances a few frames and stops. The last number in the text next to the tracker box reads “c=.2” (approximately). This indicates the correlation, or accuracy of the track. A score of 1 means a perfect correlation with the reference pattern between frame 2 and frame 1 (frame 1 as the reference pattern). A score of 0 is amazingly bad. On frame 2, you have a correlation of .2. This is bad.

The search region is too small to encompass the movement in the clip—the bus moves too far to the right. In this illustration, the desired area, artificially marked in green, extends beyond the zone of the red search region.



If the search region is too large, you waste time processing. If the search region is too small, the tracker does not find successful matches. There is also a black line around the sign along the bottom of the pattern. There is a chance due to the perspective shift in later frames that the tracker will confuse the two lines. Adjust the search area with these in mind.



- 9 Go to frame 1.
- 10 Grab the corners of the search region and scale it outward, but shrink the bottom line up a bit. Also, scale down the reference pattern to better fit the corner.



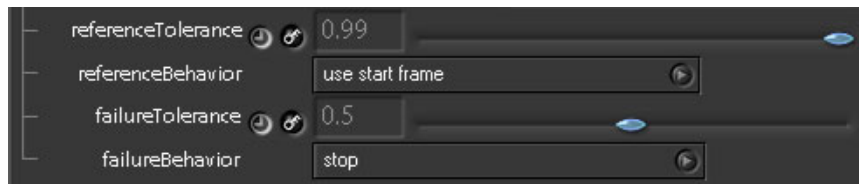
Note: When adjusting the tracker boxes, grab an edge to adjust just that edge. Grab a corner to scale the entire box. Grab the center to move everything. What is adjusted is highlighted in yellow.

- 11 Click Track Forward.
Any previous keyframes are written over.

Gee, everything was going fine until around frame 34, right? At that frame (results vary), the tracker spitefully decided that the words on the sign were the best match to the original reference pattern at frame 1.

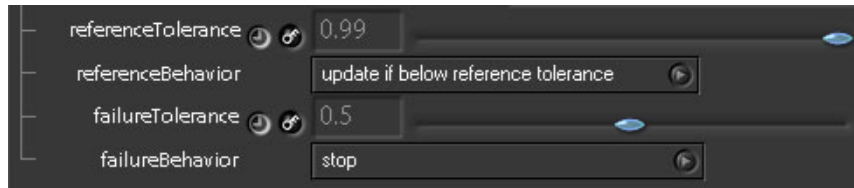


The tracker continually compares the new images back to frame 1 (or whatever frame you start the track). As the bus moves away, the sign gets smaller. At a certain point, the words match the original thickness of the sign border from frame 1 better than the sign border itself. This matching behavior is controlled by the `referenceBehavior` setting under the `tolerance` behavior subtree. It is set to use *start frame* by default—it compares the new samples to the original start frame (not necessarily frame 1 if you start at a later frame).



Two settings in the `referenceBehavior` can help with scaling changes in the image sequence. The settings are “update every frame,” that uses the sample of the previous frame for the reference pattern, and “update if below reference tolerance,” that checks the correlation of every frame. If the correlation falls below the `referenceTolerance` value (right above `referenceBehavior`), it uses the previous frame as the reference pattern. It continues to use that pattern until the correlation once again dips below the `referenceTolerance`. The second setting (update if below reference tolerance) is more accurate because you get inherent drift with update every frame as tiny errors accumulate.

- 12 In the *MatchMove* parameters, set the referenceBehavior under tolerance behavior to “update if below reference tolerance.”



- 13 Find a frame with a high correlation. Although you can track from frame 33 (the frame before the tracker planted), it is better to find a frame with a high correlation. The correlation is displayed in the yellow text next to the tracker box. Ideally, this is of course frame 1, but you may find a high correlation at approximately frame 14.

- 14 Click Track Forward.

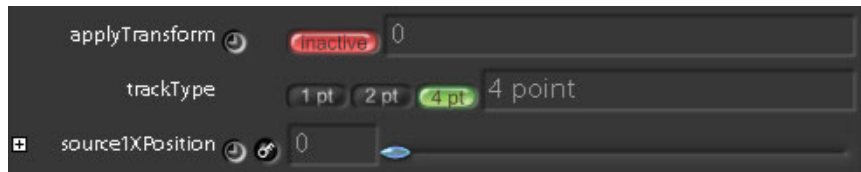
With the above settings, you should get a successful track.

Note: You can also manually enter track positions with the use of the Viewer Autokey. Note this is often the only way to get a track working.

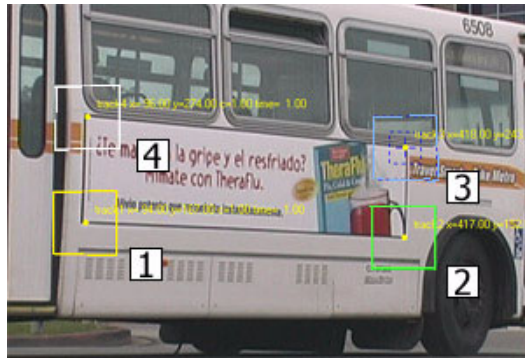
- 15 If the track fails again, repeat these steps, or manually reposition the tracker at the missed frame and restart the tracking.

To apply the other three trackers:

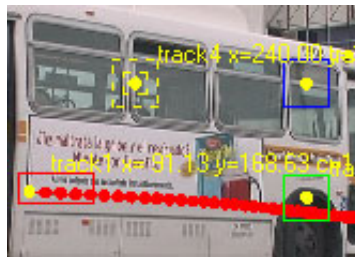
- 1 In the *MatchMove* parameters, set the trackType to 4 pt.



The three extra trackers, pre-positioned on the image, are activated. Each tracker is located in the relative corner where it should stay. For example, the upper-left corner of the foreground image is connected to track4 of the *MatchMove*. Shake uses Cartesian coordinates, so the trackers are ordered according to the following image.



Yes, you are right, this is different from some other prominent systems. Deal with it. The first track point, track1, is already set and its keyframes are visible.



Every time you start a track, all visible trackers analyze the image and create new keyframes. Therefore, turn off the visibility of track1. The four trackers are listed in the lower portion of the *MatchMove* parameters. You can click the track name text field (that is, where it says “track1”) to change the name. You can change the display color of a tracker by clicking on the Color Picker button. The V button toggles the visibility of a tracker.

- 2 Deactivate track 1 by toggling its visibility.

This ensures the track is not overwritten.



An active tracker is green in the tracker list, and yellow in the Viewer.



- 3 Go to frame 1.
- 4 Following the same principles of the first tracker, position and shape each of the remaining trackers on the remaining sign corners.



- 5 Click Track Forward to start the track.

When finished, the three points should be reasonably accurate. If not, use the visibility toggles to isolate problem trackers and repeat the above steps to correct your track.



Note: Manual tracking? Don't ever ever ever believe a product demonstration—tracking, like keying, is rarely straightforward and usually involves a lot of manual labor. Make sure Autokey is on and away you go. You frequently must adjust points by hand. Welcome to the exciting world of production! See what this magical world looks like as you drive home at 4:00 a.m.! You have been warned.

Position the Foreground Element

With the four tracks plugged into the *MatchMove*, you can now apply the foreground element.

In the *MatchMove* parameters, the *outputType* list contains several compositing options. Background is selected by default to allow you to immediately start tracking. To test the track, switch the *outputType* to a compositing operation.

- 1 In the *MatchMove* parameters, set the *outputType* to *Over*.



This produces a somewhat less-than-convincing composite.



As with *Stabilize*, you must activate the transformation.

- 2 Toggle applyTransform to active.



If an image appears brighter in an *Over* operation (in effect adding the two images), you should immediately suspect the absence of a mask in the foreground. Since the sign image is a JPEG and does not support alpha channels, you must add an alpha channel to the image.

- 3 In the *sign* node parameters (the *FileIn* node), enable autoAlpha.

The autoAlpha parameter examines the input image for an alpha channel. If no alpha channel is found, it sets the entire alpha channel to 1. If an alpha channel is present, it is left untouched by autoAlpha.



This is an improvement. The four corners of the foreground image are snapped to the four corners of the trackers. If the image is twisted around, you have done something such as put track 3 in track 4's corner. You can unravel that in the next step.

Adjust the Sign Position

The sign may not fit exactly into the background's sign limits. A little bit peeks out from behind.



To adjust the foreground sign position:

- 1 Go to frame 1.
- 2 Ensure the Viewer Autokey button is deactivated (unless you want to animate it, which you do not).
- 3 Click the BG/FG toggle button on the Viewer to show the foreground display.

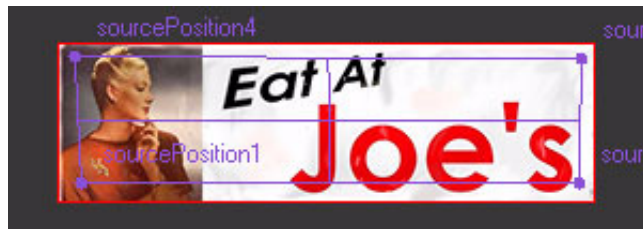


This displays 4 corner markers in the Viewer. These points correspond with what is matched to the four trackers. By default they are located in the four corners of the image.



- 4 Pull the corners in.

The corners represent the track location points. Therefore, bringing the points in expands the image size in the composition.



- 5 Toggle the Viewer to BG mode.

Note: You may need to switch your outputType back to *Over*. Occasionally this parameter is not reset properly.

The image is now slightly larger in the composition.



Color Correct the Foreground Element

Anybody who has visited Los Angeles knows that a clean bus is a mythical beast. As the final step, color correct the sign to better match it to the bus.

Before

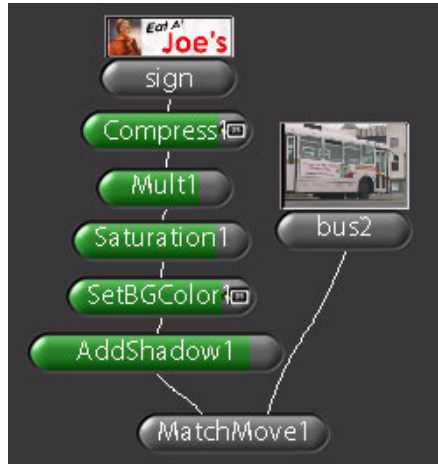
After



To build the color correction:

- 1 In the Node View, select the *sign* node.
- 2 Using the following illustration as a guide, insert these nodes from the Color tab: *Compress*, *Mult*, *Saturation*, and *SetBGColor*.
- 3 Add an Other-*AddShadow* node.

Note: You do not need a *MDiv/MMult* set because you are working on the full frame.



For the color correction, first adjust the whites and blacks with the *Compress* node. Because the sign has pure whites and blacks, and the bus has good representations of what should be white and black, *Compress* works well. It's strange actually, this is the third tutorial with *Compress*. Go figure.

- 4 In the *Compress1* parameters, click the Low Color Picker box.
- 5 Scrub color in the Viewer that represents the blackest part of the bus, for example, under the wheel well of the bus.

The blacks are raised to around 18 percent.

- 6 Click the High Color Picker box, and select a bright color from the side of the bus. As the bus changes its lighting levels as it turns, animate the High Color parameter.

To animate the Compress parameters:

- 1 Go to frame 1.
- 2 In the *Compress1* parameters, click the Autokey button in the High Color parameters.
- 3 Go to frame 39.

- 4 Scrub the same white area. Since Autokey is already activated, a keyframe is created.



The sign is a neutral grey color. Some warm color makes it stick out a bit and draws the viewer's attention (though not too much—it still has to be integrated into the scene). You can tint it yellow with *Compress1*, but you then must have the same modification for all of your keyframes, which is cumbersome. A better idea is to use the *Mult* node to tint the image. As *Compress* and *Mult* concatenate, there is no loss of color quality. This is good. Very good.

- 5 Load the *Mult1* parameters.
- 6 Using the Color Picker, tint the sign yellow.

The sign is now yellowish, but the red of “Joe’s” is too bright. Use the *Saturation* node to lower the overall saturation of the sign. You may need to boost the *Mult1* node to compensate.

- 7 In the *Saturation1* parameters, lower the saturation value.

Note: *Saturation* does not concatenate with other nodes.

The three nodes (*Compress1*, *Mult1*, and *Saturation1*) demonstrate an advantage of working with nodes and concatenation. Because *Compress1* is animated, it is awkward to apply adjustments to the overall color for the entire clip length within the *Compress* node. By breaking these steps into separate nodes, you do not have to adjust each keyframe.

Next, because the *Compress* node raised the black levels in infinite space, the *Color-SetBGColor* is used to set the area outside of the image area to a specific color (black by default). If the sign has a soft mask, you must use a *MDiv/MMult* pair.

Test the effect of the *SetBGColor1* node:

- In the Node View, select the *SetBGColor1* node and press **I** to ignore it.
- Press **I** again to activate the *SetBGColor1* node.

The sign is color balanced in the scene, but the shadow, created with the *AddShadow* node, needs to be adjusted. Without this node, the sign has no feeling of depth.

To adjust the sign shadow:

- 1 In the *AddShadow1* parameters, set the xPixels and yPixels to approximately -5.
- 2 Set fuzziness to approximately 35.

- 3 Set shadowOpacity to approximately .7.

No Shadow



With AddShadow



Note: For additional fine tuning, look at the script *doc/pix/bus_track2.shk*. This script adds some color correction and masking to take care of the reflections that go across the side of the bus from frames 36 to 40. Although interesting, it does not seem necessary for world peace to include it in the tutorial.

Stabilize as an Alternative to MatchMove

You may have noticed that tuning the corners of the sign using the FG/BG Viewer controls was less than ideal. Some might use the word “maddening and suicide-inducing,” as there is no interactive feedback. An added drawback is onscreen controls from upstream are not properly adjusted by the *MatchMove* node. This is one advantage that the *Stabilize* node has over the *MatchMove* node. As demonstrated in the Eat At Joe’s tutorial, you can use *Stabilize* instead of *MatchMove* to create matchmoves. This gives you a more intuitive control mechanism, but requires more plumbing in the Node View. Note that for this exercise, you are copying the tracks from *MatchMove1*, but you can generate the tracks with a *Stabilize* node.

To use Stabilize for creating a matchmove:

- 1 Attach *Stabilize* to the node you want to track (in this example, the bus image), and generate your tracks.
- 2 Extract the node (select the *Stabilize* node and press **E**), and attach it to the image you want to transform (in this example, the sign).
- 3 In the *Stabilize* parameters, activate the transformation with `applyTransform` and set `inverseTransform` to *match*.
- 4 Composite the *Stabilize* over the background with an *Over* node.

- 5 Optional Step (not necessary for this tutorial): Insert a Transform–*Viewport* node above the *Stabilize* node to adjust the frame around what you want to track, or a *CornerPin* and invert its transform. Either of these readjusts the corner positions to match the image corners in the next step. This is only when you are adjusting an image area that does not match the frame borders.
- 6 Insert a *CornerPin* above the *Stabilize* node. This is used to position the foreground over the background.

This technique has several advantages:

- More flexibility in your compositing
- Better control over transform concatenation of the foreground
- Better control over premultiplication of the foreground
- Accurate pass-through of upstream onscreen controls
- Intuitive control of foreground positioning
- Access to the rotation and scaling values for 2-point transforms

To switch the tracking from the MatchMove node to a Stabilize node:

- 1 In the Node View, select *AddShadow1*.
- 2 **Shift**-click Transform–*CornerPin*.
The Shift modifier sends CornerPin off as a new branch.
- 3 Connect a Transform–*Stabilize* node to the *CornerPin*. Normally, you attach the *Stabilize* to the *bus2* node to start tracking, but you already have your tracks in *MatchMove1*.
- 4 Connect an *Over* node to *Stabilize1*.

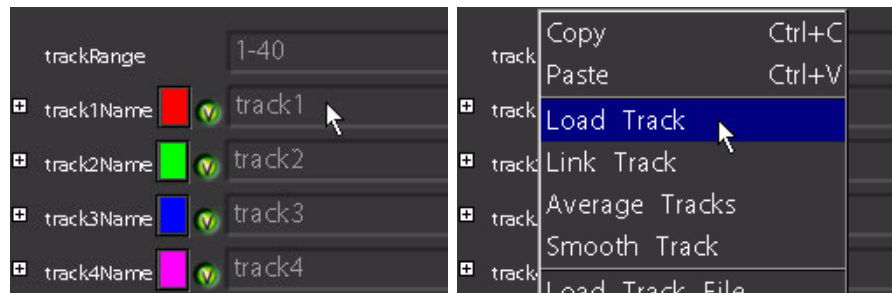
- 5 Connect *bus2* to the second input of *Over1*.



Next, copy the tracks from *MatchMove1* to *Stabilize1*.

To copy the tracks into Stabilize:

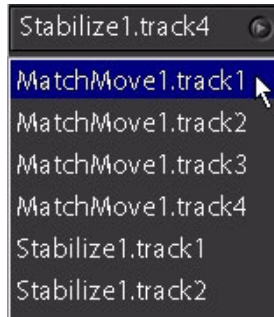
- 1 Right-click the trackNames and select *Load Track*:



A pop-up window appears that lists all tracks.

- 2 Select *MatchMove1.track1* and click OK.

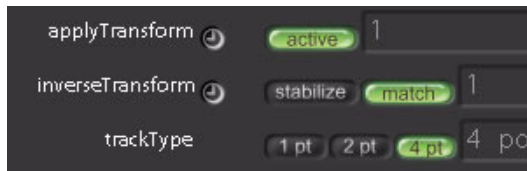
track1 is copied from *MatchMove1* to *Stabilize1*.



- 3 Repeat the process for track2 through track4.
- 4 Activate the transformation with `applyTransform`.
- 5 Set `inverseTransform` of the Stabilize node to *match*.



You are currently only applying the motion to one point, so toggle your `trackType` to *4 pt*.



Things don't quite line up in the Viewer. Oops.

Frame 1



Frame 30



MatchMove has controls for adjusting the foreground position. *Stabilize*, by default, assumes that frame 1 has the relative position that you want. The selected frame can be modified with the *referenceFrame* parameter, but it still doesn't modify the relative position. You must therefore rely on external nodes to adjust this, in this case, the *CornerPin* node.

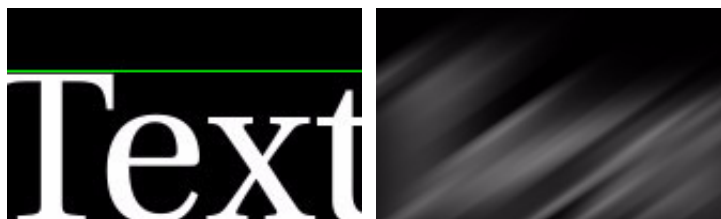
- 6 Make sure the Viewer Autokey button is deactivated.
- 7 Load the *CornerPin1* parameters.
- 8 Move the corners of the *CornerPin* to match the side of the bus.



The controls are now completely intuitive, as well as interactive. Just like the DMV.

Lesson Eight: Making a Macro

This tutorial makes a spiffy macro for a motion effect by blurring an image at any angle.



Tutorial Summary

- What Is a Macro?
- Creating a Handmade Macro
- Saving and Testing the Macro
- Adding a Button to the Interface
- How to Set Slider Ranges
- Creating Macros With MacroMaker
- Creating Sliders in MacroMaker

What Is a Macro?

So, what is a macro? Macros are commonly-used nodes that are combined into new nodes (that you can use to thwart your enemies). You control what parameters are exposed, and hide the ones you do not need. Macros are extremely powerful ways of modifying Shake to suit your own needs. Whereas you can group nodes together in the Node View to also hide multiple parameters and nodes, macros become entirely new functions that are premade for the user.

Note: There is an online version of this tutorial to allow easy copy and paste of the scripting sections.

The following images demonstrate the creation of a macro. The first image has four nodes—the *Text* node and the three nodes that combine to create the blur effect you want to create. Following a long, arduous, painful, tedious, frustrating (like this sentence) process, the second image shows the completed macro attached to an icon (*RotateBlur* icon). The third illustration shows the completed macro, *RotateBlur*, applied in the tree—it looks like any other node. The last image demonstrates that only the angle and the blur parameters are exposed from the two *Rotate* nodes and the *Blur* node that comprise the macro.



There is only one path to ultimate enlightenment. Don't let anybody say it's some silly tripe like being nice to puppies, or calling your mother once a week (although, do not let this discourage you from trying that, too). Rather, it is achieved by dipping into the Shake scripting language with a text editor. This is detailed in the first three sections of this tutorial. What fun! Before your hand leaps up to find another tutorial, the last part shows you how to interactively do the same thing inside Shake. There is a catch, however. To modify and enhance a macro, you must rely on the information found in the first parts of the tutorial. No rest for the wicked.

Creating a Handmade Macro

The first step in a macro is to build the effect using normal nodes.

To build the node tree:

- 1 Create an Image-*Text* node.
- 2 Attach a Transform-*Rotate* node.
- 3 Attach a Filter-*Blur* node.
- 4 Attach a second *Rotate* node.



The algorithm works by rotating the image, blurring the image on the X axis, and rotating back to the start position. Therefore, if *Rotate1* has an angle of 45 degrees, *Rotate2* must have an angle of -45 degrees. The best way to do this is by entering an expression to link *Rotate1* to *Rotate2*.

To link the *Rotate1* and *Rotate2* angle parameters:

- 1 Load the *Rotate1* parameters.
- 2 In the angle parameter, enter the following expression:

-Rotate2.angle

This links the angle of the *Rotate1* node to read the angle of the *Rotate2* node and invert it. As you modify *Rotate2*, *Rotate1* automatically updates.

Case sensitivity is important. In Shake, each word in a node name has capitalized letters. For example, *QuickPaint* has an uppercase Q and P. The parameters follow the same rule, except the first letter is lowercase. For example, in the parameter framesPerSecond only the P and the S are capitalized. As you develop macros, it is good form to follow these guides, as you never have to wonder what letters are capitalized.

To set the Blur parameters:

- 1 Load the *Blur1* node parameters.
- 2 Set xPixels to 200.
- 3 Set yPixels to 0.
- 4 Toggle the spread parameters to Outside Frame (1).

To test the effect:

- 1 Load the *Rotate2* parameters.
- 2 Adjust the angle parameter. The blurring rotates, but the text stays in the same spot. If the *Rotate2* angle parameter is 67 degrees, then the *Rotate1* angle is -67.

Warning Do not modify *Rotate1*, because that breaks the link to *Rotate2*.

If it is not working for some reason, don't sweat it—delete your *Rotate* and *Blur* nodes, copy the following text (**Command+C** / **Ctrl+C**), and paste it into the Node View (**Command+V** / **Ctrl+V**):

```
Rotate1 = Rotate(0, -Rotate2.angle, 1, width/2, height/2, 0, 0.5, 0);
Blur1 = Blur(Rotate1, 200, 0, 1, "gauss", xFilter, "rgba");
Rotate2 = Rotate(Blur1, -36.9218, 1, width/2, height/2, 0, 0.5, 0);
```

Because Shake is a compiler, you can copy nodes from text and paste the text into the interface. Ain't it swell?

Save the script for later use:

- Choose File > Save As and save the script as *rotateblur_tree.shk*.

You are now ready to start creating the macro. First, create a User Directory that Shake looks in when launched. This directory is in your \$HOME (for example, *Users/jobn*) directory. You place macros, preference settings, and interface modifications in the User Directory. In the following steps, add a series of subdirectories to the User Directory in the Terminal application.

To make the macro:

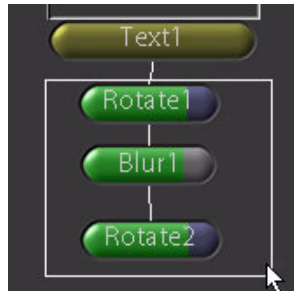
- 1 Open the Terminal, found in the */Applications/Utilities* directory.
- 2 Type:

```
mkdir -p $HOME/nreal/include/startup/ui
```

```
mkdir -p $HOME/nreal/icons
```

Note: You can add files to other directories as well using an environment variable. For more information on setting environment variables, see “Environment Variables for Shake” in Chapter 17 of the *Shake 3 Reference Guide*.

- 3 Launch a text editor (TextEdit, vi, emacs, etc.).
- 4 In Shake, drag-select the *Rotate1*, *Blur1*, and *Rotate2* nodes in the process tree.



- 5 Press **Command+C / Ctrl+C** to copy the nodes.
Note: You can also select Copy with the right-mouse (context) menu in the Node View.
- 6 In the text editor, paste the nodes (**Command+V / Ctrl+V**).

The Shake source code is pasted into the editor, and it looks something like the following:

```
Rotat1 = Rotate(0, -Rotate2.angle, 1, width/2, height/2, 0, .5, 0);
Blur1 = Blur(Rotat1, 200, 0, 1, "gauss", xFilter, "rgba");
Rotate2 = Rotate(Blur1, 42.61728, 1, width/2, height/2, 0, 0.5, 0);
```

```
// User Interface settings
```

```
SetKey(
  "nodeView.Blur1.x", "290",
  "nodeView.Blur1.y", "227",
  "nodeView.Rotat1.x", "290",
  "nodeView.Rotat1.y", "263",
  "nodeView.Rotate2.x", "290",
  "nodeView.Rotate2.y", "191"
);
```

Copying Text Into the Node View

If you copy the first two lines in the text editor, that is:

```
Rotate1 = Rotate(0, -Rotate2.angle, 1, width/2, height/2, 0, .5, 0);  
Blur1 = Blur(Rotate1, 200, 0, 1, "gauss", xFilter, "rgba");
```

you can paste it back into the Node View with **Command+V** / **Ctrl+V**.

This demonstrates that you can copy between script and interface. In our humble view, this is mighty swell.

If you already know C programming, you may want to jump down to the finished macro; otherwise, keep reading.

The text in your editor is raw script, meaning it is not formatted as a macro. If you paste it back in, you get three more nodes. You need to format the text as a single function—a macro.

- 7 Delete the layout information. This is everything including and below `User Interface Settings`. These lines tell Shake where to place the nodes if you paste them back in again. You do not need the lines, so remove them. You are left with this:

```
Rotate1 = Rotate(0, -Rotate2.angle, 1, width/2, height/2, 0, .5, 0);  
Blur1 = Blur(Rotate1, 200, 0, 1, "gauss", xFilter, "rgba");  
Rotate2 = Rotate(Blur1, 42.61728, 1, width/2, height/2, 0, 0.5, 0);
```

The left side of the argument supplies a variable name. In the above example, there are three variables: `Rotate1`, `Blur1`, and `Rotate2`. These are assigned to Shake functions, the *Rotate* and the *Blur* functions. (Inside Shake, these are referred to as “nodes.” Inside a script, they are referred to as “functions,” as you can have other types of functions.) Next are the values for the functions. The first argument is generally the incoming image, so you can see the incoming image for `Blur1` is the result of the first *Rotate* function, `Rotate1`. `Rotate1` has a value of 0 as the incoming image—there is no expected input. If you look these functions up in the documentation, you find the scripting format for each function.

If you change the variable names `Rotate1`, `Blur1`, and `Rotate2`, you must modify every reference to them.

The following is used as an example only (it is not a step in the tutorial):

```
Marshall = Rotate(0, -Holly.angle,1, width/2, height/2, 0, .5, 0);  
Will = Blur(Marshall, 200, 0, 1, "gauss", xFilter, "rgba");  
Holly = Rotate(Will, 42.61728, 1, width/2, height/2, 0, 0.5, 0);
```

Notice how the strings “gauss” and “rgba” are in quotation marks because they are strings, meaning letter data rather than numeric data. Compare these to what you see in the interface parameters tab to get an idea of how Shake saves its parameters in a script.

Starting the Macro Format

Next, add some formatting. You must declare the type of the macro—if the macro spits out an image, an integer (int), a number with decimal place (float), or a word (string). Because this macro modifies an image, it needs to return an image. After that, add the macro name. In this case, name the macro *RotateBlur*, capitalizing each word for consistency. Also, include two sets of parentheses. Incoming variables are placed in the first set, and the second set {the curly brackets} surrounds the body of the macro.

- 8 Add the formatting for the macro body and declare the macro type and name:

```
image RotateBlur( )  
{  
Rotate1 = Rotate(0, -Rotate2.angle, 1, width/2, height/2, 0, 0.5, 0);  
Blur1 = Blur(Rotate1, 200, 0, 1, "gauss", xFilter, "rgba");  
Rotate2 = Rotate(Blur1, 42.61728, 1, width/2, height/2, 0, 0.5, 0);  
}
```

Next, you need to declare the information that is passed into the macro. You can potentially change all parameters—the rotate angle, center of rotation, motion blur parameters, blur filter, and so on. Practically speaking, only the rotate angle and the blur amount need to be set, giving you two sliders. You must also declare that an image is passed into the macro to be modified. If you do not have an image, it assumes that you are somehow creating a new image, such as a *Ramp* or *ColorWheel*. Each time you add an image input variable, an input (connection point) is created on top of the node in the Node View (so you can plug in an image). Each time you add a float, string, or int, a new slider or text field is created in the Parameter View. When you add these variables, declare the type of variable (again, either image, int, float, or string), and then the name. The names of the parameters should follow that pesky rule mentioned earlier—the first letter in the parameter name is lowercase.

- 9 Add the input variables, which later become sliders or image inputs on the node:

```
image RotateBlur(  
image input,  
float angle,  
float blur  
)  
{  
Rotate1 = Rotate(0, -Rotate2.angle, 1, width/2, height/2, 0, 0.5, 0);  
Blur1 = Blur(Rotate1, 200, 0, 1, "gauss", xFilter, "rgba");  
Rotate2 = Rotate(Blur1, 42.61728, 1, width/2, height/2, 0, 0.5, 0);  
}
```

- 10 Plug the variables into the right spot inside the macro body. Instead of the number or word copied from Shake, use the variable name. These values are later modified with sliders.

Substitute the variables in the macro body:

```
image RotateBlur(  
image input,  
float angle,  
float blur  
)  
{  
Rotate1 = Rotate(input, -Rotate2.angle, 1, width/2, height/2, 0, .5, 0);  
Blur1 = Blur(Rotate1, blur, 0, 1, "gauss", xFilter, "rgba");  
Rotate2 = Rotate(Blur1, angle, 1, width/2, height/2, 0, 0.5, 0);  
}
```

- 11 Indicate what you want to spit out of the macro with the return line. This is the final image (or float, int, string) that you want to extract out of the macro function. Remember the semicolon at the end of the line—in this case, `return Rotate2`, as it has the result you need.

Add the return line:

```
image RotateBlur(  
image input,  
float angle,  
float blur  
)  
{  
Rotate1 = Rotate(input, -Rotate2.angle, 1, width/2, height/2, 0, 0.5, 0);  
Blur1 = Blur(Rotate1, blur, 0, 1, "gauss", xFilter, "rgba");  
Rotate2 = Rotate(Blur1, angle, 1, width/2, height/2, 0, 0.5, 0);  
return Rotate2;  
}
```

- 12 The following is the final macro. As a final touch, you should add default values for any parameter, so it launches without having to add values in the command line.

Add the default values:

Note: Take special care with the commas.

```
image RotateBlur(  
image input,  
float angle = 45,  
float blur = 150  
)  
{
```

```

Rotate1 = Rotate(input, -Rotate2.angle, 1, width/2, height/2, 0, .5, 0);
Blur1 = Blur(Rotate1, blur, 0, 1, "gauss", xFilter, "rgba");
Rotate2 = Rotate(Blur1, angle, 1, width/2, height/2, 0, 0.5, 0);
return Rotate2;
}

```

Saving and Testing the Macro

When this is done, save the file in your *startup* directory. Make sure the filename has a .h file extension. Some text editors automatically append an extension. This is bad. There is no correlation between the filename and the macro name.

Save the macro:

- Save the file in your `$HOME/nreal/include/startup` directory:
`/User/ImASpecialPerson/nreal/include/startup/my_macro.h`

Test the macro:

- Go to the command line.

You might be saying to yourself right now, “Hey, why am I testing this in a Terminal?” (This is a good question.) All you have done so far is create the macro—you have not plugged it into the interface. You have not written anything that tells Shake to build this function into the interface. You do that in a moment.

- Test the function by typing in the Terminal:

```
shake -help rotate
```

If everything worked properly, you should get a listing of all functions that contain the word “rotate,” including the new function, *RotateBlur*.

Note: The command line is the only place where capitalization does not matter, because it saves you the effort of having to press **Shift** all the time.

```
-irotate [density] [steps] [stepBlend]
```

```
[controlChannel] [invert]
```

```
-rotate [angle] [aspectRatio] [xCenter] [yCenter] [motionBlur]
```

```
[shutterTiming] [shutterOffset]
```

```
-rotateblur [angle] [blur]
```

If you do not get this help message, it returns an error that tells you what went wrong.

Note: When working in the Terminal, error messages are printed in that window. When working in the interface, error messages are printed in the terminal that launched Shake. If you launched Shake with the Dock, the messages appear in the Console. The errors generally detail where you have a mistake or if you have a variable that is incorrectly named.

If you cannot tell what is wrong from the error message, check the following list of possible problems:

- Match the capitalization and spelling of the variable names to where they are plugged in. For example, did you declare the variable *blur* but use *bluramount* by mistake later on?
- Use normal parentheses for the variable declarations.
- Use curly brackets for the body of the macro.
- Include a semicolon at the end of the return statement.
- Did you include an extra comma at the end of your variables on the last line of the variables? If you did, remove it.
- Is the file saved in your *startup* directory?
- Make sure the saved file has a .h extension on the name, and it is not named *nreal.b* or *nrui.b*.

If you are still unable to figure it out, copy the finished macro from above into your *startup* directory and save it as *my_macro.b*.

Now try it on an image. If you do not have an image, go to *Shake3/doc/pix*. If you are using these images, try:

```
shake traffic_bg.jpg -rotateblur
```

```
shake traffic_bg.jpg -rotateblur 100 300
```

```
shake traffic_bg.jpg -rotateblur "Linear(1,0@1,360@11)" 200 -t 1-10
```

```
shake traffic_bg.jpg -rotateblur "Linear(1,0@1,360@11)" 200 -gui
```

Terminal Shortcuts for Repeating Commands

If you press the **Up Arrow** key, it repeats the last command typed in the Terminal. Press the **Up Arrow** again to list the previous command on the list, and so on. Use the **Left Arrow** and **Right Arrow** keys to edit the command.

The “Linear...” used for the angle of the blur is an animation curve, using a Linear format. The first “1” indicates the curve is extended into infinity. The second set of numbers indicates you have a value of 0 at frame 1, and then a value of 360 at frame 11.

The last command launches the tree into the interface. You can see your new *RotateBlur* node in action. However, you still cannot actually click on anything to create a second node. For that, you need to create a button and attach an icon.

Adding a Button to the Interface

This section shows you how to create a button in the interface and attach a node to the button. The icons for the tabs have two qualities:

- The icon size is 75 x 40 pixels.
- The icons are saved in your *nreal/icons* directory as *TabName.Name.nri*.

To make the icon with the RotateBlur function:

- 1 In the Terminal, type:


```
shake -addtext RotateBlur -rotateblur -gui
```

The text is launched in the interface with your *RotateBlur* attached.

- 2 Attach a Transform—Fit node to the *RotateBlur1* node in the Node View.
- 3 Set the xSize parameter to 75.
- 4 Set the ySize parameter to 40.

The result is an image that is 75 x 40 pixels.

To save the icon file:

- 1 Attach an Image—FileOut node to the *Fit* node.
- 2 In the Save image or sequence browser, use the Directories pull-down menu to navigate to your *\$HOME/nreal* directory (that is, */User/ImASpecialHumanBeing/nreal*, for example).
- 3 Navigate to the *icons* directory. (You should have created this directory in the first step of the tutorial. If not, use the Create Folder button  to make this directory.)
- 4 Save the image as *<TabName>. <MacroName>.nri*. (The *.nri* extension stands for Nothing Real Icon.) To place the *RotateBlur* function in the *Filter* tab, enter your filename as:
Filter.RotateBlur.nri
- 5 Choose Render > Render FileOut Nodes.
- 6 In the Render Parameters window, set renderFileOuts to All.
- 7 Click OK, and a 320 x 243 thumbnail appears.

(Yes, correct, it is rather bizarre to have a thumbnail that is larger than the actual image being rendered.)

You now have the two necessary elements: The macro and the icon. You now need to write the code to place both in the interface.

- 8 Quit Shake. (You must restart Shake to write UI code.)
- 9 Go to the *\$HOME/nreal/include/startup/ui* directory, and launch a text editor in the directory (or launch TextEdit from the Dock).

- 10 This directory contains all instructions to modify the interface. Paste (or type if you are reading the printed version—again, you are encouraged to use the online version) the following into an editor:

```
nuiPushToolBox("Filter");
nuiToolBoxItem("RotateBlur", RotateBlur(0,0,0));
nuiPopToolBox();

//These set the slider ranges for the blur parameter
//nuiDefSlider("RotateBlur.blur", 0, 400 );
```

Note: “You’ve got to be kidding me.” You rarely type code in—nobody remembers this stuff. Typically, it is copied from the *nreal.b* and *nrui.b* files, or from this documentation, then pasted in and modified. Now you know why there is online help.

The first line indicates the tab where the macro is placed, the Filter tab in this case. If the tab does not exist yet (for example, the MySwankFunctions tab), Shake creates it for you. The second line adds the function into the tab. The first occurrence of “RotateBlur” calls the icon file that you just created.

Important Shake assumes you have matched the tab name with the filename, so if you actually did place this into the MySwankFunctions tab, you have to rename the icon file on your disk to *MySwankFunctions.RotateBlur.nri*.

Here is how you can create a node without an icon:

If you do not have an icon, create a plain button with text on the button by placing an @ sign before the word:

```
nuiToolBoxItem("@OopsNoIcon",RotateBlur(0,0,0));
```



As this example shows, the button name can differ from its actual function.

The next part of the code, `RotateBlur(0,0,0)`, is the function call, followed by its default values. Because Shake does not know what image it is attached to (until you actually press it), place a 0 as the first (image) variable. The second and third zeroes set the angle and blur parameters to 0. The third line closes up the tab.

- 11 Save the text file as *rotateblur_ui.b* in your `<UserDirectory>/nreal/include/startup/ui` directory.

12 Launch Shake.

In the Filter tab, the new *RotateBlur* button appears.



If there is a problem, check the following list:

- If you only see the “Missing Artwork” icon, it means Shake cannot find the icon for some reason. Make sure the icon is named `<TabName>.<IconName>.nri` and is saved in the `<UserDirectory>/nreal/icons` directory. Make sure spelling and capitalization of the filename and the call to the file match.
- If there is nothing at all, you have made an error in your ui file. Check the error messages for an indication.
- If the icon is there, but nothing happens when you click the button, check the spelling, capitalization, and number of arguments of the call to the *RotateBlur* function. Make sure there are no extra commas.
- Make sure you saved your ui file in the *ui* directory, and that it ends with a `.h` extension.

If the function worked, walk with confidence and pride in the face of your enemies as you display your compositing prowess.

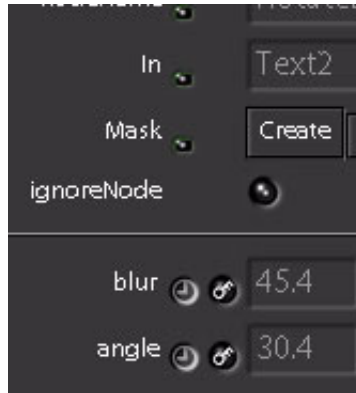
How to Set Slider Ranges

This section discusses how to test the sliders created in your macro.

To test the sliders:

- 1 Add the new *RotateBlur* node.

- 2 Move the angle and blur sliders.



Notice that the angle slider is from -360 to 360, but the blur slider is from 0 to 1, forcing you to use the Virtual Sliders (**Ctrl**-drag in the text field) to go beyond 1. Set your own slider range with a bit of code already included in the example from above. On the last line of your ui file, and only the last line, remove the comments, which are the `//`. When commented out, it is not read when Shake starts. By removing the comments, you reactivate the line.

To uncomment the slider definition:

- 1 Remove the `//` from the last line in your `ui/rotateblur_ui.b` file:

```
nuiPushToolBox("Filter");
nuiToolBoxItem("RotateBlur", RotateBlur(0,0,0));
nuiPopToolBox();

//These set the slider ranges for the blur parameter
nuiDefSlider("RotateBlur.blur", 0, 400 );
```

This sets the slider range from 0 to 400 for the blur parameter of the `RotateBlur` function. Spelling and capitalization are important, which is why that tiresome rule explained earlier starts to make sense.

You do not have to set a range for angle, because it is already set between -360 and 360 in the default Shake setup files. To change the angle slider range, duplicate the blur line and modify it to say `angle`. This example sets it from 0 to 180:

```
//These set the slider ranges for the blur parameter
nuiDefSlider("RotateBlur.blur", 0, 400 );
nuiDefSlider("RotateBlur.angle", 0, 180 );
```

- 2 Save your file, and launch Shake again.

The sliders are now set to 0 to 400 and 0 to 180.

For more information on customizing parameters, see Chapter 17, “Customizing Shake,” in the *Shake 3 Reference Guide*.

Creating Macros With MacroMaker

Not the most fun way to do things, but it beats banging on rocks next to a road in the Georgia heat. This next section of the tutorial duplicates your labor by using the MacroMaker. It is a fast way of creating reusable macros, but it is important to understand the files it creates so you can modify the files. Creating the macro by hand (above) helps demystify the process that the MacroMaker follows.

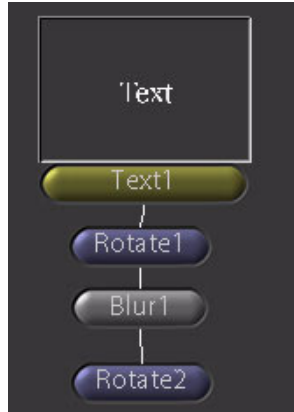
To create a macro with the MacroMaker:

- 1 Open Shake, and, if you followed the above parts of the tutorial, load the script you created called *rotateblur_tree.sbk*.

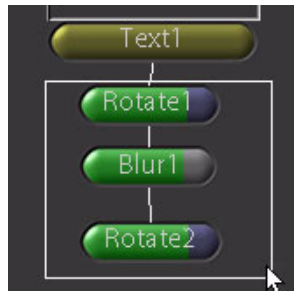
Note: If you do not have this script, paste in the original tree. If reading this tutorial in the online PDFs, copy the following text and press **Command+V** / **Ctrl+V** to paste it into the Node View:

```
Text1 = Text(300, 200, 1, "Text", "Courier", 100,
    xFontScale, 1,
    width/2, height/2, 0, 2, 2,
    1, 1, 1, 1, 0, 0, 0, 45
);
Rotate1 = Rotate(Text1, -Rotate2.angle, 1,
    width/2, height/2, 0, 0.5, 0
);
Blur1 = Blur(Rotate1, 181.9876, 0, 1, "gauss", xFilter, "rgba");
Rotate2 = Rotate(Blur1, 42.64842, 1,
    width/2, height/2, 0, 0.5, 0
);
```

The node tree appears in the Node View.



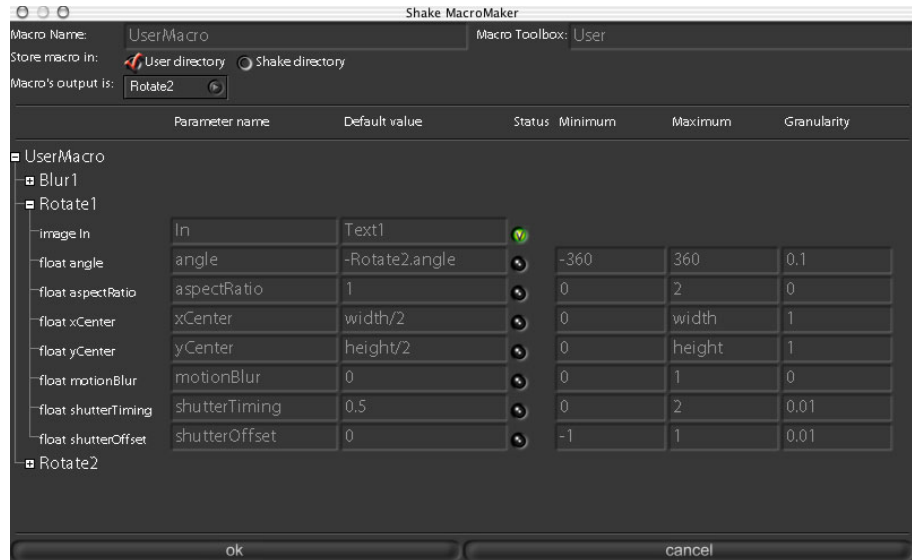
- 2 In the Node View, select the *Rotate1*, *Blur1*, and *Rotate2* nodes that build the macro.



- 3 With the cursor in the Node View, press **Shift+M**.

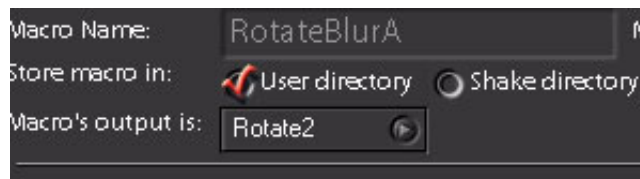
Note: You can also right-click in the Node View and select Macro > Make Macro from the context menu.

The Shake MacroMaker is launched.



The top portion of the Shake MacroMaker window contains text fields for the name of the macro, and specifies in what tab the macro is placed. By default, the macro name is UserMacro, and creates a new tab (“macro toolbox”) called User. You can change the name of the macro and its tab.

- 4 Name the macro *RotateBlur*. Shake notifies you this name is already used because you just manually created a macro called *RotateBlur*, and therefore appends an A to the end of the name.



- 5 In the Macro Toolbox text field, enter *Filter*.
- 6 Leave “Store macro in” set to the default User directory.

The “Store macro in” setting indicates where the macro is saved:

- The User Directory is $\$HOME/nreal/include/startup$. Macros stored here are only available to you on the local machine.

- Global is the `<ShakeDirectory>/include/startup` directory. The macro is available to all users that open Shake using that specific binary in that specific directory. You may have permissions problems writing to this directory.

The output tells you what node is spit out of the new macro. Shake usually makes a good guess, but you may need to explicitly specify the node if you have multiple branches selected. In this example, *Rotate2* is the proper output. This is the equivalent to the line:

```
return Rotate2;
```

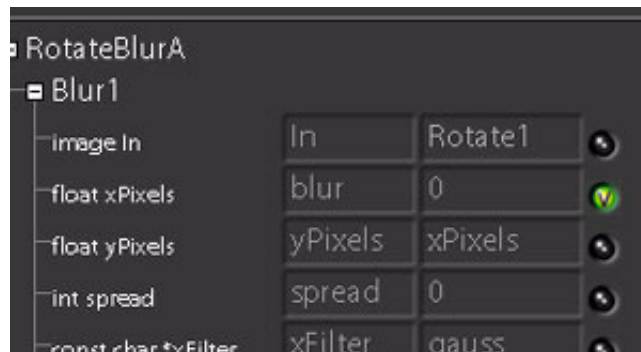
that you used in the first macro.

Creating Sliders in MacroMaker

The next area of the MacroMaker window contains the list of parameters you can expose to the user, with the name of the parameter, the default values, and the slider range.

To expose the Blur1 xPixels parameter and the Rotate2 angle parameter:

- 1 In the MacroMaker window, open the *Blur1* subtree.
- 2 Change the name of the float xPixels variable to “blur.” (This does not conflict with the *Blur* function because of capitalization.)
- 3 Set the default float xPixels value to 0.
- 4 Make sure the float xPixels Visibility light is on. Each visible parameter appears as an input or a slider.



- 5 In the *Rotate2* (not *Rotate1*) subtree, change the name of the float angle variable to “angle.”
- 6 Set the default float angle value to 0.

- 7 Make sure the float angle Visibility light is on.



The *Rotate1* “image In” parameter is activated automatically. This is because an image is fed into it, but is not accounted for by the nodes that you selected. This can cause problems (though not in this case), as all unattached image inputs are activated.

- 8 Click OK.

A new button called *RotateBlurA* appears in the Filter tab, identical to the macro you previously created.



This process creates two files, which are similar to the files you previously created by hand:

- `$HOME/nreal/include/startup/RotateBlurA.b`
- `$HOME/nreal/include/startup/ui/RotateBlurA_UI.b`

Typically, use the MacroMaker to create the initial files, and then modify the files with a text editor. You have created two macros, one by hand and an identical macro with the MacroMaker. Both of these processes create two nearly identical sets of files. The first files create the function, and appear in the `$HOME/nreal/include/startup` directory. The second set of files loads the macro into the interface, and is found in the `startup/ui` subdirectory. The interface files are separate as you may want the file to only appear in the command line. Why bother doing the function by hand? Not necessarily because you are being punished, but rather so you can expand your macros with custom behavior and buttons for the parameters, and to edit the functions. For more information, see Chapter 17, “Customizing Shake,” and Chapter 18, “Macros, Expressions, and Scripting,” in the *Shake 3 Reference Guide*.

Lesson Nine: Creating a Clean Plate

This tutorial discusses a few tips on making a simple clean plate with the *QuickPaint* node.

Tutorial Summary

- Creating a Clean Plate
- Paint With the Reveal and Clone Brushes in the *QuickPaint* Node

Creating a Clean Plate

Your first question may be, “What is a clean plate?” This is a plate of the background that has unwanted foreground material removed, kind of like when you want to remove somebody from a photo who is no longer popular with the Politburo. But once you remove the person, you have to fill in the hole. What do you fill in the hole with? That’s right, the clean plate.

In this example, you use extracts from the following two frames to create a clean plate.

The following images are courtesy of Mr. Ron Brinkmann from his film, “Hope” © 2002, by Precipice Pictures.

- Begin with this image:



- Remove the two figures, and end with this image:

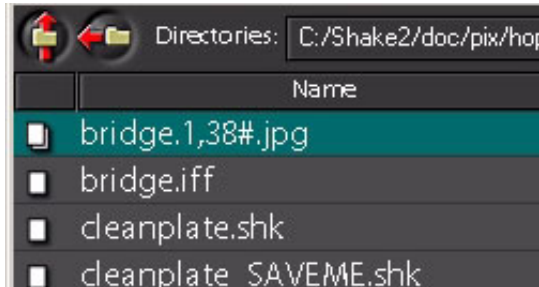


You might immediately think “Clone brush” in *QuickPaint*, and you are in fact partly correct. However, you cannot clone everything, so you also have to bleed in a different frame over the figures.

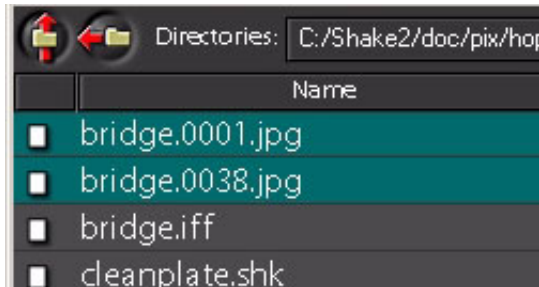
To load the tutorial images:

- 1 In the Image tab, click *FileIn*.
- 2 In the Load image or sequence browser, navigate to *doc/pix/hope*.

There are two images listed as one sequence, *bridge.1,38#.jpg*. (Do not read the images in yet.)



To read the images in as two separate images, toggle the Sequence Listing button off (the button is at the bottom of the Browser window). These images are now listed as two separate files. Drag-select (or **Shift**-click) the images and click OK.



Two separate images (*FileIns*) are read in.

Note: If Sequence listing is enabled when the files are read in, you get one *FileIn* sequence of two frames, rather than two *FileIns* of one frame each.

The two tutorial files are preselected from a much longer sequence. When creating a clean plate, try to select frames with as little overlap of the foreground characters as possible. Because this is not always practical, you should always request a clean plate shot whenever possible (and save yourself the tedious task of going through this tutorial).

Frame 1, FileIn1

Frame 38, bridge



You will be modifying frame 1 (in this case, named *FileIn1*) since the people are smaller and you have less area to worry about. The first step is to align the two plates.

To align the background and foreground plates:

- 1 In the Node view, select the *bridge* node (frame 38 *FileIn*) and attach a Transform—*Move2D* node.



- 2 Add a Color—*Invert* node to the *Move2D* node.
- 3 Select the *FileIn1* node and add a Layer—*Mix* node.

- 4 Connect the *Invert1* node to the second input of the *Mix1* node.



This conveniently sets up a situation where variations in the image stick out like frog's whiskers.

- 5 View *Mix1* (click the left side of the node) and display the *Move2D* parameters (click the right side of the node).

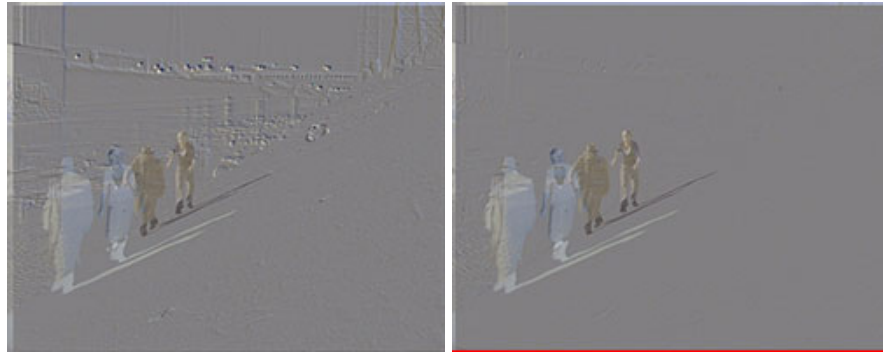
The *Move2D* onscreen controls appear.

- 6 Pan until the alignment of the plates is exact.

As an interesting side note, you can get a fake *Emboss* function this way. (Not a very good one, though. You'd get fired if you used it as an *Emboss*, so don't do it.)

Almost...

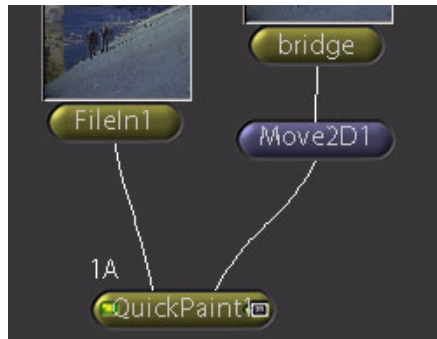
Right On



The images are aligned when most of the background noise disappears.

- 7 In the Node View, select the *Invert1* node and press **Del** / **Delete**.
- 8 Select the *Mix1* node and **Command-click** / **Ctrl-click** on Image-*QuickPaint*.

The *Mix1* node is changed to a *QuickPaint* node.




- 9 Now comes the part that is easy to forget—in the *QuickPaint* toolbar, click the “frame” button two times to toggle to “persist.”

When Persist mode is set, the paint strokes remain on all frames. (Frame mode is the default because the beta testers wanted it that way, so you only have yourselves to blame. Go complain to each other on highend2d.com.)





Paint With the Reveal and Clone Brushes in the QuickPaint Node

Use the Reveal brush to reveal the second input image into the first input image. You can use a hard-edged brush (press **F3** to toggle) for plates with a lot of grain—using a soft edge interpolates the color and softens the grain along the edges. Anyway, where the people are, reveal away.

- 1 In the *QuickPaint* toolbar, click the Reveal brush button .
- 2 In the Viewer, **Command**-drag / **Ctrl**-drag to resize the brush, if necessary.
- 3 Click and drag over the image to paint.



- 4 If you reveal the people from frame 38, use the Erase brush  to tidy up. The Reveal brush cannot do the whole job. To get rid of the last bit, use the Clone brush to copy some of the nearby color.
- 5 Click the Clone brush button .

- 6 **Shift**-drag to relocate the brush from the clone source.



You can render the image (save the script somewhere handy), or use it as part of your tree. It is recommended to render the image as it slightly reduces your scene complexity.

And, yes, the water should be moving.

The Cookbook

The Cookbook is an area for general tips that do not fit neatly into other categories. None of this stuff is gospel—there are no doubt different clever ways to do many of these things.

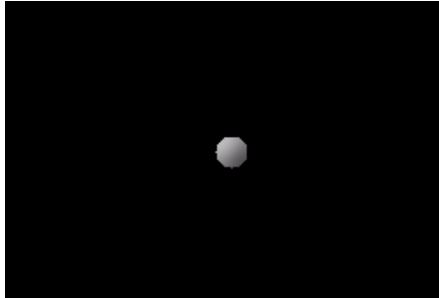
Cookbook Summary

- Good Habits
- Bad Habits
- Coloring Tips
- Filtering Tips
- Keying Tips
- Layering Tips
- Transform Tips
- Depth Tips
- Text Treatments
- Expressions
- Cookbook Macros
- Command-Line Macros
- Image Macros
- Color Macros
- Filter Macros
- Key Macros
- Transform Macros
- Warp Macros
- Other Macros
- Using Environment Variables for Projects

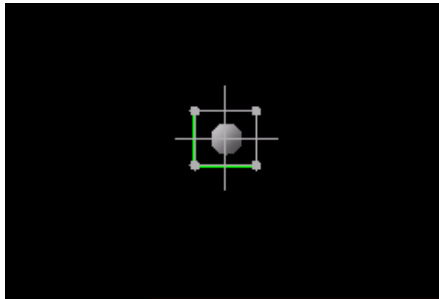
Good Habits

Use the Transform–SetDOD Node

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.



Take Advantage of Concatenation of Color Correctors and Transforms

Several nodes under the Color and Transform tabs concatenate with nodes of the same class (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 small C:



To take advantage of this, try not to mask or insert nodes between these concatenating nodes. In these two examples, the second tree is more efficient because *ContrastLum* does not concatenate, but *Compress* and *Brightness* do. Therefore, it is better to plug *Brightness* into *Compress*.

Bad

Good

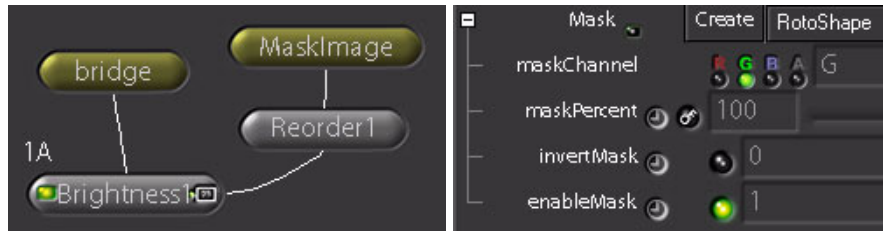


Bad Habits

Do not do the following things or poisonous monkeys will be sent to punish you.

Reorder Before Masking

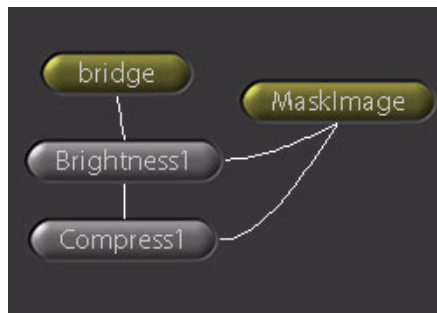
This pops up a lot in client scripts, which is a *Reorder* on an image before it is fed into a mask. This is unnecessary as the mask inputs, *SwitchMatte* and *KeyMix*, all have channel selectors. Probably there is no computational difference, but it is one more node in the tree.



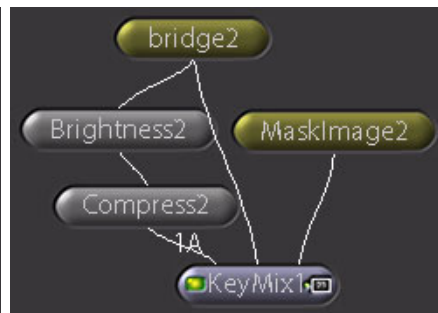
Masks on Concatenating Nodes

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*.

Bad



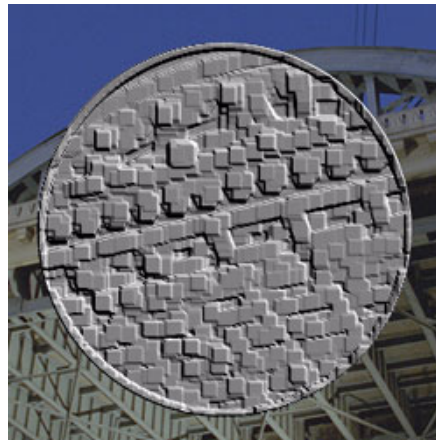
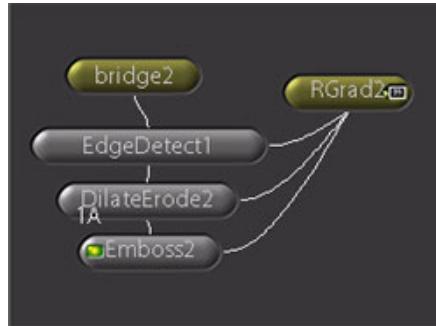
Good



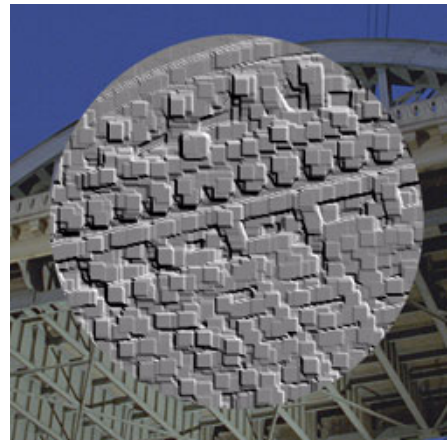
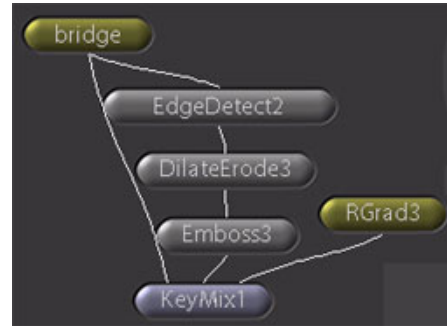
Same Mask Used Multiple Times

Even if the nodes do not normally concatenate, but are still adjacent, you get cleaner edges with the use of a *KeyMix*. In this example, a circular mask is applied on 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



Saving Black-and-White Images As RGB Images

Shake is channel agnostic—you can pipe any channel image into any other. When you generate or save mask images, save the images as formats that support 1-channel images (RLA or IFF, for example) to reduce disk space and network activity. You can quickly strip channels out using the command line:

- Strip out the RGB channels, leaving the alpha:
shake mymask.#.tif -bri 0 -fo mynewmask.#.iff

- Strip out the alpha channel, force the RGB as a monochrome 1 channel:
`shake mymask.#.tif -mono -setAlpha 0 -fo mynewmask.#.iff`
Note: Shake automatically optimizes itself to only read the channel it needs.
- If you want to save out a BW image as RGB for compatibility with other products, set the RGB output in the *FileOut*, or use the command line function *-forcergb*:
`shake myBWimage.#.iff -forcergb -fo myRGBImage.#.iff`

Coloring Tips

Tinting

The following five techniques demonstrate the versatility of the color correction nodes. Each one applies a tint to the image, meaning the midtones are pushed towards a certain color while leaving the blacks and whites alone. None of these techniques have 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.”

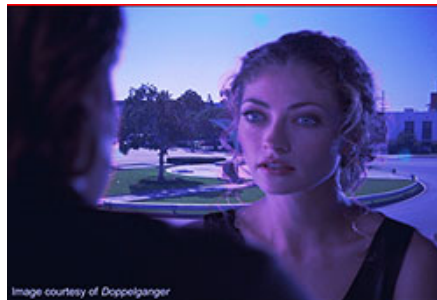
- *Brightness + Mult*: These nodes concatenate in the following node tree. This does not work well if you use a pure color in a single *Mult* (in this case, pure blue with a value of 0,0,1) because the zeroes drop the midtones out completely on the red and green channels. The *Brightness* is set to approximately 3, helping to maintain the red and green channels when the blue multiplier brings them back down (.3, .3, .8 in *Mult1*).



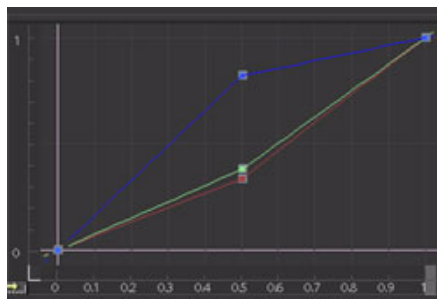
- *Monochrome + Brightness + Mult*: This is identical, except you get a purer blue color since you have made a monochrome image before applying the color correction. Note that the *Monochrome* node does not concatenate with the *Brightness* node.



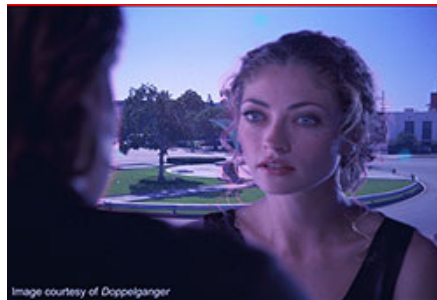
- *Lookup*: This uses a *Lookup* curve, setting the midpoints of the curve to .3, .32, .8.



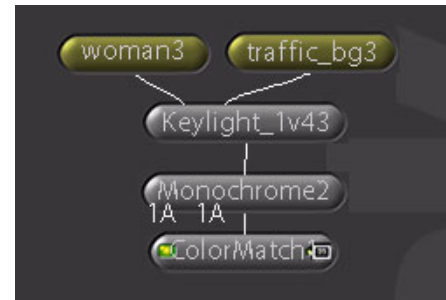
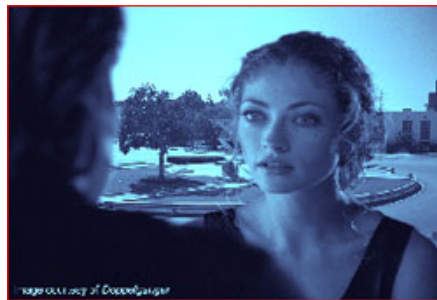
The curve looks like the following—the curves are Linear to mimic a *Tint* function from a different package.



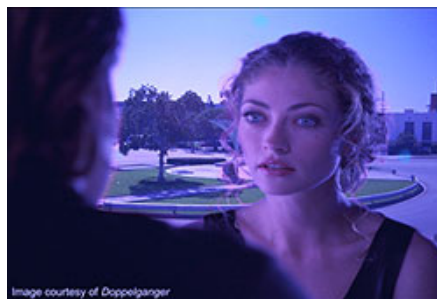
- *ColorMatch*: This is similar in theory to the *Lookup*, as it allows you to push the lows, mids, and highs. However, the internal math helps reduce solarization (hills and peaks in the curves), so you maintain a little bit more of the input color.



You can get interesting adjustments of your values if you adjust the source points on the ColorMatch. (For example, hold down **⌘** and drag left on the highSource Color Picker.)



- *ColorCorrect*: This is an Add on the Mid areas using -2 , -2 , $.5$.



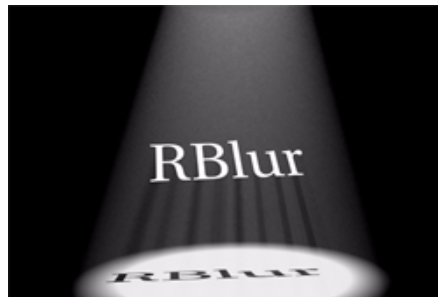
- Using *Mix*: You may end up with several nodes to achieve a particular color correction. This may be awkward to tune. Therefore, a convenient way to quickly adjust a result is to mix it back into the input image with a *Layer-Mix* node. Naturally, it is always faster to process by adjusting the original color nodes, but using a *Mix* may be easier for you to keep a handle on things.



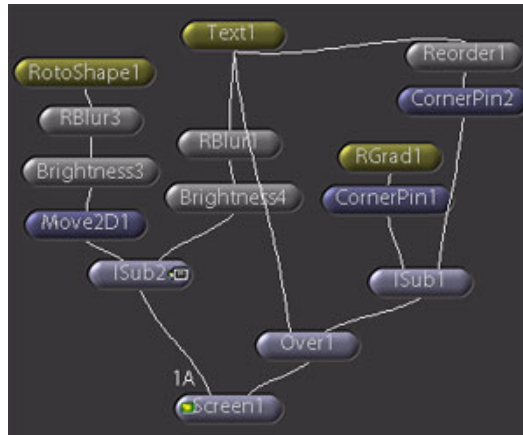
Filtering Tips

Volumetric Lighting

This script can be found in `doc/html/cook/scripts/volumetric.sbk`. 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

Keying Clouds

This script can be found in *doc/html/cook/scripts/clouds.sbk*. The images used are *doc/pix/moon.iff* and *sky.iff*.

This is a potential technique for keying clouds, but may also be useful for flames. It discusses several approaches.

This script puts the moon behind the clouds:

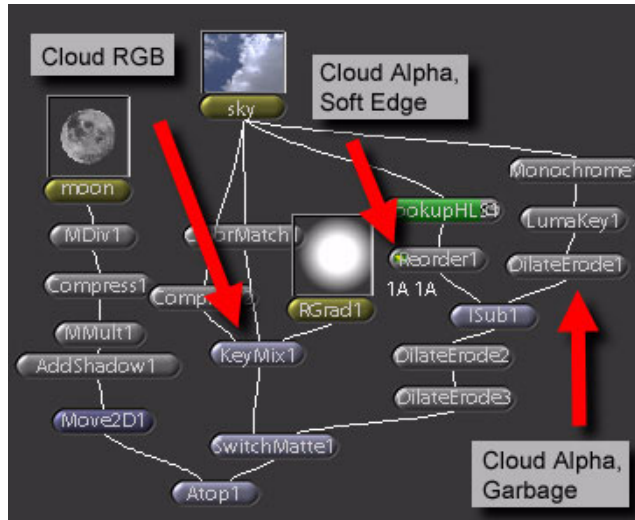


You might think a color-based key would work, but basically everything is a shade of blue, so that won't do. What were you thinking?

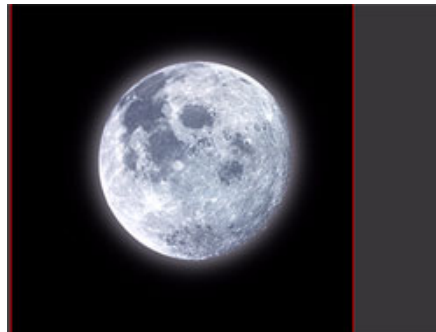
Another tactic is to use a *LumaKey*. In this attempt, the moon is corrected (using a *Compress*, an *Other-AddShadow* to add the white glow, and then positioned with *Move2D1*) to adjust the color and position of the moon. A key is pulled on the clouds with *LumaKey*, and a *Layer-Atop* is used to layer the moon only where there is alpha in the background. This displays two typical problems with this sort of keying: There is a black edge, and there is not complete opacity in the thick part of the clouds. This is bad.



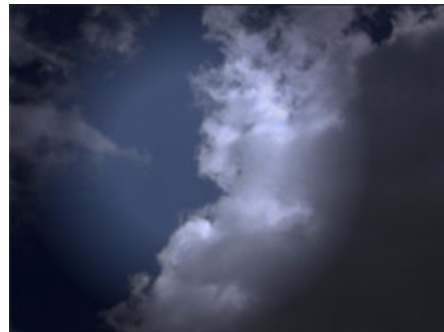
In this next attempt, there are three main branches. The first, identical to the second attempt, manipulates the moon. The second branch, terminating in *KeyMix1*, works on the RGB of the clouds. The *KeyMix* mixes a darkened cloud image with a brighter cloud image through the *RGrad*, giving it the “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. Therefore, first insert a *Color-Monochrome*, and boost the rWeight up and the g- and bWeight down before you key.

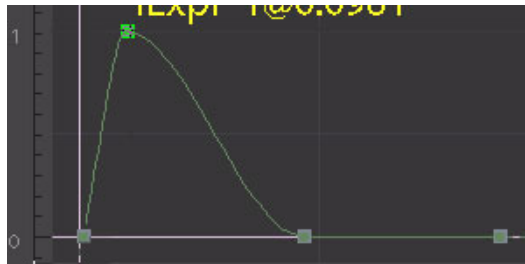
Red Channel



Blue Channel

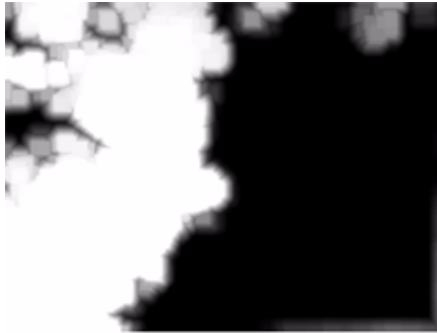


A *Color-LookupHLS* manipulates the luminance, and then switches that into the alpha with *Color-Reorder* (a macro begging to happen) to key the deep part of the clouds in the lower-right corner. The curve looks like the following:

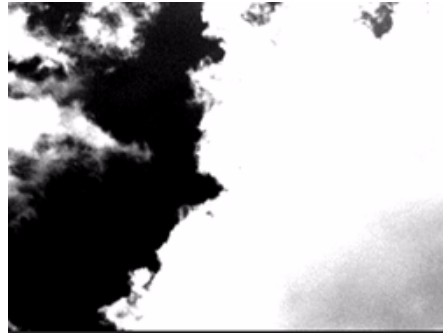


The *LumaKey* gets the edges of the cloud, and is enhanced by a *DilateErode*, and then is subtracted from the soft key.

DilateErode3



LumaKey1



You still get black edges, so sprinkle Filter–*DilateErode* nodes liberally. For the nodes attached to *ISub1*, 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 the *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.

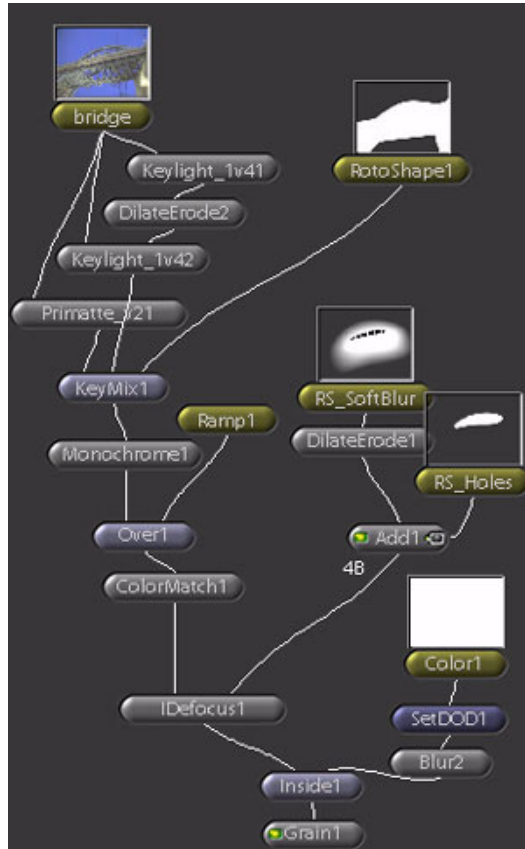
Vignette

This script can be found in *doc/html/cook/scripts/vignette.sbk*.

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, roto is your friend. (Just not a very fun friend.)

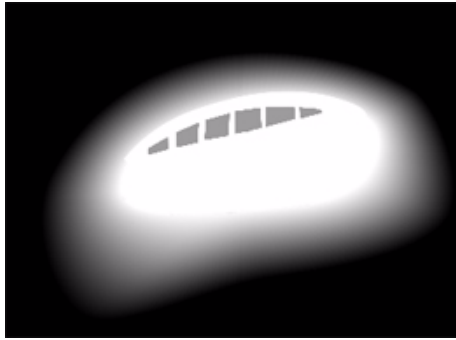


The first branch, down to *KeyMix1*, pulls a key on the sky. The first *Keylight* pulls a hard key to be fed into the second *Keylight*. This is the core key for the bridge. This is keymixed with a *Primate*-based key for the blue area through *RotoShape1*.

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 Pickers 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

Bleeding Background Color Into the Foreground

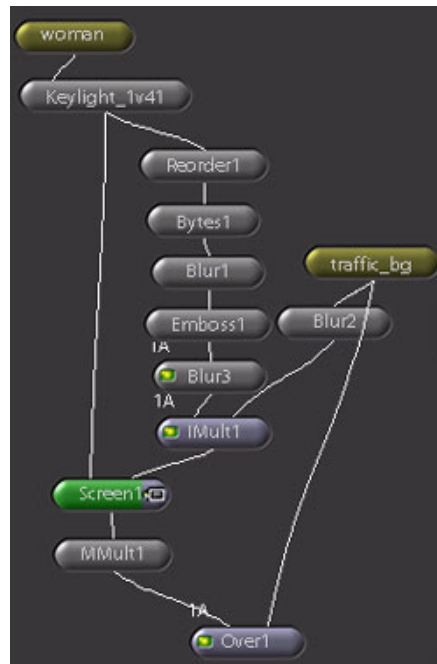
This script can be found in *doc/html/cook/scripts/edgelight.shk*.

This script, which has its effect rather exaggerated for illustration purposes, 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



With Exaggerated Rim Lighting



- *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



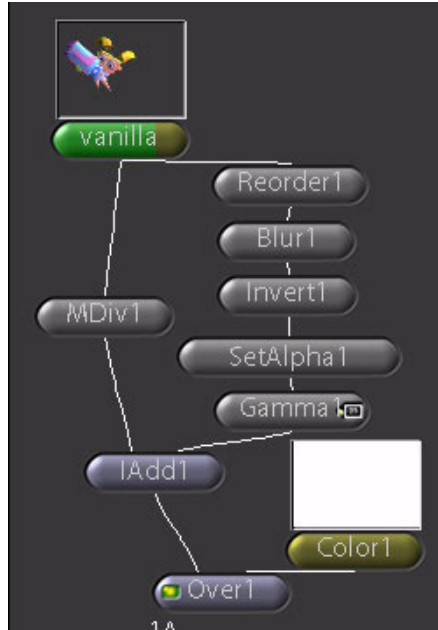
Background Flare

This script can be found in `doc/html/cook/scripts/backlight2.shk`. “Vanilla Pudding” images courtesy of Wild Brain, Inc. in San Francisco.



This script translates D.W. Kim’s fine backlighting page from highend3d.com into Shake terms. It uses the Vanilla image from the Shake tutorials. There are other ways to do this.

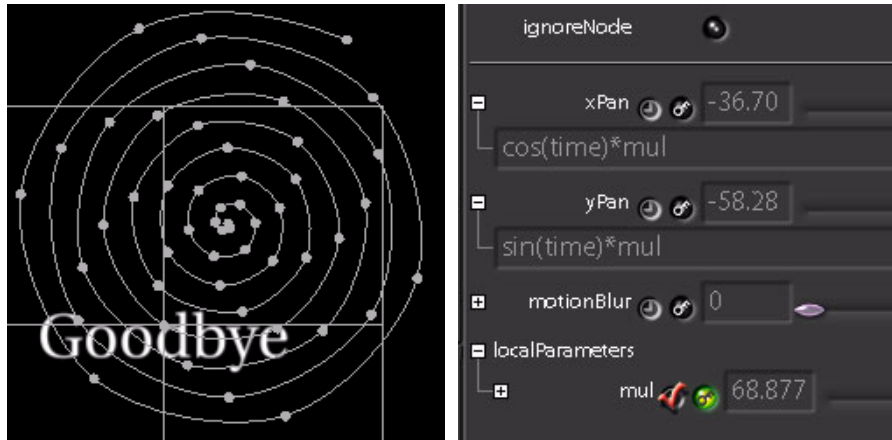
The script begins with a *Color-Reorder* that puts the alpha into the RGB channels. It then *Filter-Blurs* it, inverts that, removes the alpha channel with a *Color-SetAlpha* set to 0, and then is added back onto the plate. The *Color-MDiv* is used because the *IAdd* disrupts the premultiplication status. The *Color-Gamma* can be used to tune the intensity of the flare. Finally, *preMultiply* is activated in the *Layer-Over*.



Transform Tips

Toilet Bowl

This animates something in a spiral pattern, running ever smaller (or larger) concentric rings. First, right-click in the parameters area, select Create Local Variable, and 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:

$\sin(\text{time} * \text{freq}) * \text{mul}$

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.

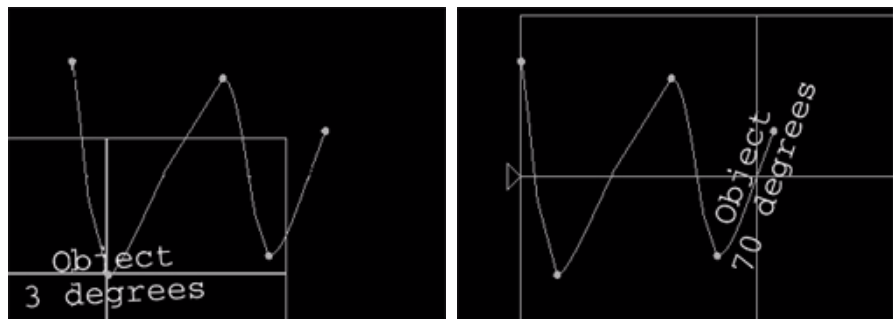
To burn the path in, do the following:

- Set timeRange under Globals (1-50, for example).
- Right-click over the xPan expression and select Save Expression.
- In the lower-right corner of the Browser is a setting for Auto or Raw. Set it to Raw.
- Enter a name for your curve, for example, curveX.txt.
- Do the same thing for yPan, saving it as curveY.txt.
- Create a second *Pan* node.
- Right-click and select Load Expression on the xPan parameter.
- Set your format in the Browser as Raw.
- Do the same for yPan.

Not elegant, but it works.

Auto Orient

This script can be found under *doc/html/cook/scripts/autoorient.sbk*.



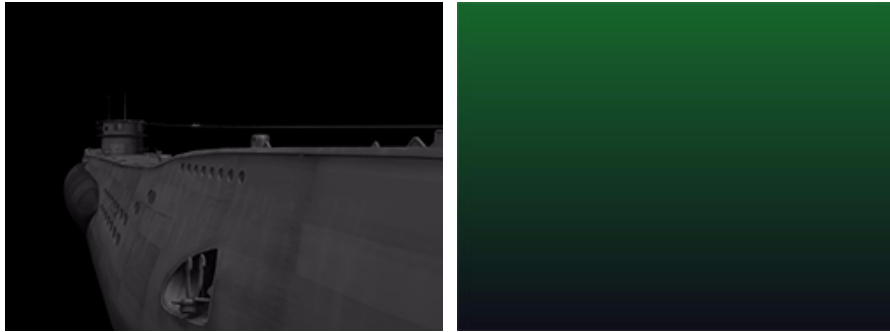
This script demonstrates how to set up a transform so that an element automatically rotates according to the tangent of the move. Although you can perform this with one node, it is better to use two so that you can continue to modify the position without destroying the expression to do the rotation. 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.

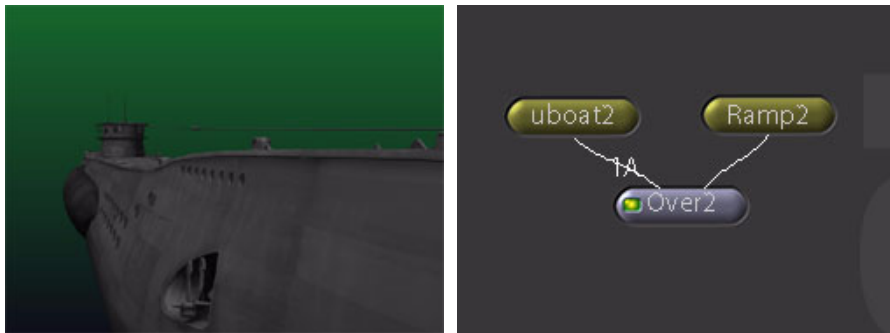
Depth Tips

Fog

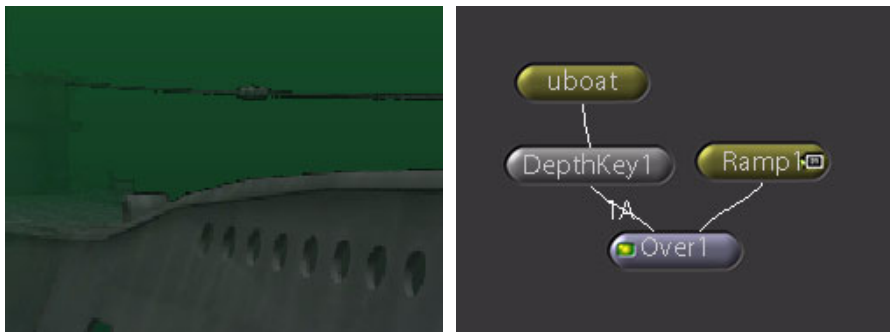
This image can be found in *doc/pix/uboaat.iff*. The background is just a *Ramp*. The trick is to apply depth-cueing to the U-boat:



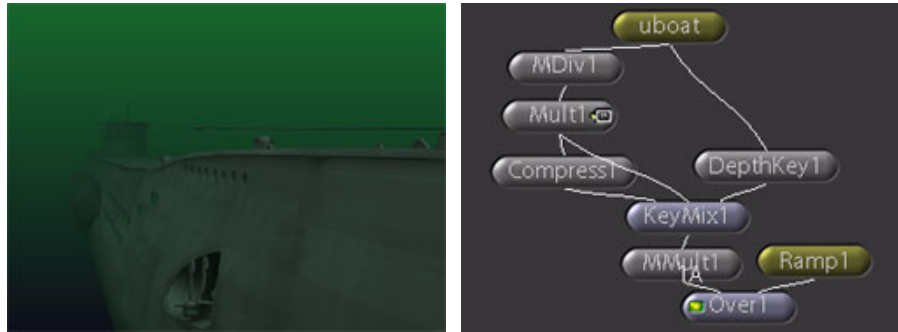
The first composite, without any fog, is not so stellar:



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:



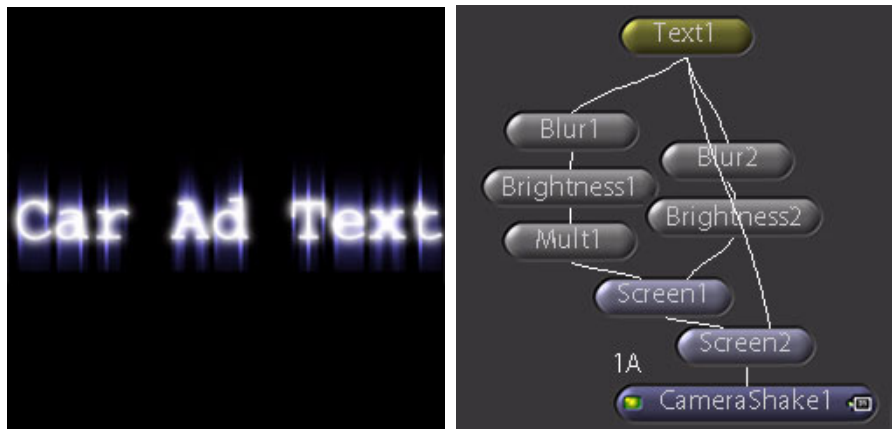
This more complex approach uses the *DepthKey* as a mask for a color correction, in this case, *Compress*, which has identical hi and lo colors. A *KeyMix* is used to get the concatenation with *Mult* and *Compress*. The *Mult* is used to tint the boat green; *Compress* gives the feeling of depth:



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. (If skillfully blended into a car advertisement, these renders lead people to believe that the purchase of a luxury SUV designed for the rugged and wild paved city streets and getting the gas economy of a Sherman tank is a really good way to dispose of \$42,000.)

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*. 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

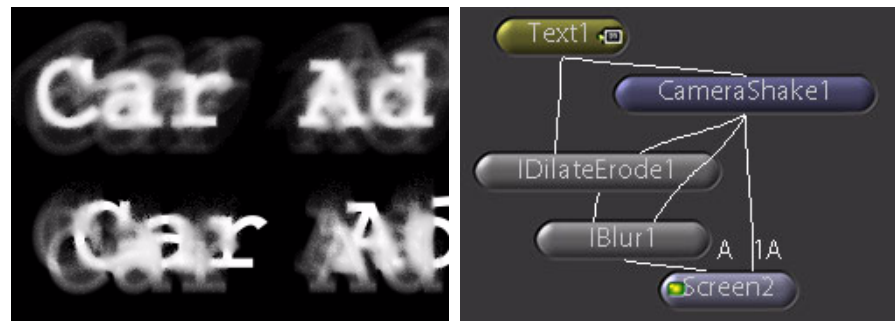


This uses the *noise()* function to randomize the xCenter of the *RGrads*. The text is then held *Inside* of these two animated shapes, and a process similar to Script 1 is applied:

- *RGrad1* xCenter expression: $noise(time/4)*Text1.width$
- *RGrad2* xCenter expression: $noise(time/4+100)*Text1.width$

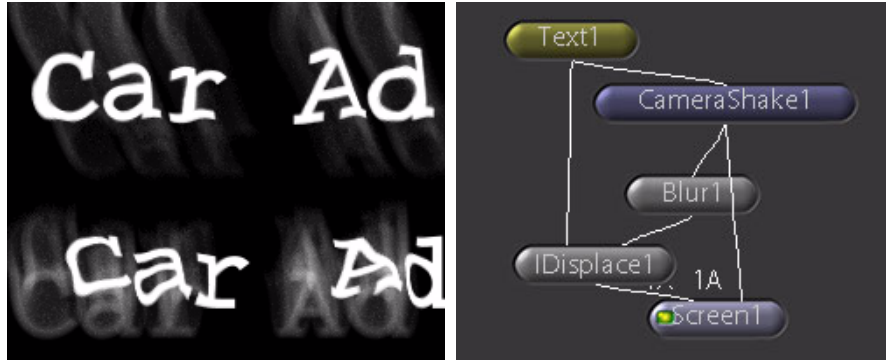
By adding 100 to the *noise* function's seed, you do not have an overlap in the animation. You can also change *time/4* to increase or decrease the frequency, that is, *time/10* or *time/2*. See "Expressions" on page 232.

Script 4



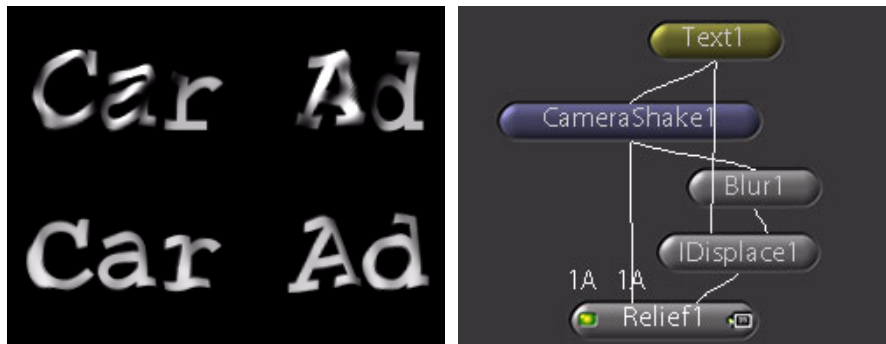
This uses a heavily motion-blurred *CameraShake* (frequency is set to 6, amplitude is set to 50 and 100 for x and y). This then drives an expanding *IDilateErode* and *IBlur* to create an interesting interaction with the text (that will somehow convince people to buy more cars).

Script 5



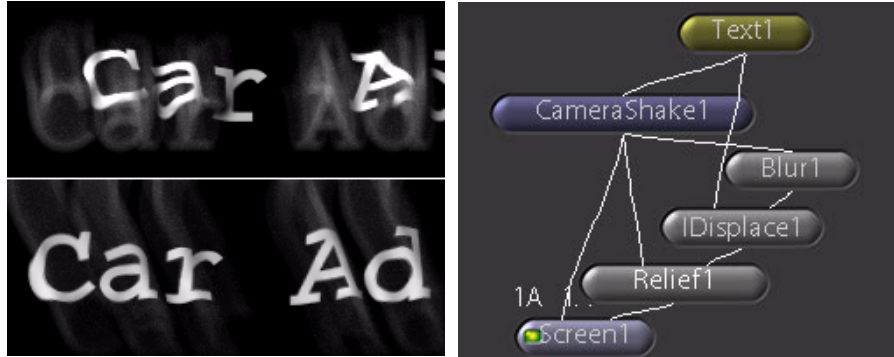
The same *CameraShake* from Script 4 is used to feed an *IDisplace*. The *Blur* helps soften the warping image.

Script 6



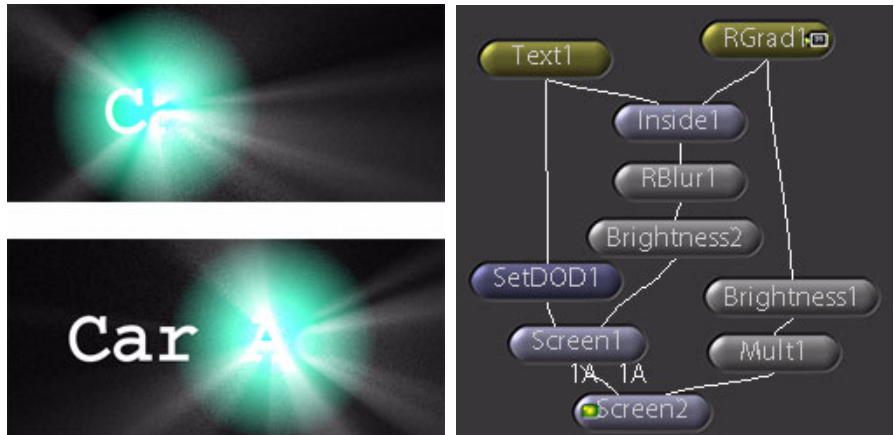
This is the same as Script 5, except the *Screen* is replaced with the *Relief* macro, found in *doc/html/cook/macros*. The script does not open without the *Relief* macro. The affect parameter of *Relief* is set to image 2.

Script 7



This is the same as Script 6, but the result is fed back over the motion-blurred text.

Script 8



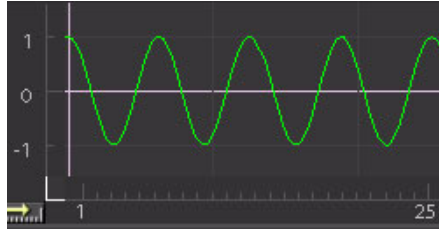
This script is driven off of the position of the *RGrad*. The center of the *RBlur* is set to the center of the *RGrad*, and the right parameter of *SetDOD1* is set to $RGrad1.xCenter + 10$, which unmask the text as the *RGrad* moves to the right.

Expressions

Signal Generators

These functions all generate a value that changes over time. Typically, you feed the variable *time* to modify the value.

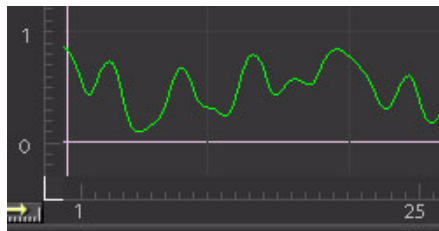
cos(time)



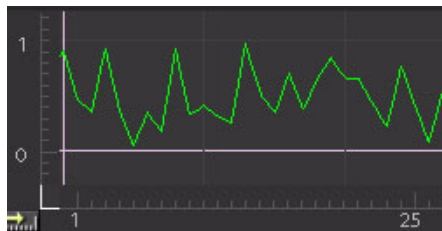
sin(time)



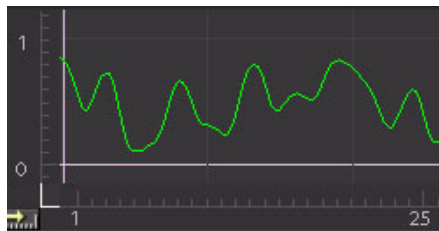
noise(time)



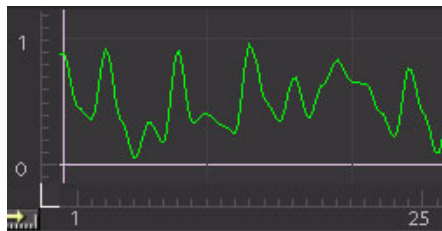
Inoise(time)



fnoise(time,1)

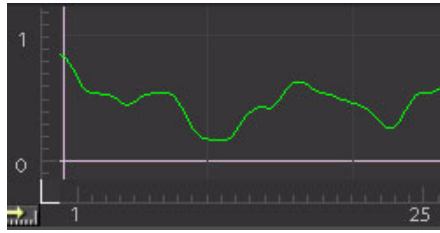


turbulence(time,1)

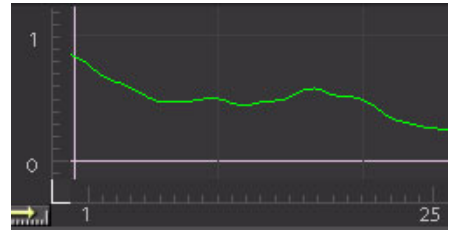


fnoise() and *turbulence()* have additional frequency factors to the noise.

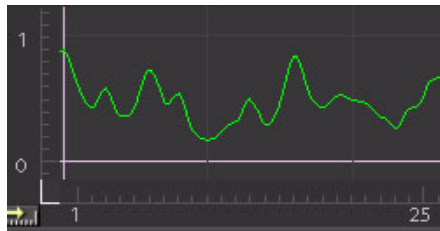
fnoise(time,2)



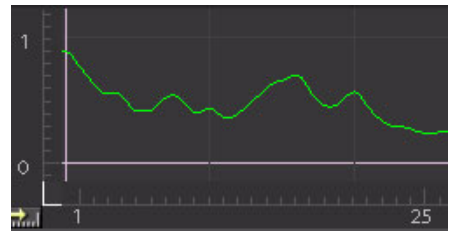
fnoise(time,5)



turbulence(time,2)



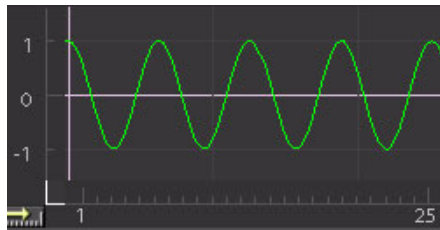
turbulence(time,5)



Offsetting a Function

To offset a function, add a value to *time*.

cos(time)



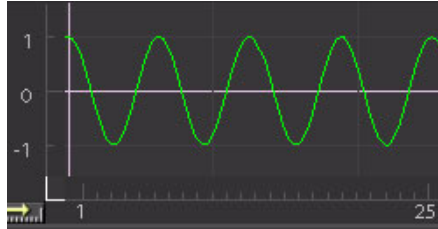
cos(time+10)



Changing the Frequency of a Function

To change the frequency of a function, multiply or divide *time* by a value. *fnoise()* and *turbulence()* both have frequency controls, but values are not modified below 1, so you may still have to modify *time*.

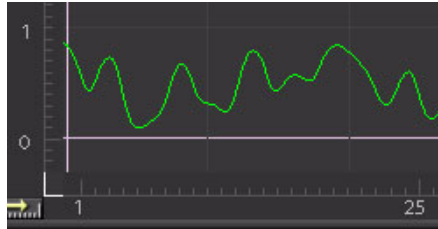
cos(time)



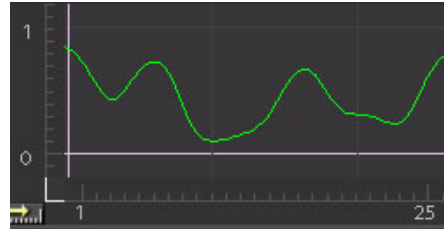
cos(time/3)



noise(time)



noise(time/2)



Frequency and Continuous Versus Discontinuous Noise

The noise generators *noise*, *lnoise*, *fnoise*, and *turbulence* are continuous noise generators, meaning you can draw a continuous smooth curve between the values. For example:

$noise(1) = .554$

and

$noise(1.1) = .5182$.

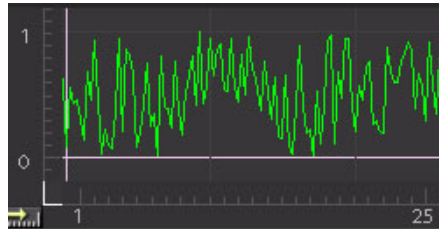
You can therefore safely predict that $noise(1.05)$ is near and probably between these two values. Thankfully, we're not rotten disreputable liars, since it equals $.5358$. However, if you use the *rnd()* function, you arrive at discontinuous noise. For example:

$rnd(1) = .4612$

$rnd(1.1) = .4536$

You might guess that `rnd(1.05)` is between those, but it in fact equals `.0174`, not `.458`. This is why it is called discontinuous noise. Looking at the neighboring values does not help you to arrive at a safe guess for the between values. For this reason, frequency changes have no practical effects on the curve.

rnd(time)



Setting Ranges for Expressions

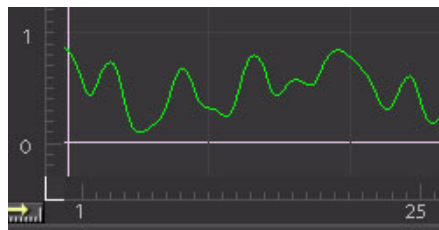
The noise generators return values between 0 and 1. `sin()` and `cos()` return values between -1 and 1. To adjust the range, use addition and multiplication. For example, suppose you want to spit out values between 100 and 400. To make a noise generator do that, subtract the low value from the high value. This is your multiplier. Then add the lower value of your range. Thus, you have:

$$\text{noise}(\text{time}) * 300 + 100$$

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:

$$(\text{sin}(\text{time}) + 1) * 150 + 100$$

noise(time)



noise(time)*300+100

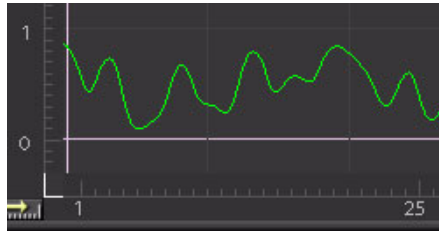


Modifying Noise

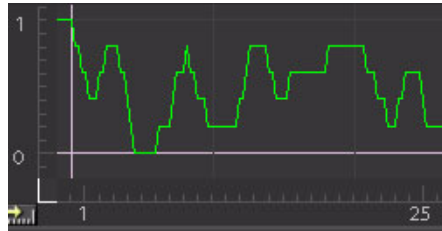
You can use other functions like `ceil()`, `floor()`, or `round()` to help you break a curve into steps. Ceiling pushes a number to the next highest integer, floor drops it to the next lowest, and round rounds off the value.

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)

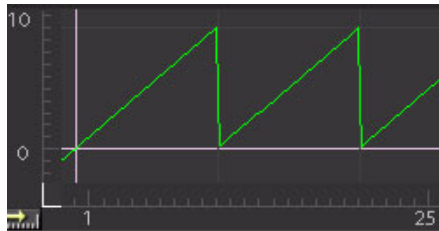


floor(noise(time)*6)/5



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.

Cookbook Macros

All Shake macros can be found in *doc/html/cook/macros*. To install the macro, place the macro .h file in *<UserDirectory>/nreal/include/startup*. Place the files containing a UI in *<UserDirectory>/nreal/include/startup/ui*.

Command-Line Macros

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. Don't tell anybody you are using this; you will get fired.

```
shake -framefill bus2.#.jpg -t 41, 45
```

Vanilla, Frame 1



Vanilla, Frame 3



Vanilla, Normal Frame 2



Framefill Frame 2



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 on the image—the coordinates appear in the title bar.

In the `doc/pix/bus` directory:

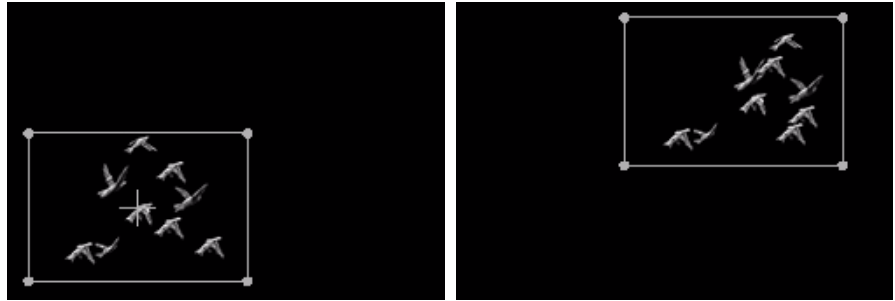
```
shake bus2.0040.jpg -unpin 91 170 417 154 418 2 42 94 274
```



Image Macros

Flock Macro

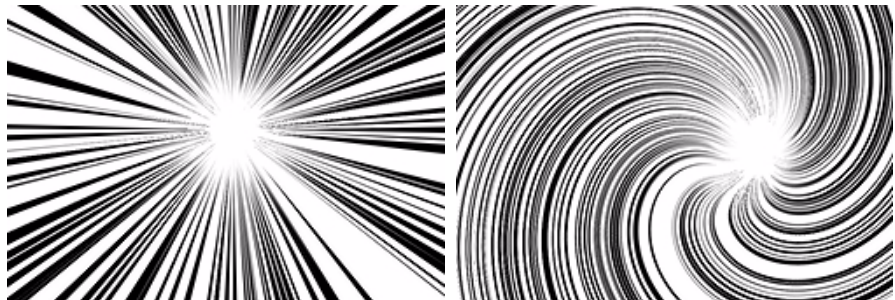
The following bird clip can be found in *doc/html/cook/macros/bird*. This takes a cycling clip and propagates copies of it offset in time, position, and scaling. There is a clip of birds provided as an example that you can use, royalty free. If you use it, let us know.



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 *Y* controls the frequency of the cosine movement (an up and down wave), and *vibrationX* and *Y* and *vibFreqX* and *Y* 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*”).

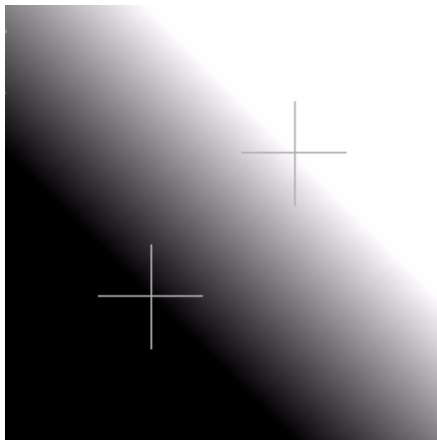
Rain Macro

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. See highend2d.com for more examples of the NGLRender commands. Check out Arc and Concentric. 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

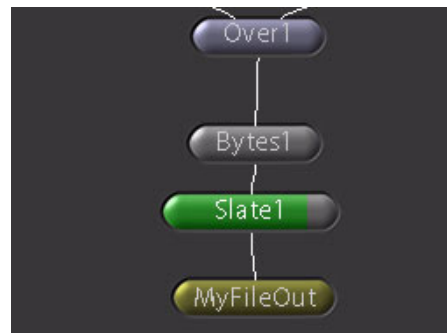


Slate Macro

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.

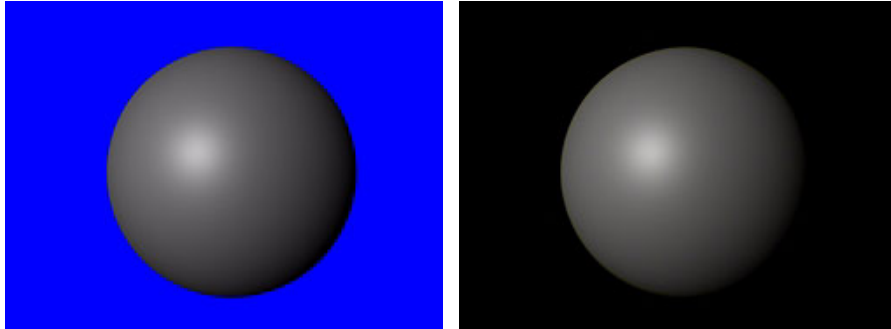
The slate appears up to the markFrame parameter, which is 0 by default. It also loads the script name and the frame range into the slate, adds the current date, and gives you the option to stamp the current frame onto the output images.



Color Macros

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 be a little iffy, so if it does not work out, let us know...



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 unfortunate 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



ColorGraded Plate



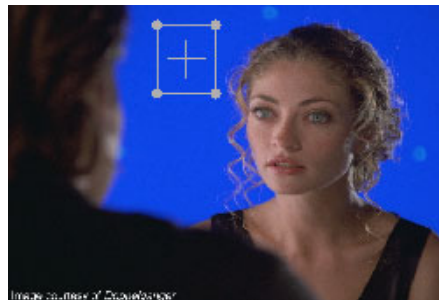
This example has a slightly high saturation, a slight blue cast, and punchier whites (but then again, 30 seconds were spent on it). Note because the tree is made of three concatenating color corrections, it was not necessary to convert up to float bit depth before the *LogLin*.



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 blue screen 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



Filter Macros

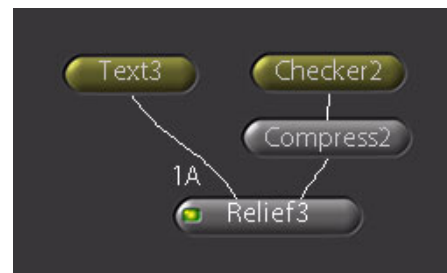
Relief Macro

In the following, Example 1 has affect set to image1; the other two are set to image2.

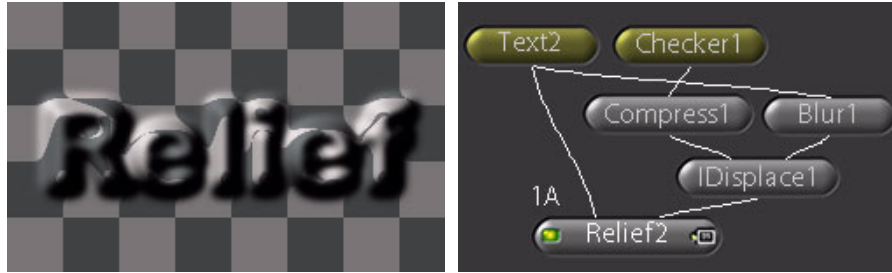
Example 1: affect = image1



Example 2: affect = image2



Example 3: Relief + IDisplace



Key Macros

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 seriously mess with your RGB channels. This macro is unnecessary for 8- or 16-bit images.

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 destroys 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

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):

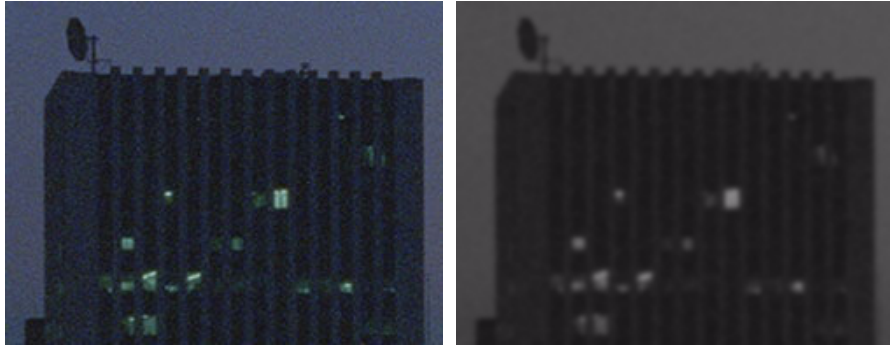
```
shake woman.sgi -autofit 2048 w
```

This resizes *doc/pix/woman.sgi* to 2048 x 1383, thereby keeping it the same aspect ratio.

PreTrack Macro

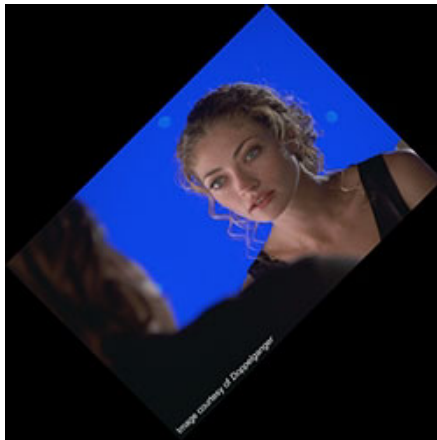
This macro helps when tracking noisy film plates. It drops the blue channel out, which tends to have the thickest grain, and also applies a slight blur to the footage. Once the track has been done, disable or remove the *PreTrack* node.

Note: The *Tracker* nodes have the ability to add blur while tracking.



RotateFit Macro

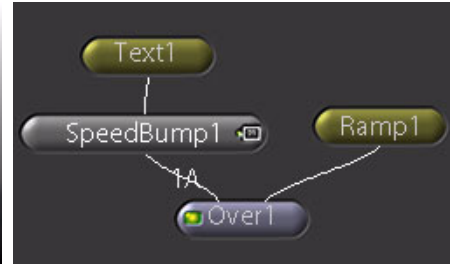
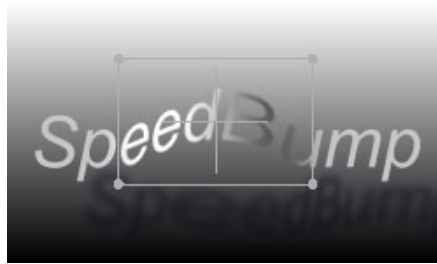
This macro resizes the frame to fit the new boundaries of the image. This version uses math to figure out the boundary, but you can also put the variables *dod[0]* - *dod[3]* into a *Crop* node.



Warp Macros

SpeedBump Macro

This macro creates a nifty bump with a shadow on your title.

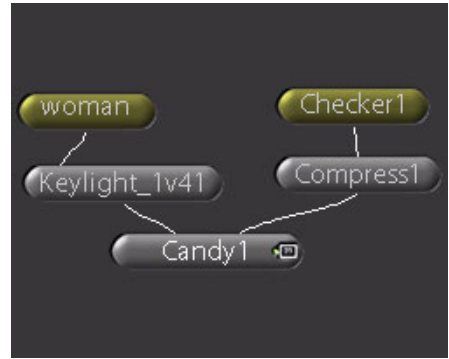


Other Macros

Candy Macro

With this macro, the drop shadow only appears 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 `Other-AddShadow` or `DropShadow` nodes. Both the Foundry and Gen Arts have similar functions, and they are much faster.





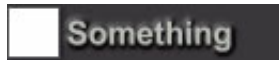
MakeNodelcon Macro

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 (for more information, see Chapter 17, “Customizing Shake,” in the *Shake 3 Reference Guide*). 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—go ahead, roll your own! It requires (get ready...) *Relief*, *ViewerButton*, *WallPaper*, and *RoundButton* to all be installed. None of these need the ui files, just the startup files. Additionally, *Roundbutton* makes a call to the *round_button.iff* file, also included in the *doc/html/cook/macros* 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

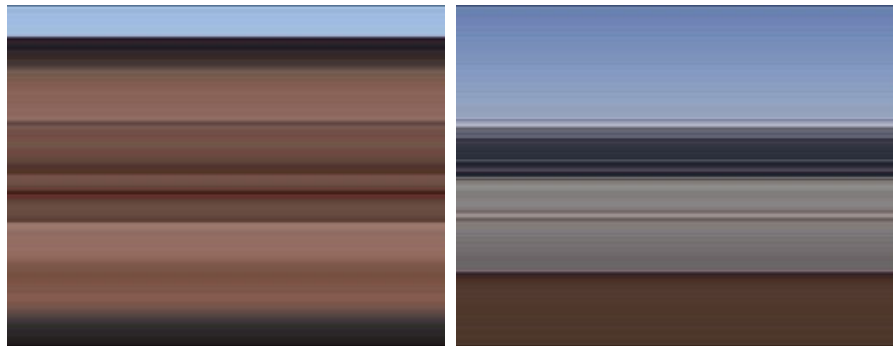
So, an example:

```
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.

For more information, see the “Creating Radio Buttons” section of Chapter 17, “Customizing Shake,” in the *Shake 3 Reference Guide*.

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).

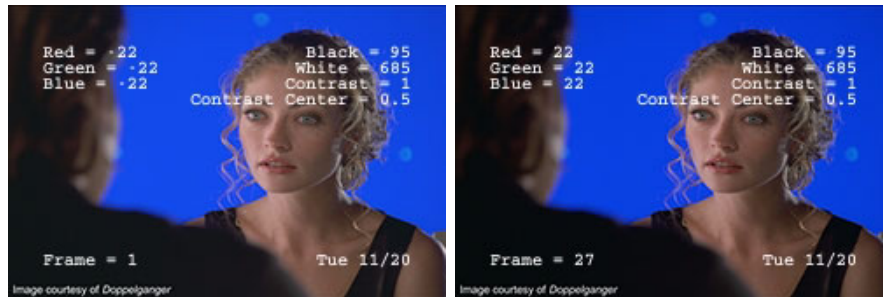
Wedge Macro

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. This points 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 and IRIX, it is typically `/usr/people/mylogin`. In Mac OS X, IRIX, 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 .tcsbrc
```

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.

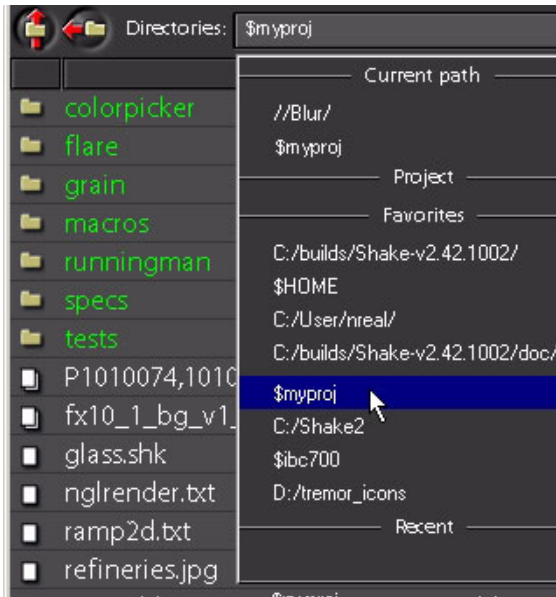
For the Shake portion:

- 7** Go to your *\$HOME/nreal/include/startup/ui* directory and create a text file called *paths.b*. 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/" ;
```

- 8** Create a directory that *\$myproj* points to, that is, if you set it to */Documents/shot1* then create */Documents/shot1*.

- 9 When you relaunch Shake, it should launch to *\$myproj*. In the Directories pull-down menu, you also see *\$myproj* listed.



- 10 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.
- 11 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, and, golly, with drive prices so low, who doesn't? 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.b* 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. For more information, see the advanced tutorials (3 -9) in the *Shake 3 Tutorials*, Chapter 17, “Customizing Shake,” and Chapter 18, “Macros, Expressions, and Scripting” in the *Shake 3 Reference Guide*.
- 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
```

