Quartz Composer User Guide

Graphics & Imaging > Tools

2007-07-17

Contents

Figures, Tables, and Listings

Introduction to Quartz Composer User Guide

Quartz Composer is a development tool for processing and rendering graphical data. Its visual programming environment lets you develop graphic processing modules, called compositions, without writing a single line of code. Quartz Composer is also a framework that lets you programmatically access, manage, and manipulate compositions created with the development tool. This document, however, is a guide to the Quartz Composer development tool. By reading this guide, you'll get an introduction to using the Quartz Composer editor and find out how to use it to create a composition. You'll also see how to use compositions as screen savers and in QuickTime movies.

You should read this document if you are a developer or visual designer who wants to:

- Get an orientation to the Quartz Composer development tool
- Create compositions that process graphical content
- Experiment with the latest Mac OS X graphics technologies

Quartz Composer brings together a rich set of graphical and nongraphical technologies, including Quartz 2D, Core Image, Core Video, OpenGL, QuickTime, MIDI System Services, RSS (Really Simple Syndication), and XML. The development tool lets you explore the visual technologies available in Mac OS X without needing to learn the programming interface for that technology.

Organization of This Document

This document is organized as follows:

- "Quartz Composer Basic Concepts" (page 9) defines the terms used in Quartz Composer and describes the evaluation model and the coordinate system.
- "The Quartz Composer User Interface" (page 19) gives an overview of the editor, the viewer, the patch creator, and the other user interface elements in the tool.
- "Basic and Advanced Tasks, Tips, and Tricks" (page 41) describes the fundamental operations needed to create a composition, gives timesaving tips, and shows how to use some of the more advanced features.
- "Tutorial: Creating a Composition" (page 55) provides step-by-step instructions for creating a composition and using it as a screen saver and a QuickTime movie.
- "Glossary" (page 69) defines the new terms used in the document.

See Also

- Sample Quartz compositions are available in `/Developer/Examples/Quartz Composer`.

- The Quartz Composer developer mailing list (quartzcomposer-dev) is an excellent place to discuss issues or topics with other Quartz Composer developers.
- *Quartz Composer Programming Guide* describes how to use the Quartz Composer framework to integrate compositions into an application.
- *Quartz Composer Custom Patch Programming Guide* provides instructions on creating your own patches that you can then use in the Quartz Composer development tool.

Quartz Composer Basic Concepts

Quartz Composer is a development tool provided with Mac OS X for processing and rendering graphical data. Its visual programming environment is suited for:

- Developing graphics processing modules without writing a single line of code
- Exploring the visual technologies available in Mac OS X without needing to learn the programming interface for that technology

After installing the developer tools provided with Mac OS X v10.5, you can find the Quartz Composer development tool in:

```
/Developer/Applications
```

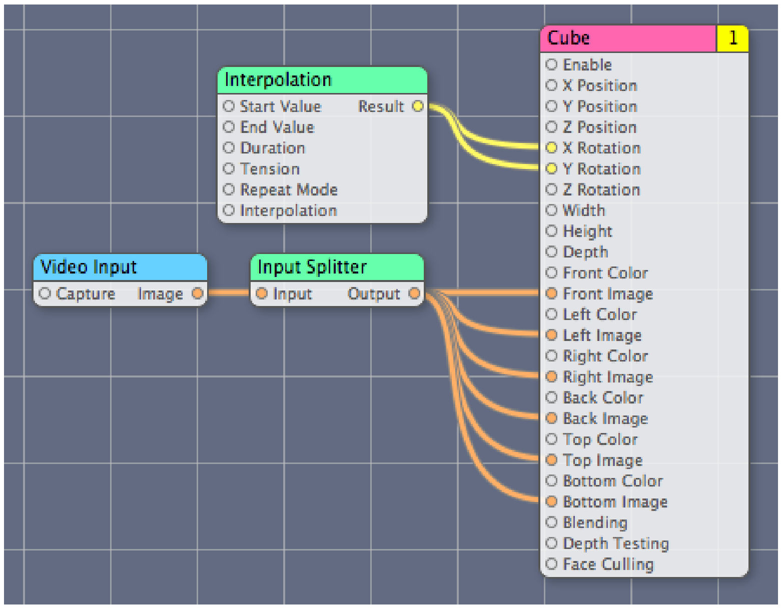
Quartz Composer brings together a rich set of graphical and nongraphical technologies, including Quartz 2D, Core Image, Core Video, OpenGL, QuickTime, Core MIDI Services, RSS (Really Simple Syndication), XML, and more.

This chapter describes the basic concepts you need to understand Quartz Composer. It defines the basic elements (compositions and patches), describes the flow of data in a composition, illustrates the coordinate system, and introduces the composition repository.

Compositions

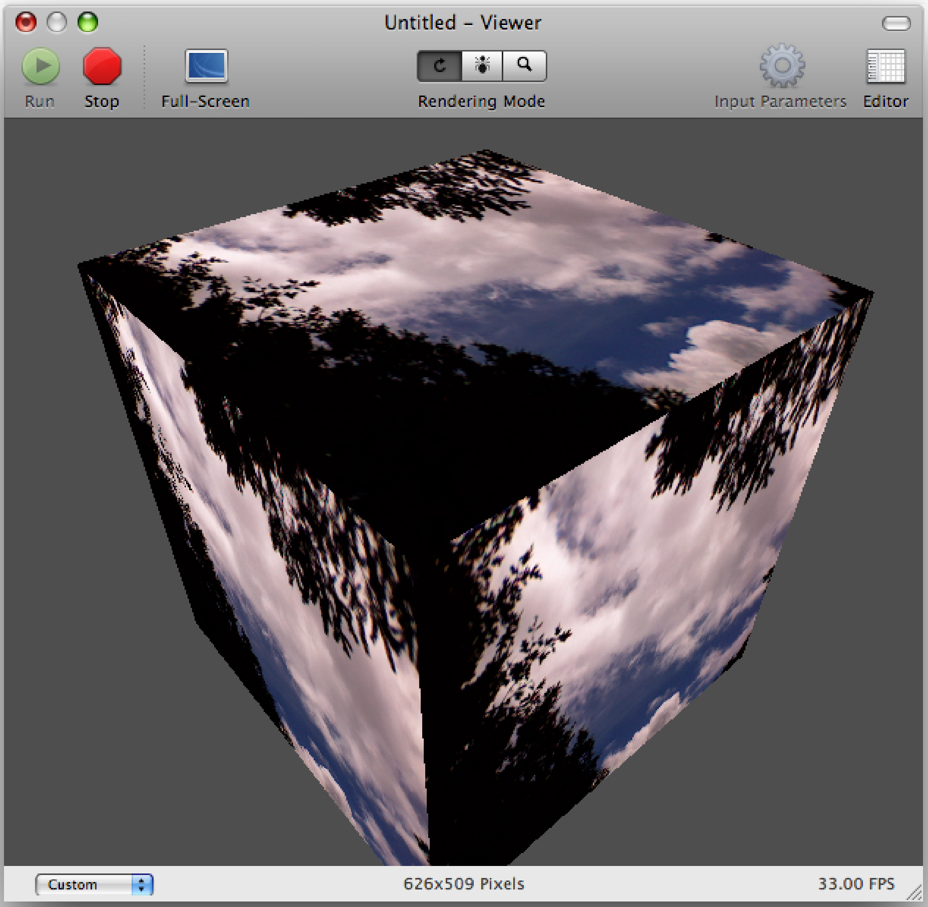
You use the Quartz Composer editor to create **Quartz compositions**, which are procedural motion graphics programs created by assembling preexisting modules (called **patches**) in a workflow for data processing and rendering. Figure 1-1 shows a simple composition.

Figure 1-1 A Quartz composition



Compositions can have input parameters and produce output results. The output produced by the composition shown in Figure 1-1 is a rotating cube whose faces show video captured by a camera connected to the computer (see Figure 1-2).

Figure 1-2 Motion graphics produced by a composition



A composition can operate autonomously, but it's also possible for any Mac OS X application to communicate with the composition and to integrate the composition into its existing work flow. (See *Quartz Composer Programming Guide* for details on integrating compositions into applications.) Compositions are stored as Quartz Composer files with the `.qtz` extension.

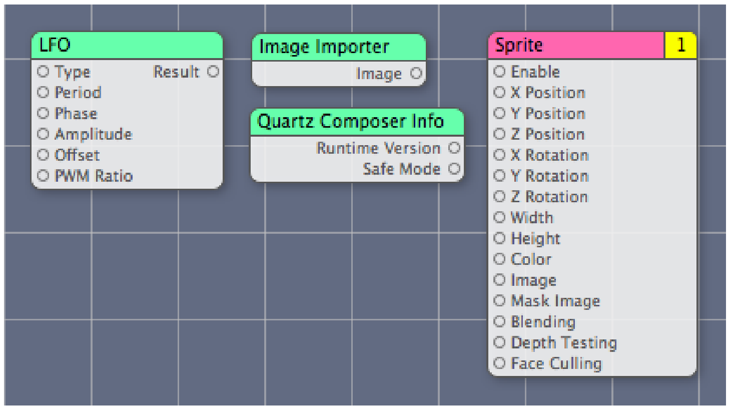
Patches

The basic elements of Quartz Composer are **patches**. Similar to routines in traditional programming languages, patches are base processing units. They execute and produce a result. Patches are equivalent to the following:

```
Result = function (time, {0 or more input parameters})
```

Unlike traditional routines, patches are visual entities (see Figure 1-3) that you add to a visual programming environment. Circles on a patch represent **ports**, with input ports on the left side of a patch and output ports on the right side. Ports pass data through them—you can think of ports as parameters.

Figure 1-3 Sample patches



Like routines, not all patches take input parameters. Figure 1-3 shows three patches that demonstrate the various configurations that ports can have. The Low Frequency Oscillator (LFO) patch has both input and output ports. The six input ports—Type, Period, Phase, Amplitude, Offset, and PWM Ratio—provide data that is used to calculate the amplitude of an oscillation at a specific time. The calculated amplitude value is available on the output port of the patch.

The Image Importer and Quartz Composer Info patches don't have any input ports, but each has an output port. The Image Importer patch produces an image, while the Quartz Composer Info patch produces a value that specifies the version of Quartz Composer running on the system. The Sprite patch, has many input ports, but no output port. Instead, the Sprite patch renders its result to a destination. Connections between the ports define how data flows when the composition runs.

Execution Modes

Patches are divided into groups that designate their execution mode—consumer, processor, or provider. A **consumer** renders a result to a destination. The Cube patch in Figure 1-1 (page 10) is an example of a consumer. It draws a textured cube to the screen. A consumer patch has these behaviors:

- Executes if its Enable input is set to `True`.
- Executes in a defined order. Note the number in the top-right of the Cube patch. This designates the execution order (also called the rendering layer) with respect to other consumer patches. Consumer patches are executed in numerical order from lowest to highest.
- Pulls data from processors and providers.

A **processor** processes data at specified intervals or in response to changing input values. The Interpolation patch in Figure 1-1 (page 10) is an example of a processor patch. It returns a value calculated by interpolating between a starting and ending value for a given time. The Interpolation patch updates its output value when input values or the time changes. In this case, the output value changes based on the duration of the interpolation and the repeat mode.

A **provider** supplies data from an outside source to a composition. This type of patch executes on demand—that is, whenever data is requested of it, but at most once per frame. The Video Input patch in Figure 1-1 (page 10) is an example of a provider patch. It supplies images captured from an external video source.

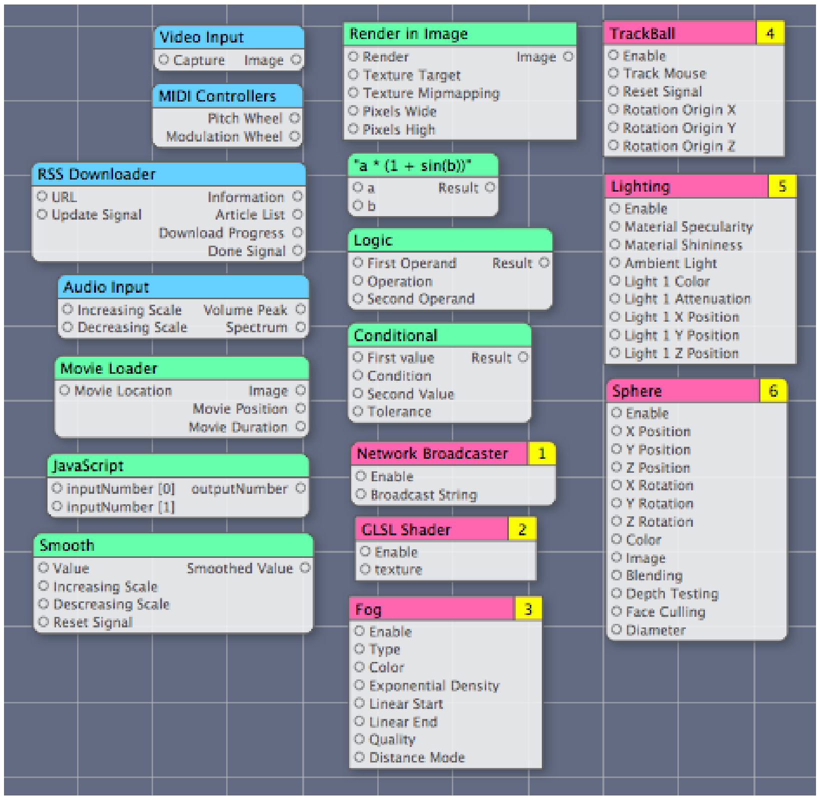
The title bars of patches are color coded to indicate their execution mode. Processors are green, providers are blue, and consumers are pink. Simply by looking at the color, you can determine the execution mode of each patch in Figure 1-1 (page 10).

The Patch Hierarchy

A Quartz composition is similar to any complex C or Objective-C program that has a main routine and many subroutines. The **root macro patch** is similar in concept to a main routine. A **macro patch** (or simply macro) is similar to a subroutine in a traditional program. Like subroutines, a macro can use (or call) another macro, which means macros can be nested, forming a **patch hierarchy** in a composition.

Quartz Composer provides several macro patches that require you to add subpatches to them. For example, the Lighting patch is a macro. To use this patch to illuminate an object, you place, inside the Lighting patch, the patches that create the object that you want to illuminate. You can also create custom macros. Packaging patch collections as macros keeps complex compositions manageable and easy to read. Macro patches look different from other patches. Macros have squared corners while other patches have rounded corners, as you can see by looking at the assortment of patches in Figure 1-4.

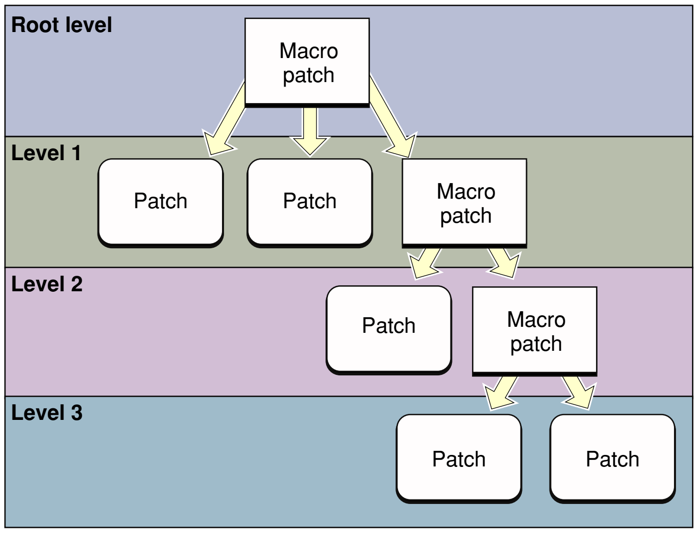
Figure 1-4 Macro patches have squared corners; other patches have rounded corners



The Evaluation Path

The Quartz Composer evaluation path determines when and how often each patch in a composition executes. When Quartz Composer executes a composition, it traverses the patch hierarchy, from the root macro patch level to lower levels, and attempts to execute the macro patches. Figure 1-5 shows the evaluation path for a composition with multiple levels. Evaluation begins at the root level with the macro patch. Quartz Composer must move to level 1 to obtain the data it needs for the macro patch at the root level. Level 1 contains a macro that must be evaluated, so evaluation moves to level 2. That level contains a macro, so evaluation moves to level 3. Level 3 does not contain any macros, so evaluation begins. After level 3 is evaluated, Quartz Composer moves to level 2 to complete that evaluation and then moves to level 1 to complete that evaluation.

Figure 1-5 The evaluation path for a hierarchical composition



From within a macro patch, Quartz Composer executes consumers first, beginning with the consumer whose rendering layer is lowest. Processor and provider patches execute when consumer patches pull data from them. Figure 1-6 shows the contents of a macro patch that renders a sprite. The mouse position controls the position of the sprite. A low-frequency oscillation controls the width and height of the sprite.

Figure 1-6 The contents of a macro patch that renders a sprite

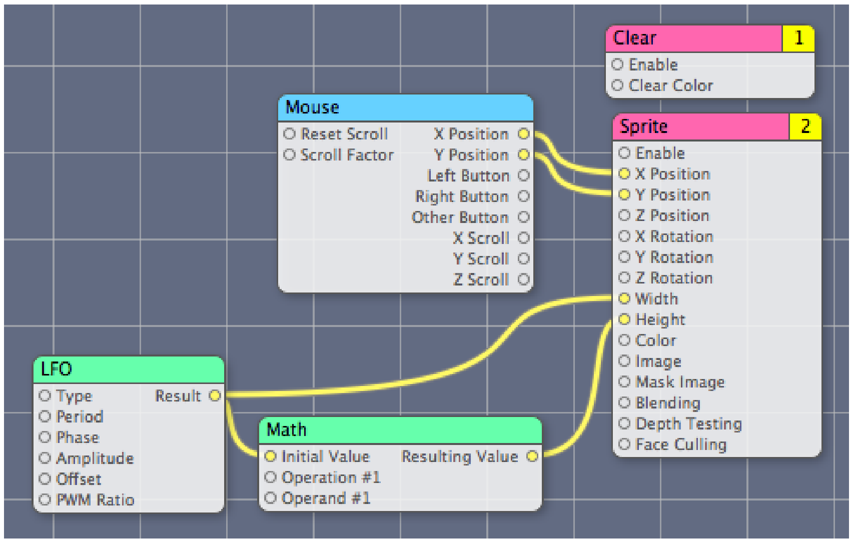
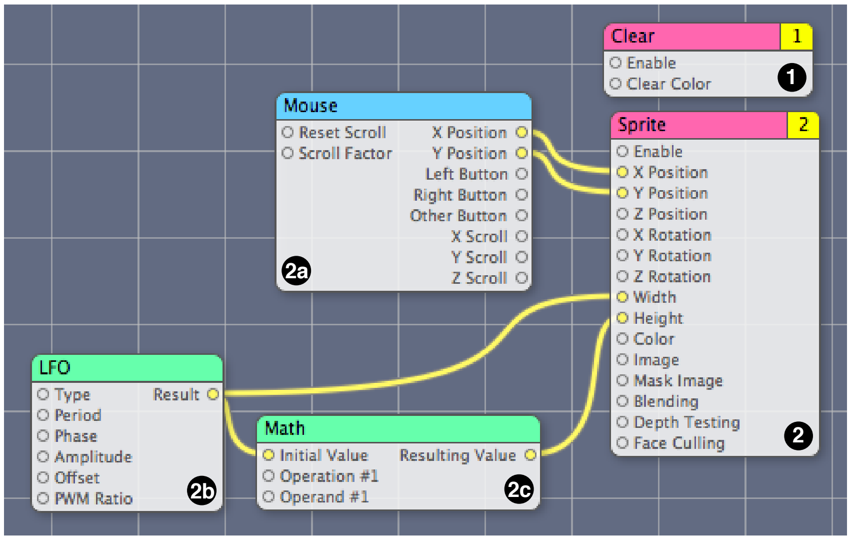


Figure 1-7 shows the evaluation order of the macro shown in Figure 1-6. There are two consumer patches—Clear and Sprite. Recall that consumers evaluate in numerical order, from lowest to highest. For example, in Figure 1-7, Clear evaluates first, then Sprite. This means that Quartz Composer clears the viewing area before rendering a sprite. But in order for the Sprite patch to complete its execution, the patch pulls data first from the Mouse patch, then from the LFO patch, and finally from the Math patch. After the Sprite patch has all the data it needs, it can then render its result.

Figure 1-7 Evaluation order

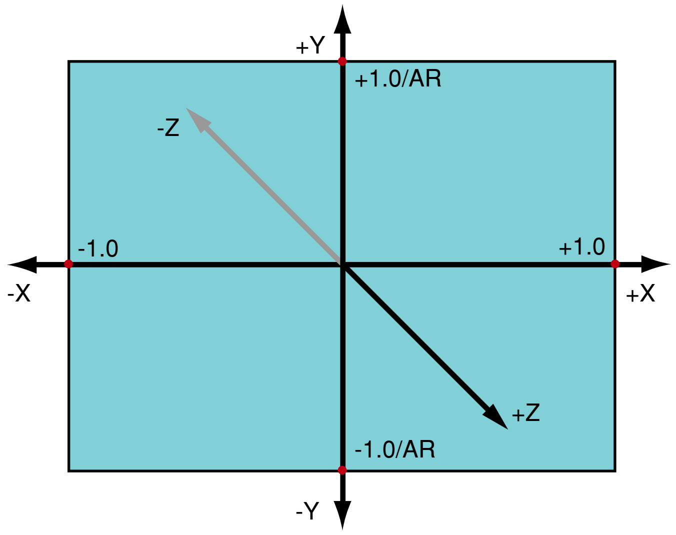


The Coordinate System

Quartz Composer uses a three-dimensional homogeneous coordinate system, as shown in Figure 1-8. The origin is at the center of the screen. The *x* axis is horizontal and the *y* axis is vertical. The *z* axis is orthogonal to the *x* and *y* axes, so that it comes out of the screen, towards the viewer. The left and right borders of the screen have coordinates `-1.0` and `+1.0`, respectively. (See the *x* axis in the figure.) The coordinates of the top and bottom borders (the *y* axis in the figure) depend on the screen aspect ratio (AR). In the case of a 4:3 aspect ratio, the values at the borders are `+1.0 / AR = +0.75` and `-1.0 / AR = -0.75`, respectively.

It's also possible to have an aspect ratio for which the top and bottom borders of the screen have coordinates `-1.0` and `+1.0`, respectively, while the left and right borders are `+1.0 / AR` and `-1.0 / AR`, respectively. For example, a 3:4 aspect ratio.

Figure 1-8 The Quartz Composer coordinate system



The Quartz Composer coordinate system maps to the rendering destination. The full width of a rendering destination is 2.0 units—one unit in the positive *x* axis added to one unit in the negative *x* axis. You can set the size of the graphics that are rendered. A size of 2.0 specifies to use the full width of the rendering destination. A size less than 2.0 units specifies to use a proportion of the rendering destination (1.0 specifies to use half, 1.5 specifies to use 75%, and so forth).

By default, the untransformed coordinate system is identical to the eye coordinate system. That is, the projection of a 3D object onto the 2D rendering destination is from the perspective of a viewer positioned directly in front of the monitor, with the *z* axis traveling perpendicular to the monitor.

Composition Repository

A Quartz composition provides a standard way to express motion graphics in Mac OS X, whether it's for animation or effects processing. For that reason, Mac OS X v10.5 includes a **composition repository**—a central location for storing compositions. Any application can, using the Quartz Composer framework, query the repository for specific types of compositions or simply browse the repository to see what's available. The repository resides across these folders:

- `/System/Library/Compositions`
- `/Library/Compositions`
- `~/Library/Compositions`

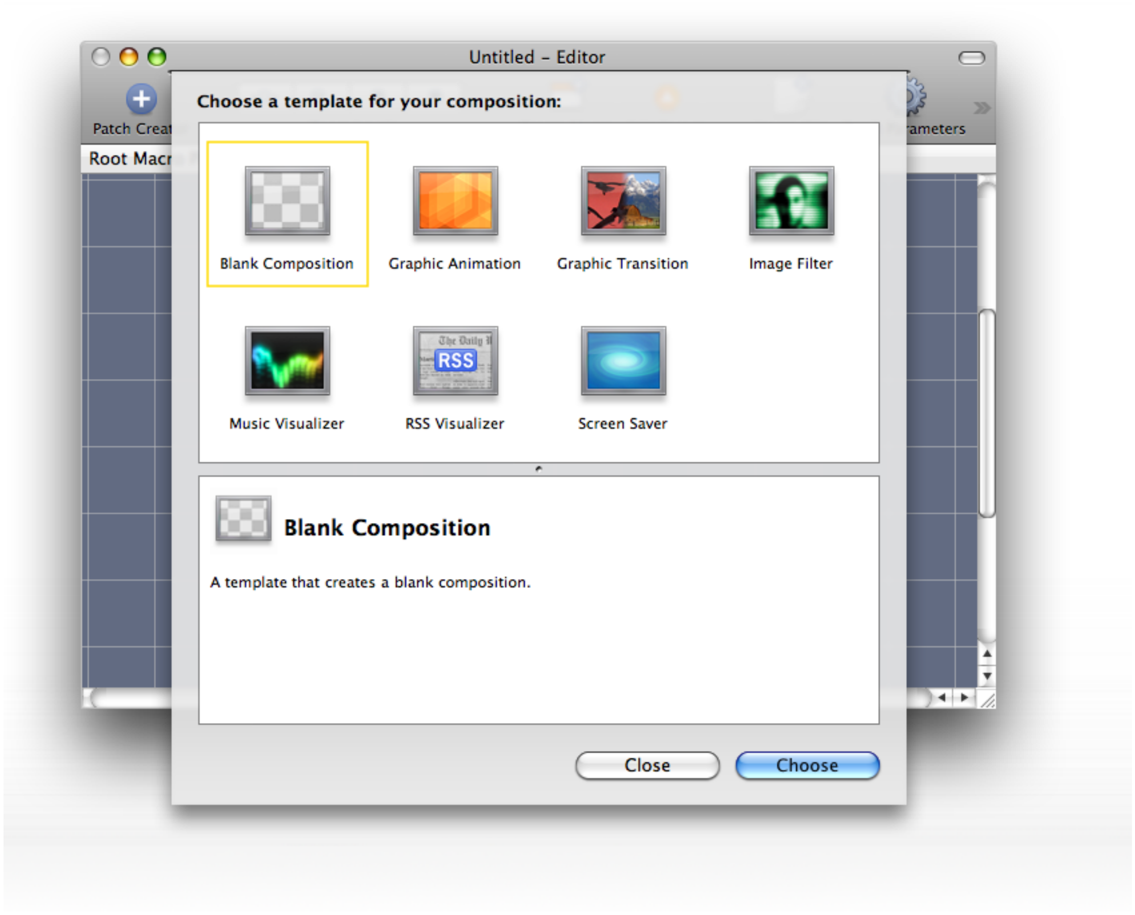
Any composition stored in the repository must conform to one of the protocols listed in Table 1-1. Quartz Composer provides templates for each of these protocols, which are available when you startup Quartz Composer (see Figure 1-9 (page 18)) or by choosing File > New From Template. When you choose a template in the Quartz Composer user interface, you'll see a detailed description for that protocol along with information about the required and optional parameters. After modifying a template, you can store the composition in the repository to allow other applications to use your composition.

Table 1-1 Composition protocols

Protocol	Required input parameters	Optional input parameters	Output parameters
Graphic animation	None	Primary color, secondary color, pace, and preview mode	None required, but must render to the screen
Graphic transition	Source image, destination image	Preview mode	None required
Image filter	Image	Position of the center of the effect, preview mode	Image
Music visualizer	Audio peak, audio spectrum	Track info, track position, track signal	None required
RSS visualizer	URL for an RSS feed	Display duration	None required
Screen saver	None	Screen image, preview mode	None required, webpage URL optional

The Quartz Composer programming interface provides Objective-C classes (`QCCompositionRepository` and `QCComposition`) that allow developers to access the repository programmatically and provide support within an application for browsing and choosing compositions. You do not need to know Objective-C to create a composition for the repository, but you must know Objective-C to access the repository programmatically. The Quartz Composer programming interface makes it easy for developers to support the sort of motion graphics that application like iMovie and iDVD use. For more information, see *Quartz Composer Reference Collection*.

Figure 1-9 Templates available in Quartz Composer



Blank Composition

A template that creates a blank composition.

The Quartz Composer User Interface

The Quartz Composer development tool has two main windows: the editor and the viewer. The editor window is for assembling and connecting patches to create a composition. The viewer window shows the output produced by the composition, the rendering resources consumed by the composition, and debugging information.

This chapter describes the editor and viewer windows, the utility windows and views, the key menu items, and other aspects of the user interface for Quartz Composer. After reading this chapter, you'll be ready to use the editor to create a composition.

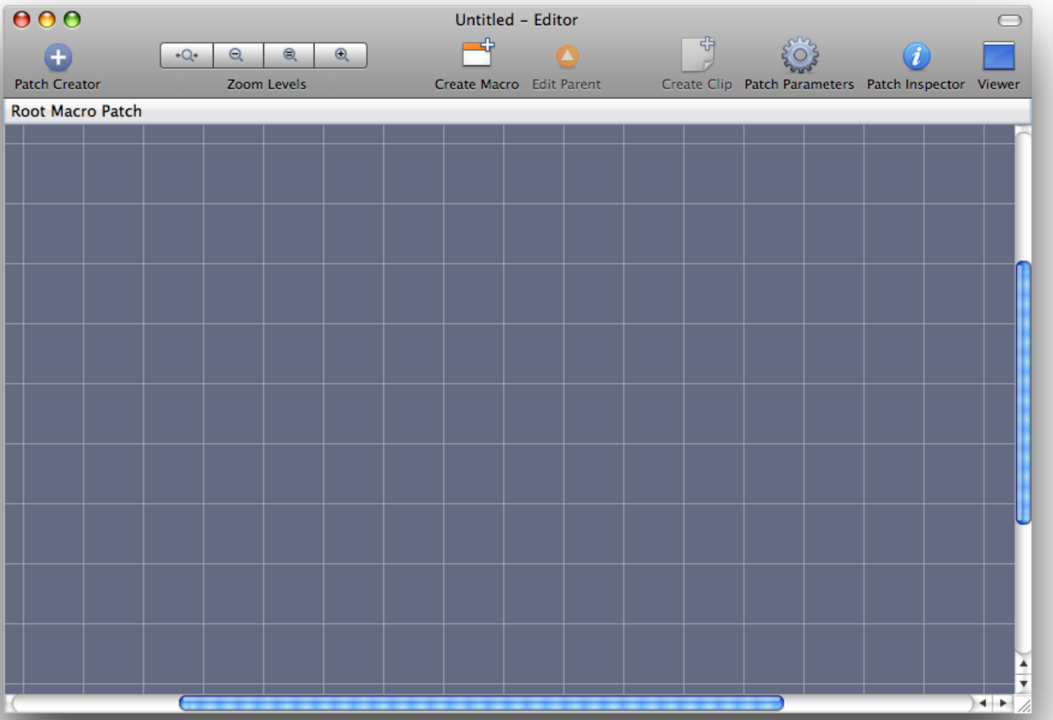
The Editor Window

The main part of the Quartz Composer Editor window is the **workspace**—a grid for assembling and connecting patches (see Figure 2-1). The set of connected patches on the workspace is sometimes called a **graph**. In theory, the workspace is infinite in size. The size of the scrollers in the scroll bars provides a clue as to the size of the graph.

The **toolbar** at the top of the editor window contains controls that either open utility windows or modify what's shown in the editor window.

The Patch Creator button opens the Patch Creator utility window. You can browse patch names and categories in this utility window, or you can use the search field to look for patches. For more details see Figure 2-4 (page 23).

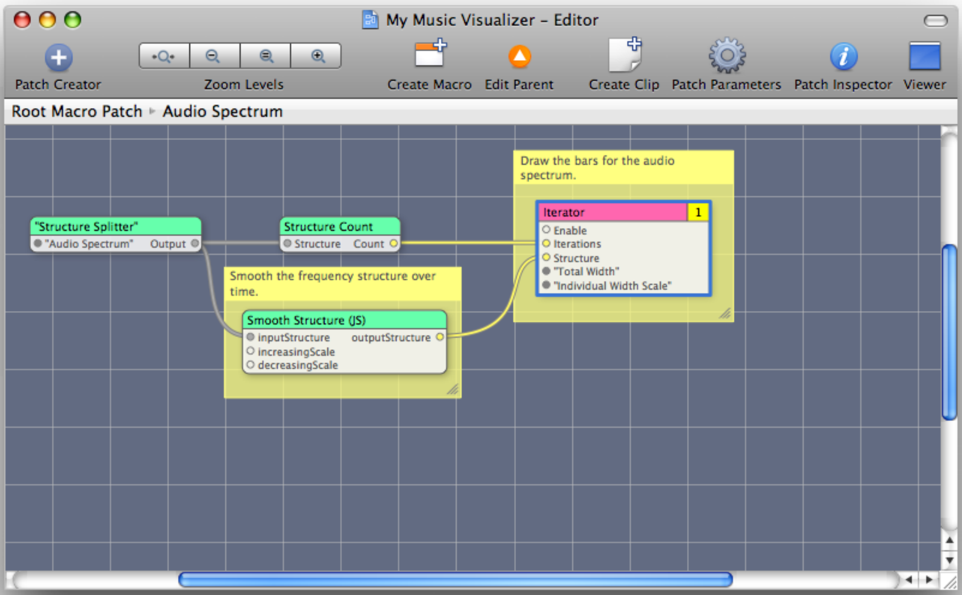
Figure 2-1 The Editor window



The Zoom Levels controls come in handy for complex compositions that contain a lot of patches. You can zoom instantly to fit all patches in the visible area of the workspace, zoom out, return to the default magnification, or zoom in to read the text more easily.

The Create Macro and Edit Parent buttons support compositions that have nested patches (or macros). Clicking Create Macro adds an empty macro patch to the workspace. To see what's inside or to edit a macro, simply double-click it. If you are in a macro, you can go up one level by clicking the Edit Parent button. Figure 2-2 (page 21) shows the content of an audio spectrum macro. You can tell you are inside a macro by looking at the space just below the toolbar, which displays the breadcrumb, or path. You can also change levels by clicking an item in the path.

Figure 2-2 A macro that draws an audio spectrum



A **clip** contains a set of patches that perform a task, such as rendering a rotating cube or an animated background. Clips, like routines in a library for a traditional programming language, speed development. They are prepackaged "mini" compositions that you can drag into your composition and customize for your own use. Quartz Composer provides just a few clips but you can add your own by clicking the Create Clip button. See "Adding a Clip to the Patch Creator" (page 48).

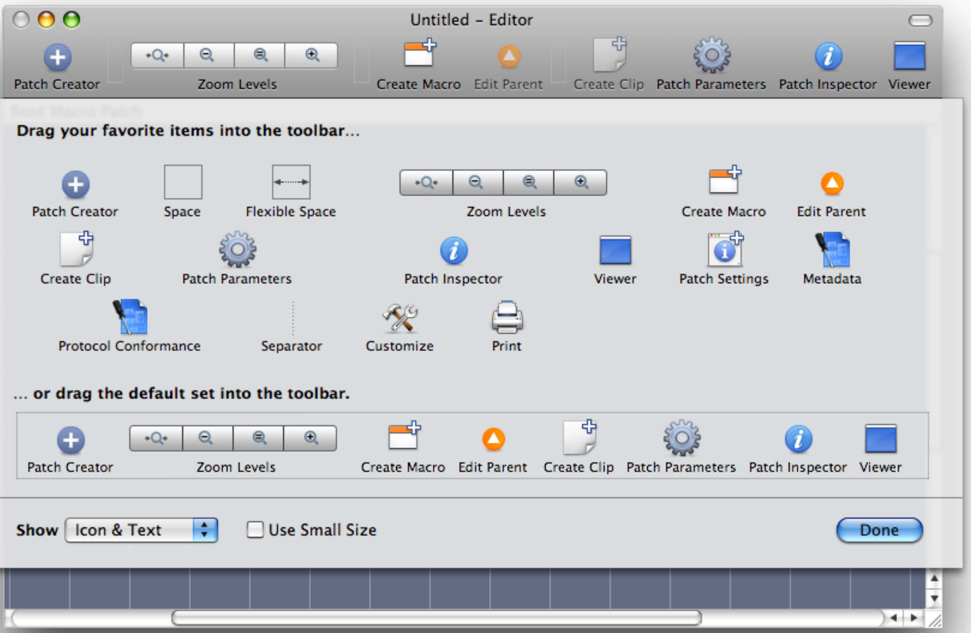
The Patch Parameters button toggles a view that contains the input parameters of the selected patch. See "Patch Parameter Pane" (page 30).

The Patch inspector button opens a utility window that lets you set input parameters, view and edit patch information, change internal settings for patches that have them, and inspect published input and output ports if there are any. See "The Patch Inspector" (page 31).

The Viewer button opens a window that shows the rendered output from the composition. See "The Viewer Window" (page 35).

Control-click the toolbar to open a sheet that allows you to customize the toolbar. You can drag items into and out of the toolbar, as well as rearrange them.

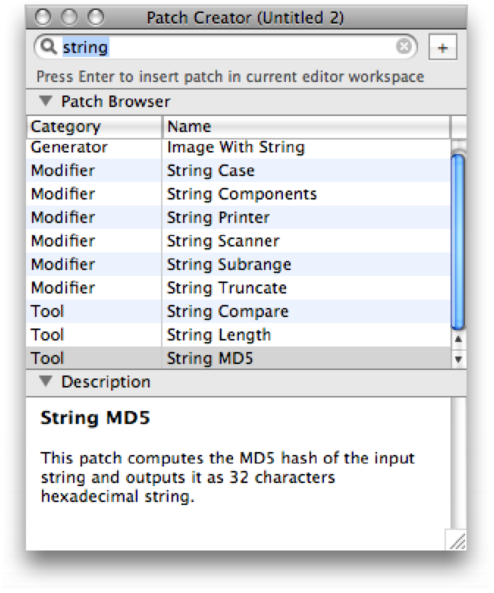
Figure 2-3 The sheet for customizing the toolbar



The Patch Creator

The Patch Creator is a utility window for browsing and getting information about Quartz Composer patches and clips. Within this window is the Patch Browser, which lists patches and clips by name and category. If you are new to Quartz Composer, you can use the browser to familiarize yourself with the available patches. When you click a patch name, you can read its description. (See Figure 2-4.)

Figure 2-4 The Patch Creator


Patch Creator (Untitled 2)
string
+
Press Enter to insert patch in current editor workspace
Patch Browser
Description

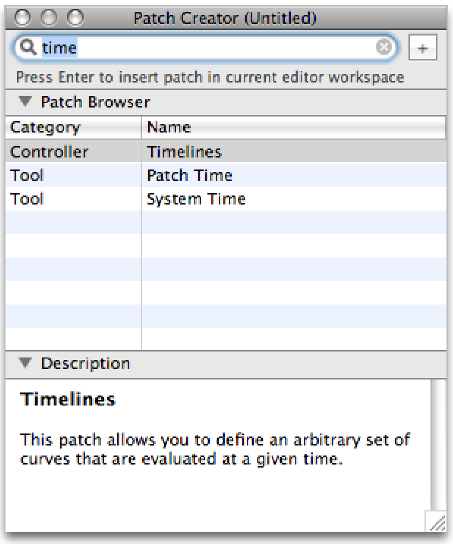
Category	Name
Generator	Image With String
Modifier	String Case
Modifier	String Components
Modifier	String Printer
Modifier	String Scanner
Modifier	String Subrange
Modifier	String Truncate
Tool	String Compare
Tool	String Length
Tool	String MD5

String MD5

This patch computes the MD5 hash of the input string and outputs it as 32 characters hexadecimal string.

If you already know the name of a patch, the fastest way to locate it is to start typing its name in the search field. As you type, the list narrows to those patches whose name contains the typed string. Figure 2-5 shows the list of patches after a search for the key word "time."

Figure 2-5 The search feature can help you to locate patches



Category	Name
Controller	Timelines
Tool	Patch Time
Tool	System Time

Timelines

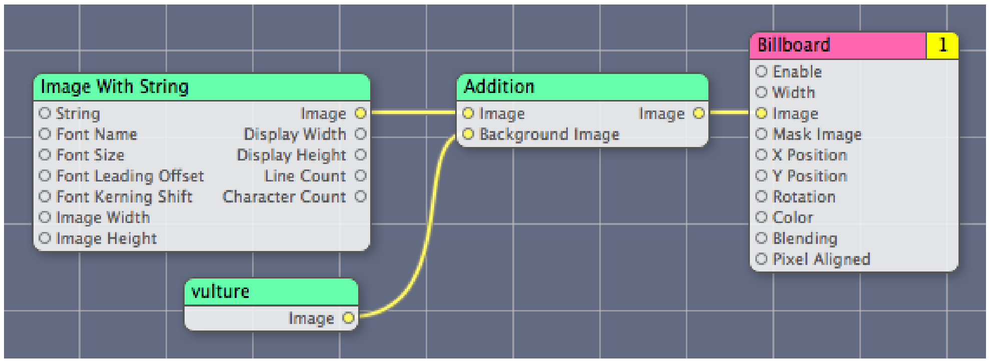
This patch allows you to define an arbitrary set of curves that are evaluated at a given time.

Because Quartz Composer supports custom patches and clips, it's possible for the list of patches in the Patch Creator to change. The best way to find out what's available is to browse through the list. However, you might find it helpful to know more about the patch categories. The following sections describe the categories and highlight some of the more interesting and powerful patches.

Composite Patches

Composite patches take two input images and produce one output image. Quartz Composer provides patches for the commonly used compositing and blending operations (addition, source over, source in, source out, and so on). Figure 2-6 shows an addition patch that produces an image that's rendered to a 2D billboard.

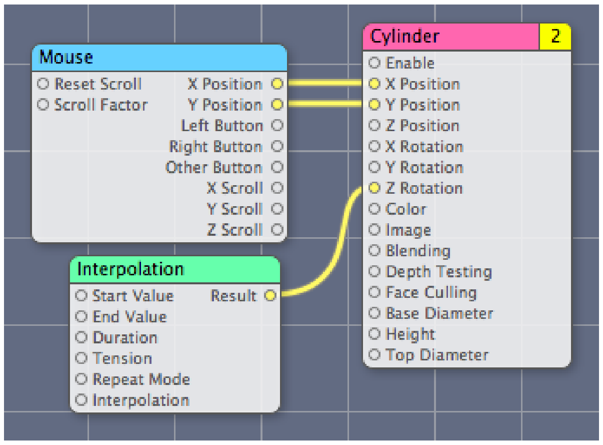
Figure 2-6 The Addition patch is a Composite patch



Controller Patches

Controller patches produce one or more values that you can use to control the output of another patch. They can produce output values based on user input or as the result of mathematical operations, such as functions you supply, values from waveforms, and randomly generated values. Figure 2-7 shows the x and y position of the mouse used to control the position of a cylinder onscreen, while an Interpolation patch provides a value to control the rotation of the cylinder in the *z* plane.

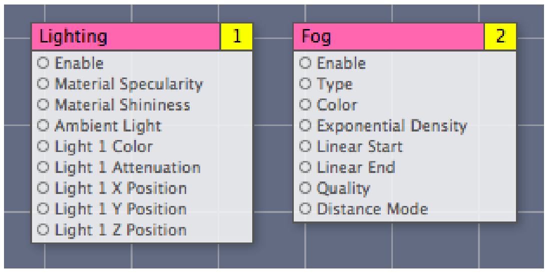
Figure 2-7 The Mouse and Interpolation patches are Controller patches



Environment Patches

Environment patches are typically macros that operate on a motion graphic to transform its look in some way. The Lighting and Fog patches in Figure 2-8 are two examples, but there are many more, including patches that transform 3D space and macro patches that replicate their contents.

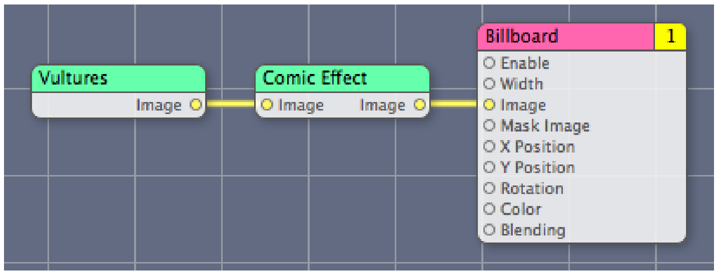
Figure 2-8 The Lighting and Fog patches are Environment patches



Filter Patches

Filter patches operate on the pixels in an input image to produce an output image. The Comic Effect patch in Figure 2-9 is a Filter patch that operates on a single image to make the image look as if it is published in a comic book. There are dozens of Filter patches, including Pixellate, Gloom, Unsharp Mask, and Edges.

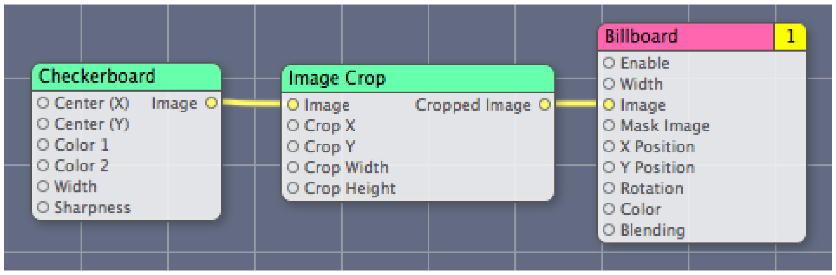
Figure 2-9 Comic Effect is an image processing filter patch



Generator Patches

Generator patches produce an image algorithmically. The Checkerboard patch shown in Figure 2-10 produces a checkerboard pattern whose colors, square size, and sharpness are based on input values. If there aren't any external input values, the patch uses its default values. Other Generator patches include Image with String, Star Shine, and Random Generator.

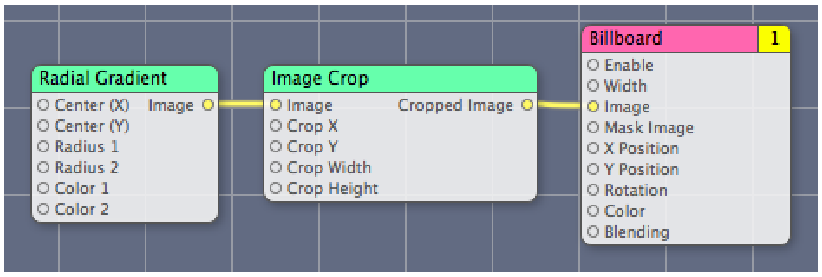
Figure 2-10 Checkerboard is a Generator patch



Gradient Patches

Gradient patches produces an image based on Gaussian, linear, or radial algorithms. Note that Quartz Composer also has a patch named "Gradient" in the render category that renders a gradient to a destination. Patches in the Gradient category, by contrast, must produce an output image that can then be used as input to another patch. The Gradient category also has Linear Gradient and Gaussian Gradient patches.

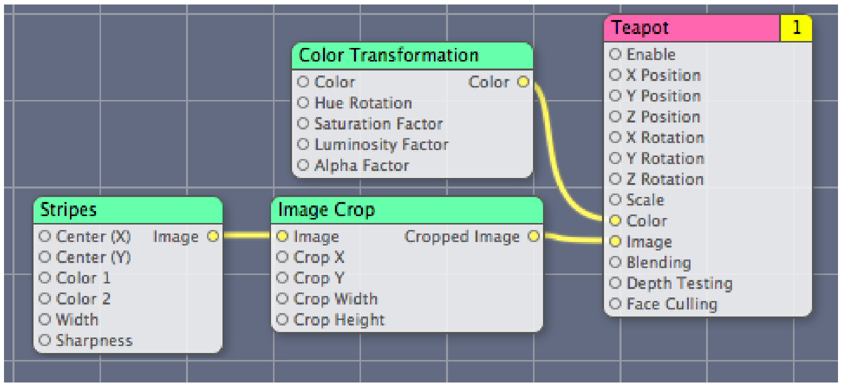
Figure 2-11 Radial Gradient is a Gradient patch



Modifier Patches

Modifier patches take input values or objects (such as images), change or transform the input in some way, and then output the transformed value or object. The Color Transformation and Image Crop patches in Figure 2-12 are Modifier patches. There are also patches for modifying strings (such as String Case and String Truncate), inspecting structures (such as Structure Index Member and Structure Key Member), and modifying textures (Image Texturing Properties).

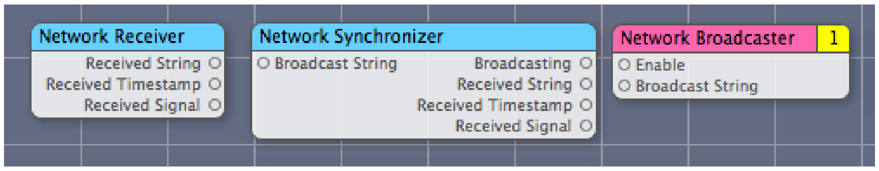
Figure 2-12 Image Crop and Color Transformation are Modifier patches



Network Patches

Network patches send, receive, or operate on network data. Figure 2-13 shows three examples: Network Receiver, Network Synchronizer, and Network Broadcaster.

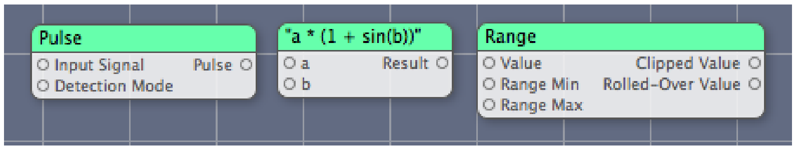
Figure 2-13 Some sample Network patches



Numeric Patches

Numeric patches produce values based on a mathematical expression that you supply or that is built in to the patch. The center patch in Figure 2-14 is one that allows you to define an expression. Quartz Composer automatically creates the input ports based on the expression. There are many more helpful Numeric patches to explore, such as Counter, Conditional, and Round.

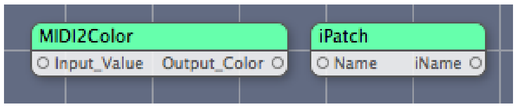
Figure 2-14 Some sample Numeric patches



Plug-in Patches

The Plug-in category includes custom patches packaged as a Quartz Composer plug-in. For information on creating them, see *Quartz Composer Custom Patch Programming Guide*. You won't find the MIDI2Color and iPatch patches shown in Figure 2-15 unless you build them yourself, which you can do by following the instructions in *Quartz Composer Custom Patch Programming Guide*.

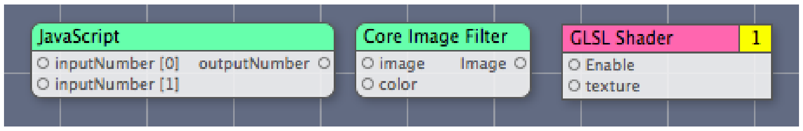
Figure 2-15 Two custom patches



Programming Patches

Programming patches (see Figure 2-16) provide a text field, accessible in the inspector, for adding a routine that the patch executes. The JavaScript patch requires you to use JavaScript. The Core Image Filter patch requires that you provide a kernel routine written using the Core Image Kernel Language (see *Core Image Kernel Language Reference*). The GLSL Shader patch requires that you supply a shader written using OpenGL Shading Language. For more details, see "Using Programming Patches" (page 49).

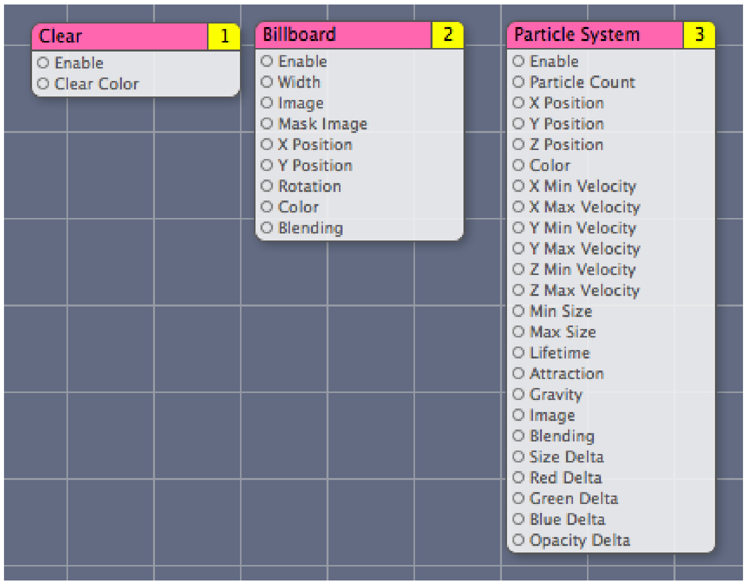
Figure 2-16 Some example Programming patches



Renderer Patches

Renderer patches render to a destination—the viewer window. Figure 2-17 shows three of them—Clear, Billboard, and Particle System. The Clear patch clears the rendering destination, optionally using a color you supply. The Billboard patch renders a 2D image. The Particle System patch renders a set of particles, each of which have a lifetime. There are several other interesting renderers, including Cube, Psychedelic, Sprite, Sphere, and Teapot.

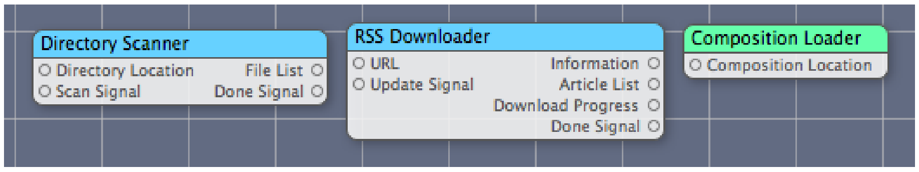
Figure 2-17 A few Renderer patches



Source Patches

Source patches provide data (such as a file list, RSS feed, composition) from a source. Figure 2-18 shows Directory Scanner, RSS Downloader, and Composition Loader. But there are also patches for getting audio input, images from folders, and information about the computer.

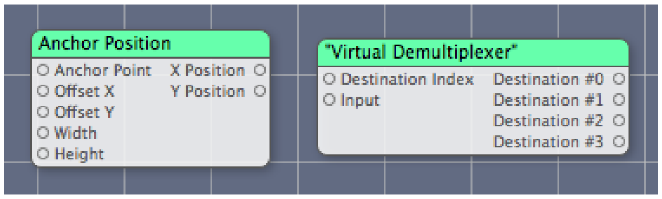
Figure 2-18 Some sample Source patches



Tool Patches

Tool patches perform tasks that are useful for creating a composition, such as the Anchor Position and Demultiplexer patches shown in Figure 2-19. (The title of the Demultiplexer patch in the figure is in quotes to show that the title was created dynamically. The title changes according to the output port type you choose.) There are many other tool patches including OpenGL Info, Image Dimensions, Input Splitter, Date Formatter, Structure Count, and System Time.

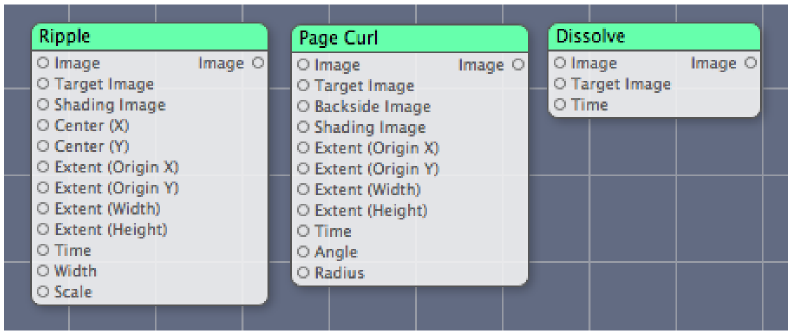
Figure 2-19 Two Tool patches



Transition Patches

Transition patches produce an effect between a source image and a destination image—the sort of effect you would use in a slideshow. Ripple, Page Curl, and Dissolve, shown in Figure 2-20, are three such patches. There are others, including Flash, Swipe, Mod, Copy Machine, and Disintegrate with Mask.

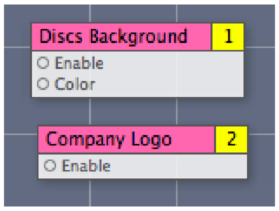
Figure 2-20 Three Transition patches



Clip Patches

Clip patches are those that you create by selecting a group of connected patches and clicking Create Clip. Quartz Composer provides a few clips, such as the Discs Background shown in Figure 2-21 and the Rotating Cube clip, which you'll use if you follow the tutorial in "Creating a Glow Filter" (page 55). Quartz Composer doesn't provide the Company Logo clip, also shown in Figure 2-21; a logo or other graphic that you repeatedly use is a good candidate for a clip.

Figure 2-21 Two Clip patches

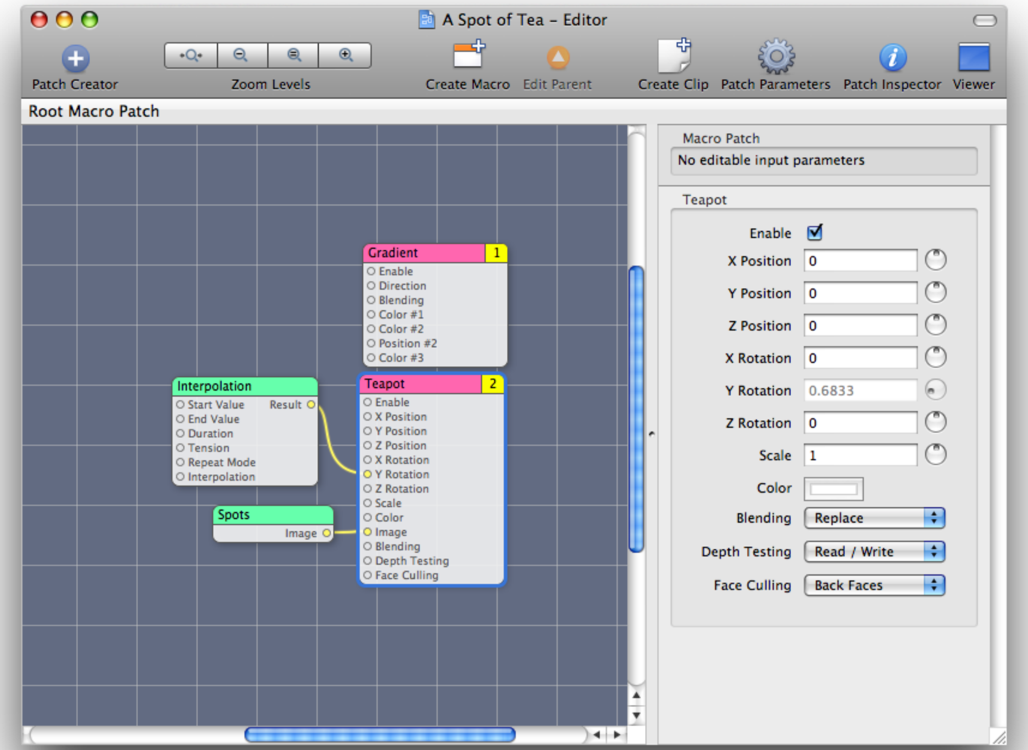


Patch Parameter Pane

The Patch Parameters button on the editor window toolbar toggles the patch parameters pane shown to the right of the workspace in Figure 2-22. This pane provides convenient access for editing input parameters

The Macro Patch field at the top of the pane shows the input parameters for any published ports for that level. (You can find out more about published ports in "Publishing Ports" (page 63).) The pane also shows the input parameters for the patch, or patches, currently selected in the workspace. When you select another patch, the pane changes to the input parameters for that patch. If you select more than one patch, the input parameters for all selected patches appear in the pane, making it easy to change parameters in different patches.

Figure 2-22 The patch parameter pane



Macro Patch

No editable input parameters

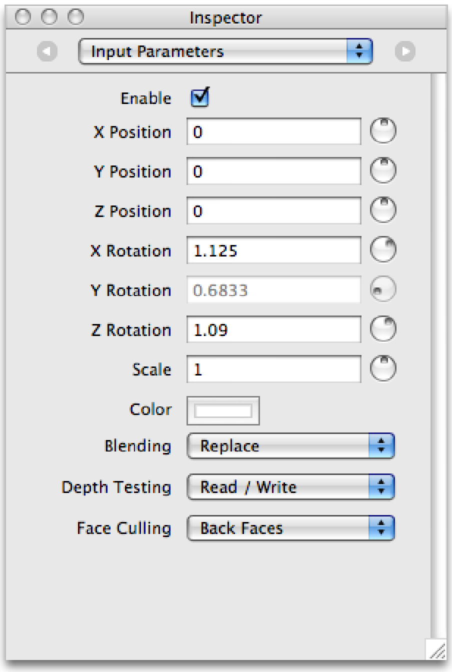
Teapot

Enable	✓
X Position	0
Y Position	0
Z Position	0
X Rotation	0
Y Rotation	0.6833
Z Rotation	0
Scale	1
Color	⬜
Blending	Replace
Depth Testing	Read / Write
Face Culling	Back Faces

The Patch Inspector

The Patch inspector is a utility window that has one to three panes, depending on the patch that's currently selected in the workspace. All patches have an Input Parameters pane. The Input Parameters pane (see Figure 2-23) allows you to edit the input parameters for a patch. It's equivalent to the patch parameter pane, which you view within the editor window.

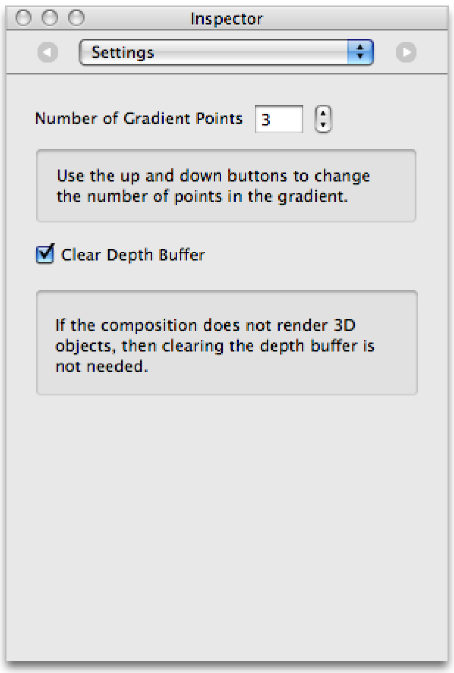
Figure 2-23 The Input Parameters pane



Enable ✓
X Position 0
Y Position 0
Z Position 0
X Rotation 1.125
Y Rotation 0.6833
Z Rotation 1.09
Scale 1
Color
Blending Replace
Depth Testing Read / Write
Face Culling Back Faces

Some patches also have a Settings pane. You'll see it only if the patch has settings that are advanced or that aren't typically animated. Figure 2-24 shows an advanced option for clearing the depth buffer along with an explanation of why you might use the option. It also provides an option for setting the number of gradient points.

Figure 2-24 The Settings pane



Number of Gradient Points 3

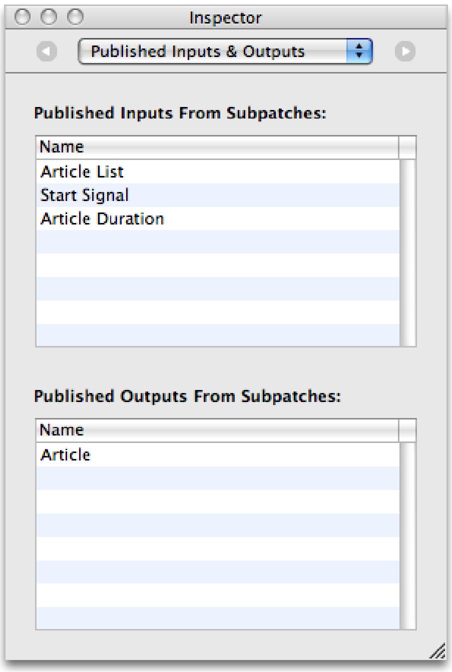
Use the up and down buttons to change the number of points in the gradient.

☑ Clear Depth Buffer

If the composition does not render 3D objects, then clearing the depth buffer is not needed.

Quartz Composer allows you to publish the input and output ports for a patch. Publishing allows you to access values for patches that are nested within a composition (see "The Patch Hierarchy" (page 13)). You can inspect the published ports using the Published Inputs & Outputs pane of the inspector. Figure 2-25 shows the published input and output ports for a macro patch from a rather complex, hierarchical composition. Publishing makes these ports available to the parent level of this patch.

Figure 2-25 The published ports for a macro patch



Published Inputs From Subpatches:

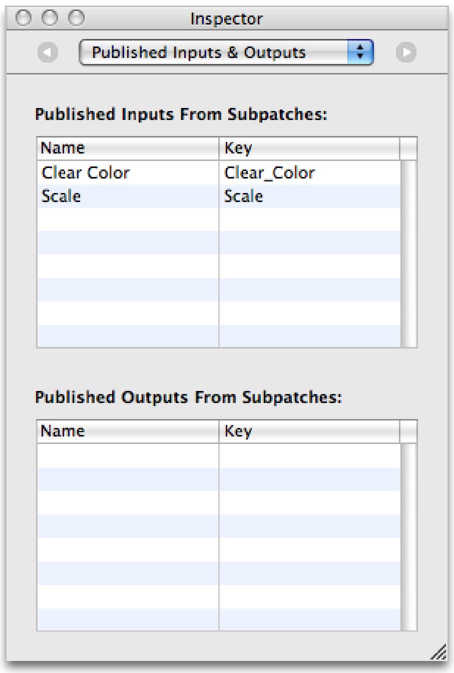
Name
Article List
Start Signal
Article Duration

Published Outputs From Subpatches:

Name
Article

Ports published at the topmost level of a composition are available outside the composition, as you'll see in "Publishing Ports" (page 63). You can inspect the ports published at the topmost level (that is, the root macro patch) by opening the inspector and clicking the workspace (make sure that no patches are selected). Then, the inspector provides information about the entire level of the composition rather than about a single patch.

Figure 2-26 The published ports for the root macro patch, or top level of a composition



Published Inputs From Subpatches:

Name	Key
Clear Color	Clear_Color
Scale	Scale

Published Outputs From Subpatches:

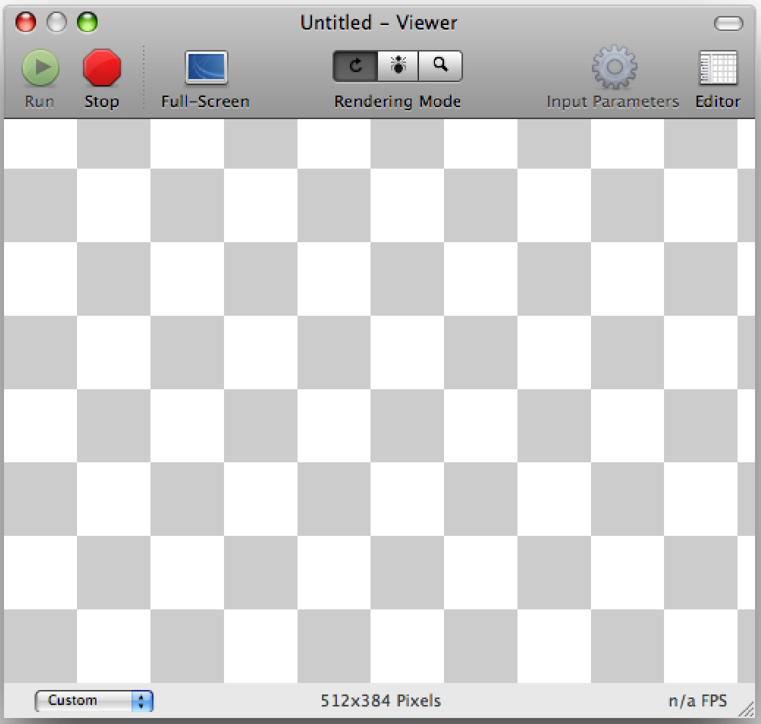
Name	Key

The Viewer Window

The viewer window has its own toolbar. You can start and stop rendering, switch to full-screen rendering, change the rendering mode (see "Rendering Modes" (page 37)), view composition input parameters, and switch to the editor window.

Tip: To exit from full-screen mode, press the Escape key.

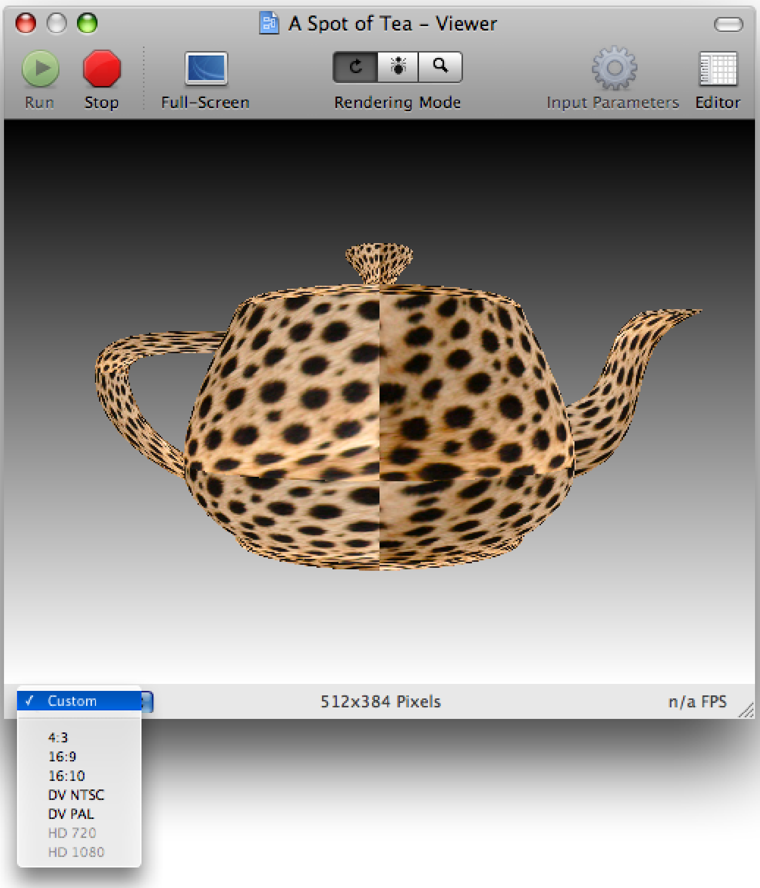
Figure 2-27 The Quartz Composer viewer window



The viewer window shows the average frame rate at the bottom right of the window and the image size at the bottom center. The pop-up menu at the bottom left lets you constrain the viewer to a fixed size or aspect ratio, as shown in Figure 2-28 (page 37).

Tip: You can manage viewer sizes and aspect ratios in Quartz Composer Preferences. See "Setting Preferences" (page 43).

Figure 2-28 The aspect ratio pop-up menu

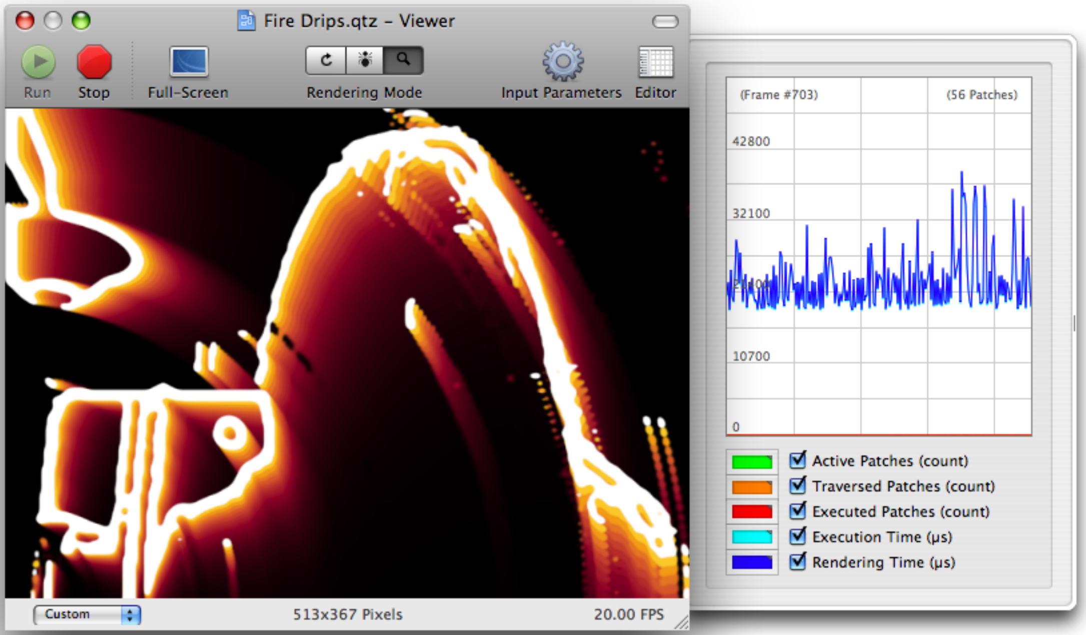


Rendering Modes

The viewer window has three rendering modes available:

- Performance mode is the normal rendering mode and provides optimal performance. When you enable full-screen mode, Quartz Composer always uses performance rendering mode.
- Profile mode gathers statistics that are displayed in a drawer to the right of the viewer window (see Figure 2-29).
- Debug mode displays a log of debug information that is displayed in a drawer below the viewer window (see Figure 2-30) and color codes each patch to indicate its execution state. Notice that the viewer in the figure isn't rendering content. The content in the drawer provides clues as to the underlying problem.

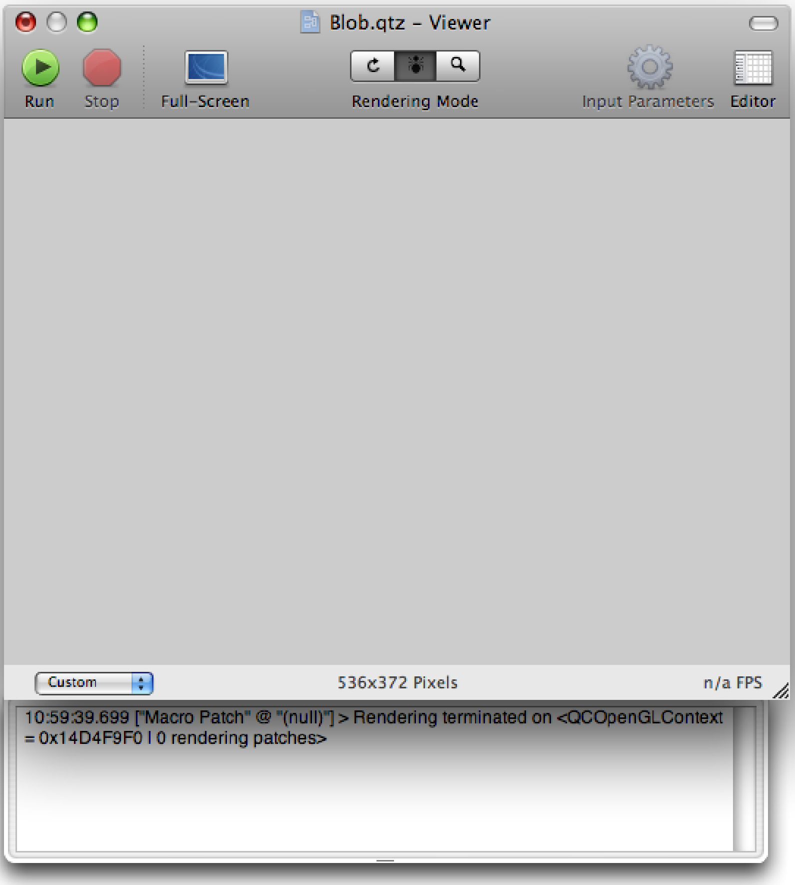
Figure 2-29 Profile rendering mode



Profile rendering mode analyzes each frame that is rendered and graphs information that you can use to optimize the rendering performance of a composition. You can obtain the following information:

- Active Patches, the number of patches that are currently active.
- Traversed Patches, the number of patches that are "touched" during the patch tree traversal, but not necessarily executed.
- Executed Patches, the number of patches executed for the analyzed frame.
- Execution Time, an estimate of the length of time in milliseconds used by Quartz Composer to compute the analyzed frame.
- Rendering Time, an estimate of the length of time in milliseconds used by Quartz Composer and OpenGL to render the analyzed frame.

Figure 2-30 Debug rendering mode



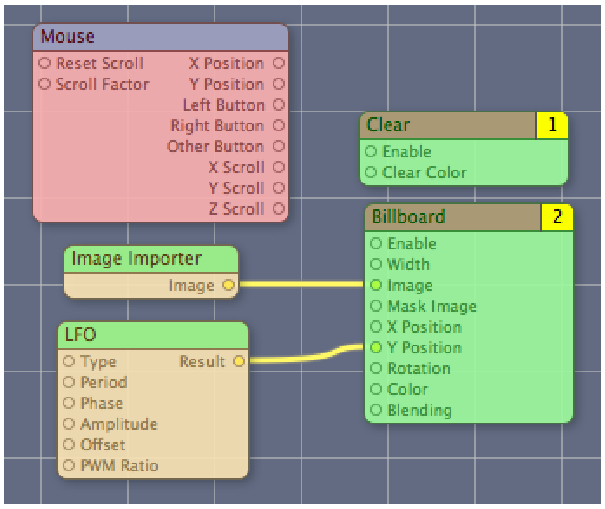
```
10:59:39.699 ["Macro Patch" @ "(null)"] > Rendering terminated on <QCOpenGLContext
= 0x14D4F9F0 | 0 rendering patches>
```

In addition to displaying debugging information posted by Quartz Composer, **debug rendering mode** shades each patch in the editor window with a color, similar to the shading shown in Figure 2-31.

- Green shading indicates that the patch is enabled and running. For example, the Clear and Billboard patches in the figure are running.
- Red shading indicates that the patch is not enabled. The Mouse patch in the figure is not enabled.
- Orange indicates that a patch is enabled but not running. The Image Importer and LFO patches are enabled but not running.

You can examine the data flow as a composition runs in debugging mode. Patches change colors as they move from one state to another.

Figure 2-31 The debug rendering mode color-codes patches



Menus

Although you will want to explore all the menu items available in Quartz Composer, there are several menu items that you won't want to overlook:

- File > Export as QuickTime Movie saves a composition as a QuickTime movie. See "Turning a Composition into a QuickTime Movie" (page 67) for details.
- Edit > Undo and Edit > Redo.
- File > Compare Compositions opens a window that lets you compare compositions in a number of interesting ways. See "Comparing Compositions" (page 46).
- Editor > Edit Information allows you to add information about the composition, such as the copyright and a description.
- Viewer > Save Snapshot saves an image of the composition that's rendering in the viewer.

Basic and Advanced Tasks, Tips, and Tricks

This chapter describes how to use the editor to add patches to the workspace and connect them, inspect the values of input and output ports, and publish ports. You'll learn tips and shortcuts for getting the Quartz Composer development environment to support your working style. You'll see how to use some of the advanced features including templates and patches that process code.

Adding Patches

There are three ways to add a patch from the Patch Creator:

- Press Return to add the patch whose name is selected.
- Double-click a patch name.
- Drag a patch name to the workspace.

To duplicate a patch that's already in the workspace, select the patch and press Command-D.

Tip: Instead of adding an Image Importer patch, you can drag an image from the Finder directly to the workspace. Quartz Composer automatically creates an instance of the Image Importer patch and uses the name of the image as the title of the patch. Quartz Composer provides a similar convenience for movies. Instead of adding a Movie Loader patch, simply drag a QuickTime move from the Finder to the workspace.

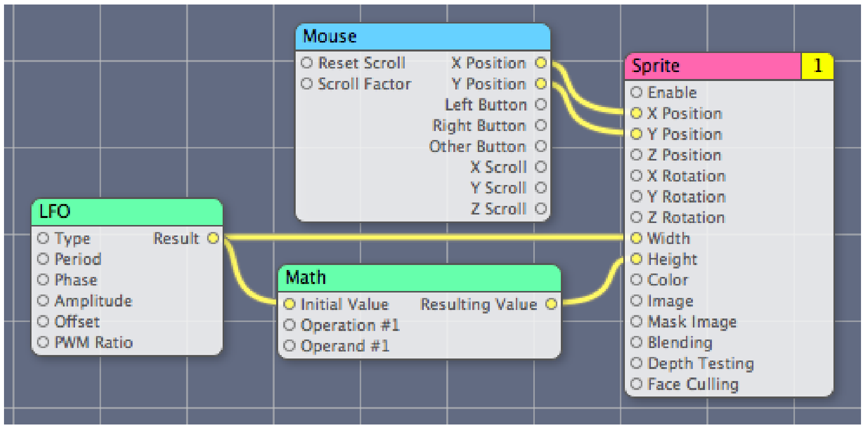
Making Connections Between Patches

Connections between input and output ports of different patches establish a data flow in a composition. Simply click the output port, then click the input port you want to connect—you don't need to hold down the mouse button while you move to the input port. Note that you can establish a connection only between ports that have compatible types—they must have the same type or a type conversion must be possible.

To break a connection between patches, either double-click the input port or drag the connector and release it. The connector disappears. Figure 3-1 shows some typical connections between patches.

Tip: You can redirect an existing connection by clicking the connection at the input port and then clicking another compatible input port. This action is similar to dragging, but you can release the mouse button while you move to the input port.

Figure 3-1 Patch connections



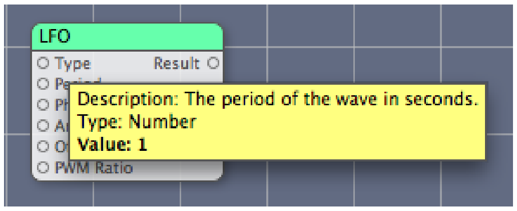
The output from a patch can connect to the input of more than one patch. For example, the Result output port of the LFO patch (Figure 3-1) provides input to the Math patch and to the Sprite patch.

Tip: You can connect the output of a port to multiple input ports by holding down the Option key while clicking on each of the input ports.

Inspecting Port Values

Hover the pointer over a port to view a short description of the port, its data type, and the current value. (See Figure 3-2.) If the input port is not an object port, you can edit the port value by double-clicking the circle associated with the port, entering the new value in the text editor, and then pressing the Return key to validate the value or the Escape key to cancel. Note that input ports have a default value, which is always null for object ports.

Figure 3-2 A help tag for an input port

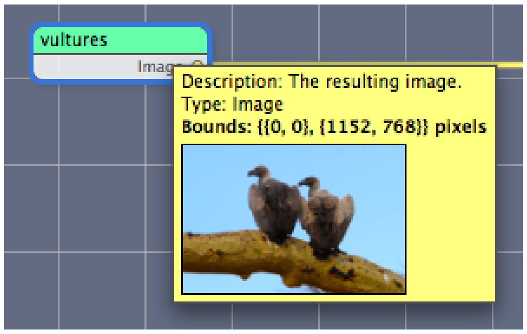


Description: The period of the wave in seconds.
Type: Number
Value: 1

You can also inspect and set the values for input ports by opening the inspector and displaying the Input Parameters pane or by clicking the Patch Parameters button in the editor to toggle the input parameters pane.

You can't set the value for output ports, but you can view them. Image ports show a thumbnail representation of the image, as shown in Figure 3-3.

Figure 3-3 Image ports display a thumbnail image

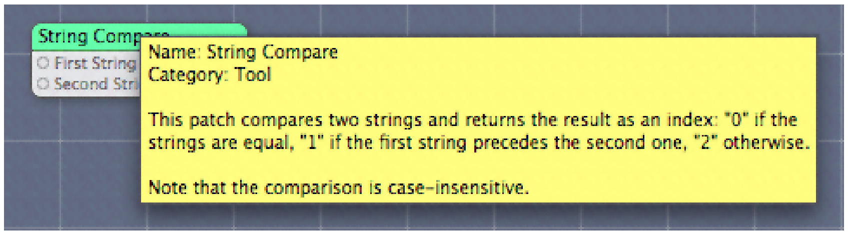


Description: The resulting image.
Type: Image
Bounds: {{0, 0}, {1152, 768}} pixels

Finding Out What a Patch Does

When you select a patch in the Patch Creator, its description appears in the Description pane. But if you are working with a patch in the workspace, you can hover the pointer over the patch title bar to see a help tag that contains the name, category, and patch description, as shown in Figure 3-4.

Figure 3-4 The patch description appears in a help tag



Name: String Compare
Category: Tool

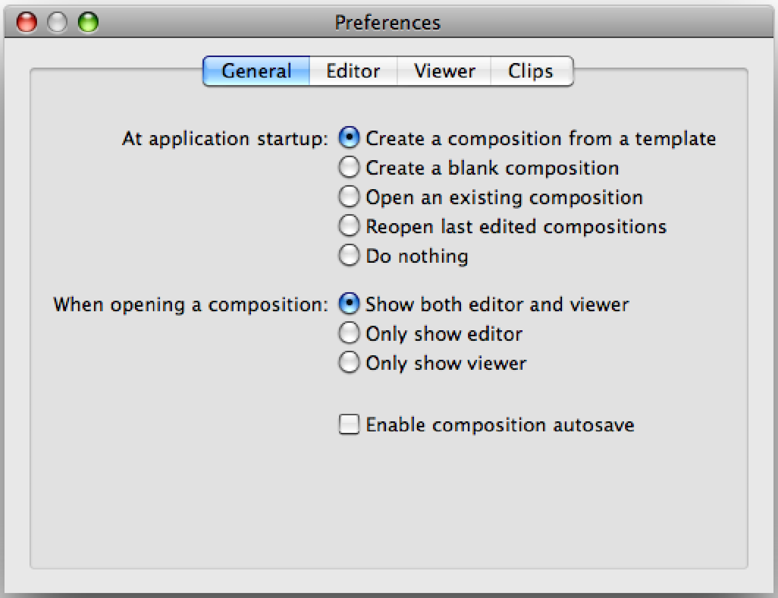
This patch compares two strings and returns the result as an index: "0" if the strings are equal, "1" if the first string precedes the second one, "2" otherwise.

Note that the comparison is case-insensitive.

Setting Preferences

Quartz Composer lets you set preferences for the editor, the workspace, and the viewer, and manage clips. By default, when you launch Quartz Composer it displays templates that you can choose from. If you prefer, you can turn off that option in the General pane of Quartz Composer preferences and choose from among the options shown in Figure 3-5. You can also set whether the editor and viewer are visible when you open a composition and enable autosaving.

Figure 3-5 The General pane of Quartz Composer preferences



At application startup: [✓ Create a composition from a template] Create a blank composition / Open an existing composition / Reopen last edited compositions / Do nothing

When opening a composition: [✓ Show both editor and viewer] Only show editor / Only show viewer

☐ Enable composition autosave

Editor preferences include options that affect the appearance of the workspace (colors, shadows, grid) and the behavior of the Patch Creator.

You can set and define aspect ratios in the Viewer pane. You can also set up options for debugging, full-screen mode, and the rendering frame rate.

The Clips pane lets you remove and edit clips, and view and edit clip information.

Keyboard Shortcuts

Table 3-1 summarizes the keyboard shortcuts for many of the actions available in the editor window.

Table 3-1 Shortcuts for common actions in the editor window

Action	Shortcut
Add a note	Control-Click the workspace
Fast zoom in or out	Press Command as you use the scroll wheel
View all temporarily	Press and hold the = key. When you release the key, the view snaps back. This is a good way to get your bearings in a complex composition.
Show the Patch Creator	Command-Return

Action	Shortcut
Compare compositions	Option-Command-O
Create macro	Shift-Command-M
Show hierarchy browser	Command-B
Show input parameters	Command-T
Create sticky inspector	Command-double-click
Show inspector	Command-I
Edit information	Option-Command-I
Edit protocol conformance	Option-Command-P
Customize the toolbar	Command-click the toolbar and chose Customize Toolbar
Show viewer	Shift-Command-V
Show editor	Shift-Command-E

Table 3-2 summarizes the keyboard shortcuts for the actions available in the viewer window.

Table 3-2 Shortcuts for common actions in the viewer window

Action	Shortcut
Run the viewer	Command-R
Stop rendering	Command-.
Save snapshot	Shift-Command-C
Enable full-screen mode	Command-F
Performance rendering mode	Shift-Command-R
Debug rendering mode	Shift-Command-D
Profile rendering mode	Shift-Command-L
Customize the toolbar	Control-click the toolbar and chose Customize Toolbar

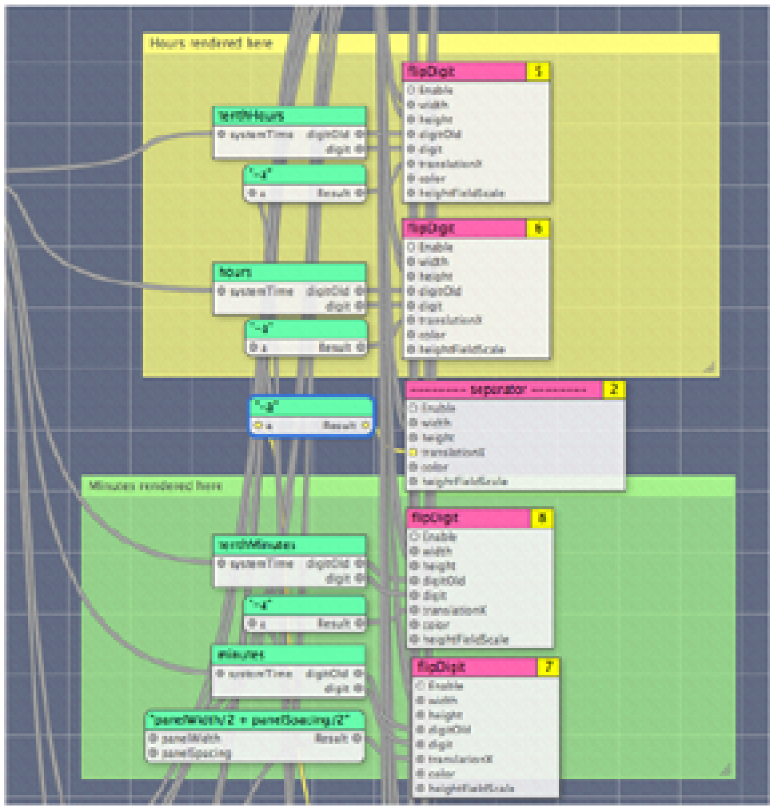
Commenting a Composition

Commenting code is good practice, whether it’s in a program that uses a traditional coding language or in a composition. Quartz Composer provides three items that support comments for all patches:

- The properties list. You can view and change items in this list by choosing Editor > Editor Information.

- The patch title. You can change the default patch title to a more meaningful one for your composition by double-clicking the title.
- Notes. You can add a note to the workspace by Control-clicking an empty area. A note that you can resize and type text into appears. Figure 3-6 show part of a complex composition that uses two notes. You type the text in the opaque area of the note, and the translucent area gives an indication of which patches the note pertains to. Control-click the note to view the contextual menu that lets you set the color. The example in Figure 3-6 shows short notes, but you can enter as much text as you like—several paragraphs or more. The opaque area expands with the amount of text.

Figure 3-6 Two notes used to comment a complex composition



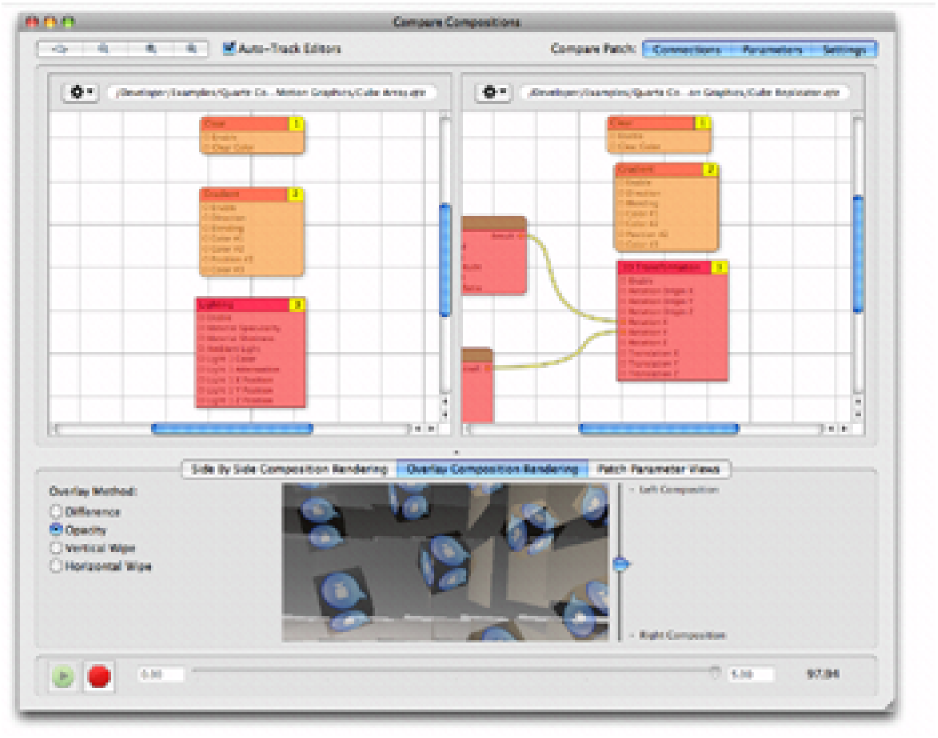
You can provide comments for programming patches directly in patch, along with the code that you provide.

Comparing Compositions

There are several reasons why you might want to compare two compositions. The most typical is to see how two versions of a composition differ. Choosing File > Compare Compositions opens a window that provides several comparison options. You can compare patch connections, parameters, and settings by clicking the appropriate controls at the top right of the window. (See Figure 3-7). You can compare patch parameter views for each composition in the bottom portion of the window.

What's unique about this window is that you can also compare the compositions visually, and in a variety of interesting ways. In the lower part of the window, you can select an overlay method or whether to view the compositions side by -side. You can also choose to view the patch parameters.

Figure 3-7 Comparing compositions



Overlay Method:
Difference
[✓ Opacity]
Vertical Wipe
Horizontal Wipe

Checking Patch Compatibility and Security

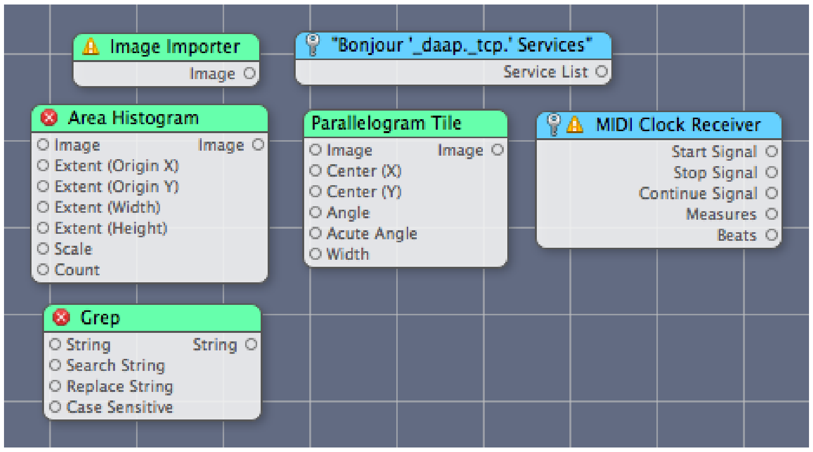
Staring in Mac OS X v10.5, Quartz Composer can provide information on whether a patch is compatible with Mac OS X v10.4. You can also find out whether a patch could compromise security. Security information is important if you plan to embed a composition in a webpage or widget, or export it as a QuickTime movie. Webpages, widgets, and QuickTime movies won't render compositions that contain unsafe patches. (For more information on embedding compositions in webpages and widgets, see *Quartz Composer Programming Guide*.)

To check the compatibility of patches in a composition, choose Editor > Display 10.4 Compatibility Information. Patches that are not compatible with Mac OS X v10.4 display a small red icon with a white "x". Patches that display a yellow icon with an exclamation point identify patches that might not be compatible because the patch was revised in some way, such as with additional input ports or changed settings. If you hover the pointer over the small icon, the tool tip for the patch provides more information about compatibility, such as what's changed or that the patch is new.

To check whether any patches in a composition are unsafe, choose Editor > Indicate Unsafe Patches. Patches that are not safe display a small key icon.

Figure 3-8 shows a variety of patches after choosing the compatibility and unsafe patches commands. The Image Importer shows a caution icon because the latest version has an additional setting that is not available in Mac OS X v10.4. The Area Histogram and Grep patches are new in Mac OS X v10.5. The Parallelogram Tile patch does not display any icons because it is compatible with Mac OS X v10.4 and it is a secure patch. The Bonjour and MIDI Clock Receiver patches are unsafe patches. Note that the MIDI Clock Receiver patch also has a caution icon because its settings changed since Mac OS X v10.4.

Figure 3-8 Compatibility and security information



Improving Rendering Performance

To reduce execution time and improve rendering performance:

- Limit image size to the dimensions you need.
- Don't use RGBA images for masks.
- Make sure that the Render In Image patch is configured appropriately. The information in the Settings pane of the Patch inspector can help you choose the appropriate settings.
- Set the Enable input of consumer patches to `false` whenever possible.
- While in debug mode, minimize the number of patches that you use.
- Don't draw unnecessarily. For example, don't draw 200 sprites when 50 would suffice.
- Design your composition to adapt to hardware features.
- If you intend for your composition to run on a variety of hardware, you might want to avoid Core Image effects. These effects perform best when Core Image uses the GPU. The CPU fallback, if needed, can cause your composition to render less optimally.

Adding a Clip to the Patch Creator

If you find yourself assembling and connecting the same set of patches repeatedly, you may want to create a clip.

Follow these steps to add a clip to the Patch Creator:

1. In the editor window of your composition, select the set of patches that you want to create a clip from.
2. Choose Editor > Create Clip.
3. Enter the clip name into the sheet that appears, along with copyright information and a description.
4. Click Done.

Your new clip is now available in the Patch Creator. The name you entered in step 3 shows in the Patch Name list and the description shows in the Description box.

Note: Quartz Composer stores clips in `/Developer/Library/Quartz Composer/Clips`.

Using Templates

A template is a handy way to start from a preconfigured composition.

To make a composition available as a reusable template from within Quartz Composer, copy the composition file to this directory:

- `/Developer/Library/Quartz Composer/Templates`

To use the template later, choose File > New From Template.

Quartz Composer provides a number of templates that conform to protocols (see "Composition Repository" (page 17)). The templates you create are not required to conform to a protocol unless you want compositions created with your template to reside in the composition repository.

Using Programming Patches

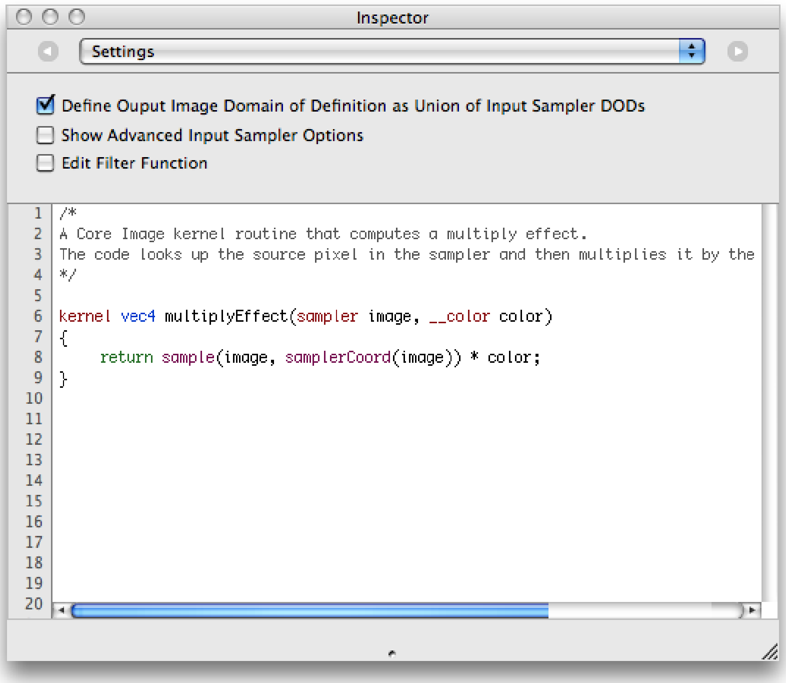
Although you can use Quartz Composer to create powerful motion graphics without any code, the ability to write code within programming patches provides additional flexibility for Core Image, JavaScript, and OpenGL programmers.

The Core Image Filter Patch

The Core Image Filter programming patch is extremely useful for creating custom image processing filters. To use this patch effectively, you need to understand Core Image filters and be familiar with the Core Image kernel language, which is an extension to the OpenGL Shading Language. Two good sources of information on writing Core Image filters are *Core Image Programming Guide* and *Core Image Kernel Language Reference*. *Image Unit Tutorial* provides additional details on writing kernel routines and contains several kernel examples.

Figure 3-9 shows the Settings window for the Core Image Filter programming patch. You enter code for a kernel directly into this window. You can provide more than one kernel routine if you like, but it must be written using the Core Image Kernel Language.

Figure 3-9 The Settings window for the Core Image Filter patch



- [✓] Define Ouput Image Domain of Definition as Union of Input Sampler DODs
- [ ] Show Advanced Input Sampler Options
- [ ] Edit Filter Function

```
/*
A Core Image kernel routine that computes a multiply effect.
The code looks up the source pixel in the sampler and then multiplies it by the
*/

kernel vec4 multiplyEffect(sampler image, __color color)
{
    return sample(image, samplerCoord(image)) * color;
}
```

Quartz Composer automatically creates input ports for the patch based on the prototype of the kernel function that you supply in this window. Kernel input parameters whose type are `float`,`vec2`, `vec3`, and `vec4` become number input ports. A `__color` data type becomes a color input port. A `sampler` data type becomes an image port. If you change the name of a kernel function parameter, you'll break the connections to the related input port. The patch has a single output image that represents the result produced by the kernel.

Selecting Show Advanced Input Sampler Options adds two input ports to the patch—one for specifying a wrapping mode (transparent or clamp) and another for specifying whether to use linear or nearest neighbor sampling.

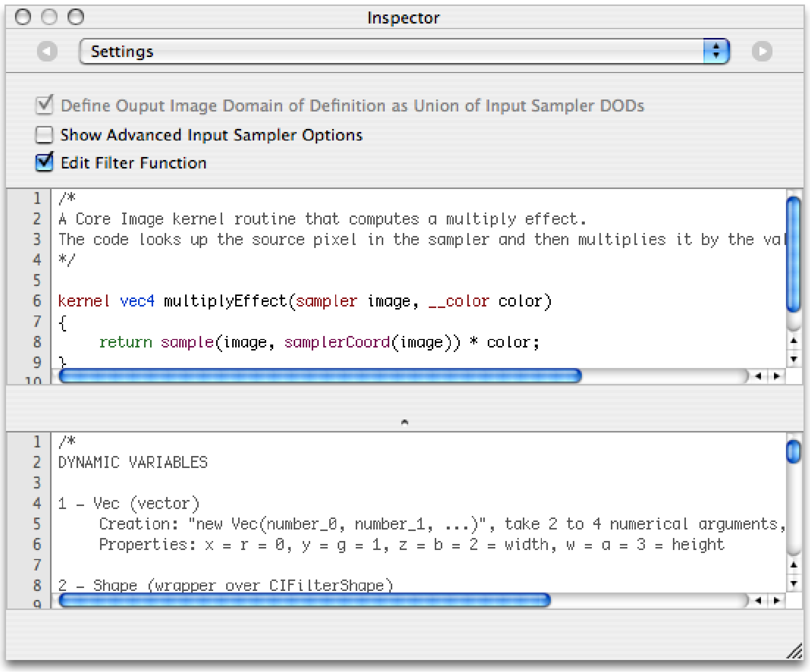
When you select Edit Filter Function, a text field appears at the bottom of the Settings pane, as shown in Figure 3-10. You can use this to define an ROI function (if one is needed) and a main function. You use JavaScript for the code in this window and wrapper functions provided by Quartz Composer. The main function must return an image (`__image`) and must follow this form:

```
function __image main([arg_type_0] arg_0, [arg_type_1] arg_1, ...)
```

where arguments are of any of the following types:

`__image`, `__vec2`, `__vec3`, `__vec4`, `__color`, `__number`, `__index`

Figure 3-10 The filter editing window



[✓] Define Ouput Image Domain of Definition as Union of Input Sampler DODs
[ ] Show Advanced Input Sampler Options
[✓] Edit Filter Function

```
/*
A Core Image kernel routine that computes a multiply effect.
The code looks up the source pixel in the sampler and then multiplies it by the va
*/

kernel vec4 multiplyEffect(sampler image, __color color)
{
    return sample(image, samplerCoord(image)) * color;
}
```

```
/*
DYNAMIC VARIABLES

1 - Vec (vector)
    Creation: "new Vec(number_0, number_1, ...)", take 2 to 4 numerical arguments,
    Properties: x = r = 0, y = g = 1, z = b = 2 = width, w = a = 3 = height

2 - Shape (wrapper over CIFilterShape)
```

This patch supports Core Image filters of varying complexity, including:

- Single kernel, single pass filters for which the region of interest (ROI) and domain of definition (DOD) coincide. For this type of filter, you need only to provide a kernel function in the top window, such as the `multiplyEffect` kernel shown in Figure 3-10.
- Filters for which one or more kernels require you to define the region of interest. In this case, you use the bottom window to define an ROI function, and then supply a main routine in the bottom window to apply to the kernel function defined in the top window.
- Multipass filters. You use a main function in the bottom window to apply multiple kernels or Core Image filters. You can apply any of the kernel functions that you define in the top window as well as any Core Image filter. Quartz Composer supplies wrappers for `CIFilterShape` and `NSAffineTransform`. See Listing 3-1 (page 51).

Listing 3-1 A multipass filter that uses Core Image filters

```
// Assume myKernel is a routine defined in the top window.
// This code is the main function defined in the bottom window.
function __image main (__image image, __vec4 color){
    var image1, image2;
    var transform;

    // Apply the kernel that's defined in the top window.
    image1 = myKernel.apply(image.extent, null, image);
    // Apply the Core Image Sepia filter.
    image2 = Filter.sepiaTone(image1, 1.0);
    // Set up and apply a transform
    transform = new AffineTransform().scale(0.5);
    return Filter.affineTransform(image2,transform);
}
```

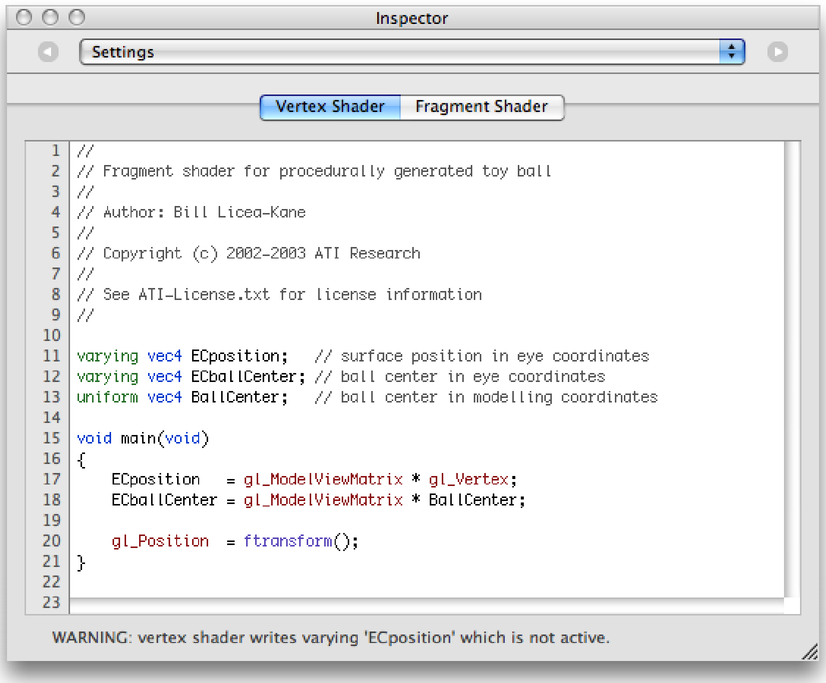
The GLSL Patch

The GLSL patch (see Figure 3-11) uses the vertex and fragment shaders you provide to render a result. You need to use the OpenGL Shading Language to write the shaders.

You can find several examples of compositions that use the GLSL patch in:

`/Developer/Examples/Quartz Composer Compositions/GLSL`

Figure 3-11 The GLSL programming patch



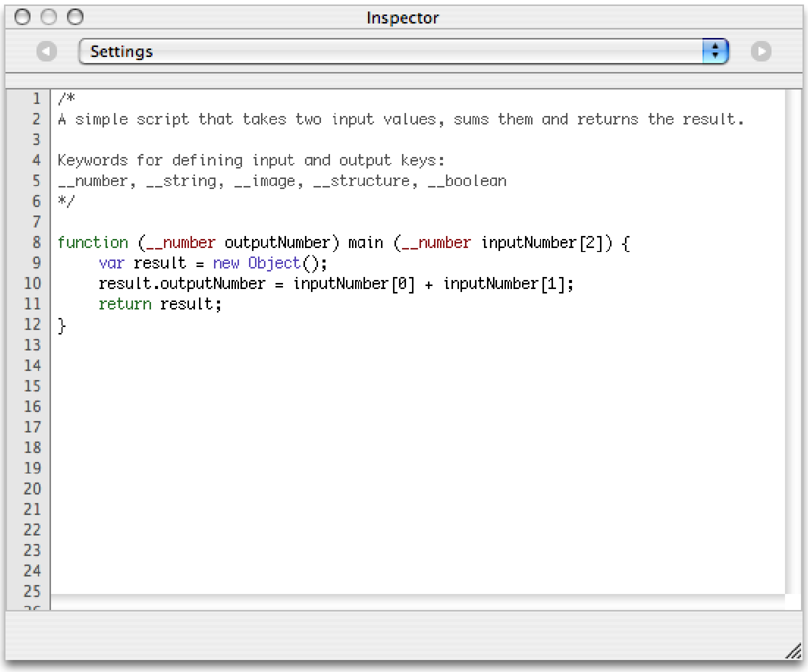
```
//
// Fragment shader for procedurally generated toy ball
//
// Author: Bill Licea-Kane
//
// Copyright (c) 2002-2003 ATI Research
//
// See ATI-License.txt for license information
//

varying vec4 ECposition;   // surface position in eye coordinates
varying vec4 ECballCenter; // ball center in eye coordinates
uniform vec4 BallCenter;   // ball center in modelling coordinates

void main(void)
{
    ECposition   = gl_ModelViewMatrix * gl_Vertex;
    ECballCenter = gl_ModelViewMatrix * BallCenter;

    gl_Position  = ftransform();
}
```

WARNING: vertex shader writes varying 'ECposition' which is not active.

The JavaScript Patch

The JavaScript patch (see Figure 3-12) implements the function that you provide in the Settings pane of the patch. Quartz Composer automatically creates input and output ports for the patch based on the prototype of the function. You must use these keywords to define input and output keys for your JavaScript function: `__number`, `__index`, `__string`, `__image`, `__structure`, and `__boolean`.

Figure 3-12 The JavaScript programming patch



```
/*
A simple script that takes two input values, sums them and returns the result.

Keywords for defining input and output keys:
__number, __string, __image, __structure, __boolean
*/

function (__number outputNumber) main (__number inputNumber[2]) {
     var result = new Object();
     result.outputNumber = inputNumber[0] + inputNumber[1];
     return result;
}
```

Tutorial: Creating a Composition

Creating a composition with Quartz Composer is easy. You need only to add patches to a workspace and connect them. What's more, you can get instant feedback on your composition as you add patches, allowing you to tune your composition as you create it.

This chapter is a tutorial that provides step-by-step instructions for creating a glow filter and applying the filter to a rotating cube. After you create the composition, you'll see how to use it as a screen saver and turn it into a QuickTime movie.

Creating a Glow Filter

This section shows how to use Quartz Composer to create a composition that renders a textured, rotating cube and then to apply a glow effect to the cube. By following the steps in this section, you'll learn to:

- Create a hierarchical composition that uses macro patches
- Set up consumer patches so they evaluate in the correct order

Before you start creating the composition, you'll learn what a glow filter is and how it can be implemented as a Quartz composition. Then, you'll create your own glowing cube composition by following these steps:

1. "Setting Up a Rotating Cube" (page 56)
2. "Adding a Background Color" (page 58)
3. "Creating a Render in Image Macro" (page 59)
4. "Applying a Gaussian Blur" (page 61)
5. "Increasing the Glow Effect" (page 62)

Glow Filter Overview

A glow filter creates a diffuse lighting effect around the edges of an object to make the object appear as if it is glowing. This standard effect appears in many image processing applications. Creating an effect like this in Mac OS X v10.3 and earlier requires many lines of code. Quartz Composer doesn't require any code to create this and other effects; you simply assemble and connect patches.

A glow filter achieves its effect by adding together an image and a blurred copy of the image. The addition operation brightens the highlights in an image to create an area of diffuse brightening along the edges in the image. The beach scene on the right of Figure 4-1 is the result of adding the scene on the left to a blurred copy of the scene.

Figure 4-1 A beach scene without (left) and with (right) the glow effect applied

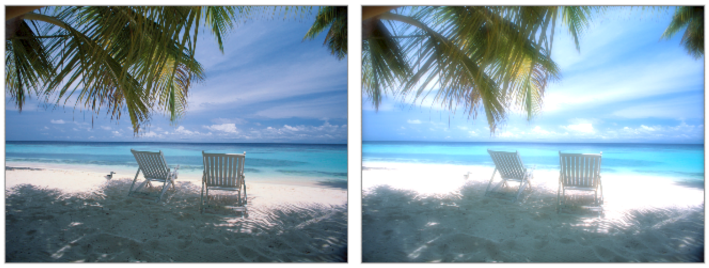
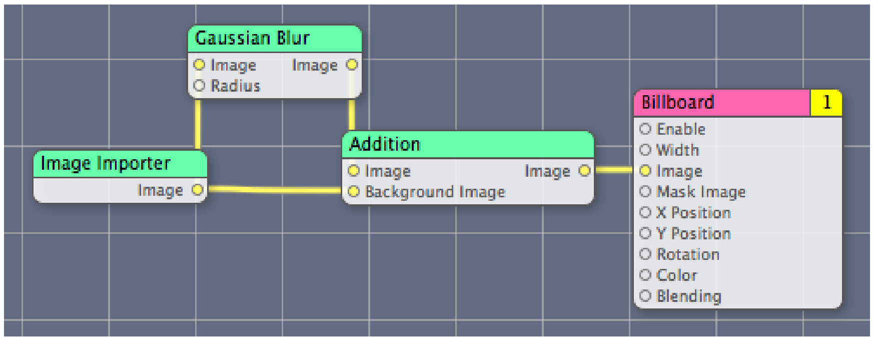


Figure 4-2 shows the Quartz composition for creating a glow effect on a bitmap image. The composition uses four patches:

- Image Importer generates a bitmap image from a JPEG, TIFF, PNG, BMP, GIF, or PDF file.
- Gaussian Blur blurs an image by the amount specified by a radius.
- Addition adds the blurred image to the bitmap image and provides the result as an output parameter.
- Billboard renders the image.

Figure 4-2 A composition that creates a glow effect on an image



Now that you know how a glow filter is implemented, it's time to launch Quartz Composer and get started creating your own composition. Quartz Composer is located the following directory:

```
/Developer/Applications/
```

You'll need a texture file to complete the composition described in the rest of this chapter. If you don't have one, you can use one of the image files in `/Library/User Pictures`.

Setting Up a Rotating Cube

Quartz Composer provides a Rotating Cube clip that renders a cube using an input image and a duration value. The image you supply will appear on the six faces of the cube. The duration controls the period of the cube's rotation.

Follow these steps to set up the rotating cube.

1. Launch Quartz Composer. Choose the Blank Composition template.

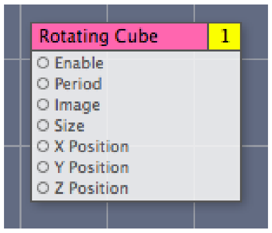
   If you've turned off the option to choose a template, then choose File > New Blank.

2. Click the Patch Creator button and type Rotating Cube in the search field.

3. Press Return to add the Rotating Cube clip to the workspace as shown in Figure 4-3.

   When you add a clip to the workspace, Quartz Composer makes a copy of the clip and adds the copy to the workspace. This means that you can modify the added clip without affecting what's in the Patch Creator.

   Figure 4-3 The Rotating Cube clip in the workspace



4. In the Finder, locate an image file that contains the texture to use for the faces of the cube. Then drag the image directly from the Finder to the workspace.

   This example uses a brick image as a texture, but you can supply any image you'd like to appear on the faces of the cube.

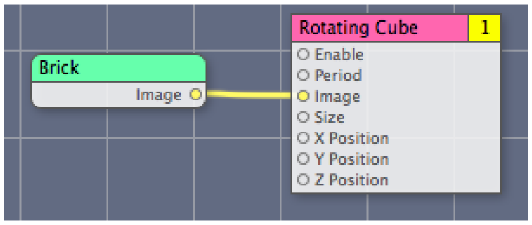
   Quartz Composer automatically creates an instance of an Image Importer patch, copies the texture to the patch, and renames the patch to have the name of the image file.

Tip: Make sure images are not larger than necessary. For example, avoid using a 1024 x 1024 texture when rendering to a destination that is smaller, such as 320 x 240.

5. Connect the Image output to the Image input of the Rotating Cube as shown in Figure 4-4.

   To create a connection, click the output port of one patch and then click the input port of another patch. (You'll notice that you can't create a connection in the opposite direction.)

   Figure 4-4 The Image connected to the Rotating Cube



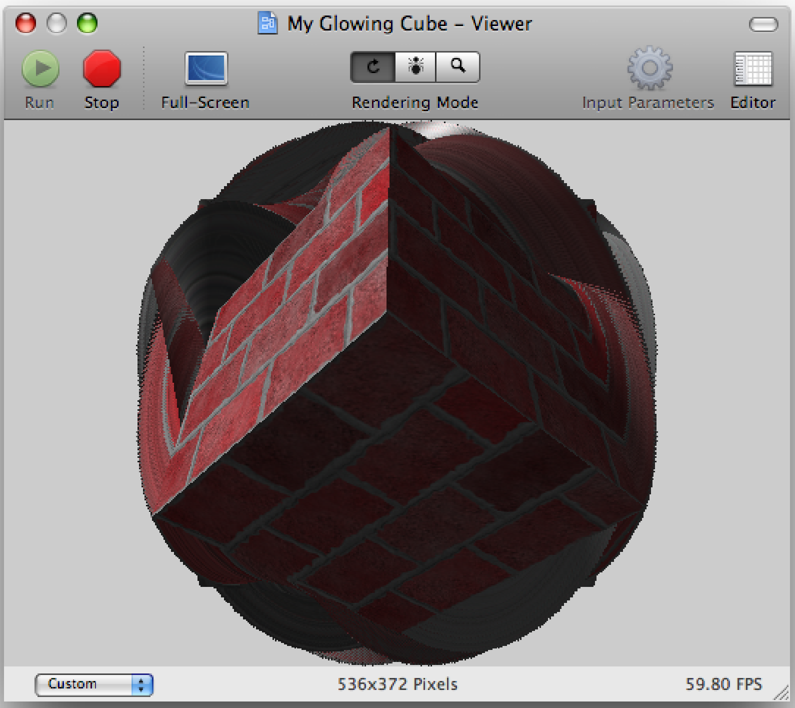
6. Open the Input Parameters pane of the inspector for the Rotating Cube.

The default period value is 10, which means that the cube completes a full rotation every 10 seconds. Change the value to 1 and look at the viewer window to observe how the cube rotation speeds up. Then, change the value back to 10. As you can see, it is easy to change input parameter values.

7. Click Viewer to bring it to the front.

   A brick-faced cube (see Figure 4-5) rotates. The background appears smeared. To correct that, you'll need to clear the viewer prior to rendering each frame of the animation. You'll do that next, in "Adding a Background Color" (page 58).

Figure 4-5 The background smears



Adding a Background Color

So far the composition is a rotating cube; Quartz Composer is rendering nothing except the cube. Because Quartz Composer doesn't erase the viewing area between frames, you see smearing. In this section you'll add a Clear patch so that you can not only eliminate the smearing, but you can also control the background color of the rendering destination.

Colors in Quartz Composer have an alpha component that lets you specify the opacity of the color. A fully transparent color has an opacity of 0 and a fully opaque one has an opacity of 1.

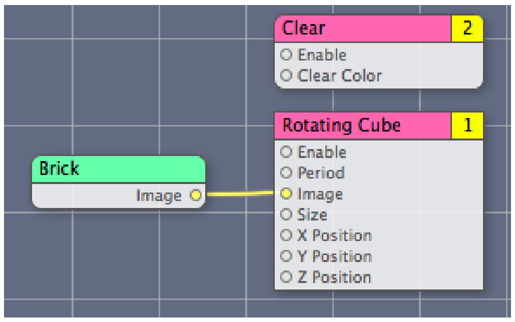
Note: Opacity is also referred to as the alpha value, which is the term you'll see if you use the Quartz 2D programming interface.

Follow these steps to add a background color to the composition:

1. Use the Patch Creator to find and add a Clear patch to the workspace.

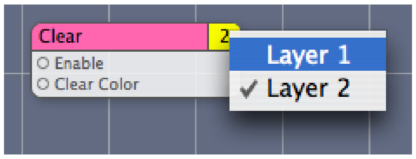
   Notice in Figure 4-6 that Clear is numbered 2 and Rotating Cube is numbered 1. This means that the Rotating Cube is rendered first and then the rendering area is cleared, which is the wrong order for these operations. If you leave these patches in this rendering order, all you'll see is the solid clear color.

   Figure 4-6 The number in the title bar of a consumer patch indicates its rendering order



2. To change the rendering order, click the number on the Clear patch and choose Layer 1 from the pop-up menu, as shown in Figure 4-7.

   Figure 4-7 The rendering layer for the Clear patch



3. Look at the viewer window.

   The cube rotates over a black background. The default color is black with an alpha value of 0.0. (An alpha value of 0.0 means 0% opacity or completely transparent.) You can change the color and the opacity in the Input Parameters pane of the inspector for the Clear patch by clicking the color well.

Creating a Render in Image Macro

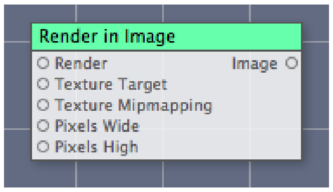
The Render in Image patch is a macro that renders what's in it (that is, its subpatches) to an image. By rendering part of your composition into an image, you can then use that image as you would any image in Quartz Composer.

Tip: How do you tell the difference between a patch that is a macro and one that isn't? Macro patches have square corners instead of rounded corners.

The steps in this section show how to use the Render in Image macro to convert the rotating cube into a reusable image. You'll then be able to apply a glow effect to that image.

1. Use the Patch Creator to find and add a Render in Image macro (see Figure 4-8) to the workspace.

   Figure 4-8 The Render in Image macro

2. In the Input Parameters pane of the inspector for the Render in Image macro, make sure the Image Dimensions are set to `0` x `0` pixels.

   This is the default setting. When image dimensions are set to `0` x `0`, Quartz Composer renders the image so that it has the same dimensions as the rendering destination. When you set the image dimensions to specific values, Quartz Composer forces the image to render to those dimensions.

3. Select all the patches except the Render in Image macro and choose Edit > Cut.

4. Double-click the Render in Image macro and choose Edit > Paste.

   You've just moved all the patches into the Render in Image macro, creating another level in the composition.

5. Take a look at the viewer window.

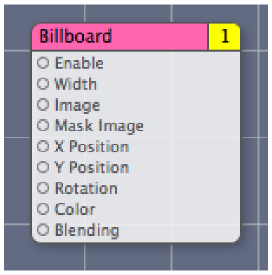
   Note that the composition no longer has a patch in the root macro patch level that draws to a rendering destination.

6. Click Edit Parent to move up to the root macro patch level of your composition.

7. Use the Patch Creator to find and add a Billboard patch (see Figure 4-9) to the root macro patch level of the workspace.

The Billboard patch renders a simple quad that takes the perspective of facing the viewer.

Figure 4-9 A Billboard patch



8. Connect the output port of the Render in Image macro to the Image input port of the Billboard patch.

   At this point, you might notice that your composition does not fill the viewer window. In the next step, you'll remedy that.

9. Look at the Input Parameters pane of the inspector for the Billboard patch. Notice that the Width is set to 1.

   The Width parameter specifies the width of the quad. Quartz Composer automatically computes the height of the quad based on the aspect ratio. Recall that the coordinate space is 2.0 units wide (see Figure 1-8 (page 16)). Because the current width is 1.0, the billboard quad uses half the width of the rendering area.

   For this composition, you will make the billboard use the full width of the Quartz Composer coordinate space.

10. In the Input Parameters pane of the inspector for the Billboard patch, enter `2` in the Width text box.

    The composition should now render to the full width of the viewer window.

11. Take a look at the Viewer. You should see the rotating cube over a black background.

Applying a Gaussian Blur

A Gaussian Blur patch combines neighboring pixels to effectively produce a blurred image. (For those who want to know the details, the convolution kernel used to compute the blue represents the shape of a Gaussian distribution (also known as a bell curve.)

Recall that the glow effect results from adding a blurred copy of an image to the original image. (See Figure 4-1 (page 56).) To achieve the glow effect, you'll use a Gaussian Blur patch to blur a copy of the rotating cube image. Then, you'll use an Addition compositing filter patch to combine the two images.

1. Add the Gaussian Blur and Addition patches to the root macro patch level of the workspace.

2. Connect the Image output port from the Render in Image macro to the Image input port of the Gaussian Blur patch and make another connection to the Background Image input port of the Addition patch.

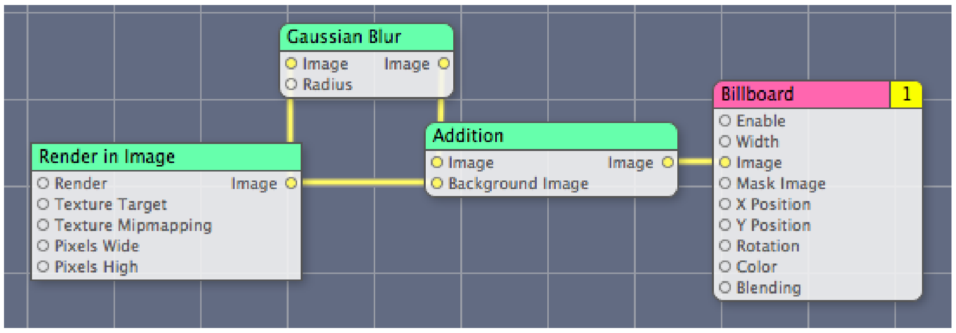
Tip: To make multiple connections from the Image output port, hold the Command key as you click the output port and then each of the input ports.

3. Connect the Image output port of the Gaussian Blur patch to the Image input port of the Addition patch.

4. Finally, connect the Image output port of the Addition patch to the Image input port of the Billboard patch.

   The composition should look as shown in Figure 4-10.

Figure 4-10 A composition that produces a glowing, rotating cube



5. Take a look at the composition in the Viewer.

   The glow effect around the edges of the cube is subtle. The next section shows how to increase the effect.

Increasing the Glow Effect

With older video tube technology it was possible to turn up the brightness adjustment so much that the images on the screen would get little halos around them. Although this sort of behavior is undesirable when you want to view nice crisp images, it is exactly what you need to achieve a glow effect. In this section you'll set up the digital equivalent of "turning up the brightness" by adding a Gamma Adjust patch.

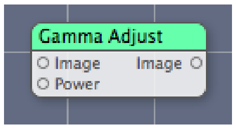
The Gamma Adjust patch is usually used to compensate for the nonlinear effects of displays by adjusting midtone brightness. Adjusting the gamma effectively changes the slope of the transition between black and white.

Using the Gamma Adjust patch in this composition has nothing to do with display adjustment. It's really a hack, but one that works quite well. When the gamma-adjusted image is then blurred, the rotating cube attains a halo that results in a more pleasing glow effect.

Follow these steps to add the Gamma Adjust patch.

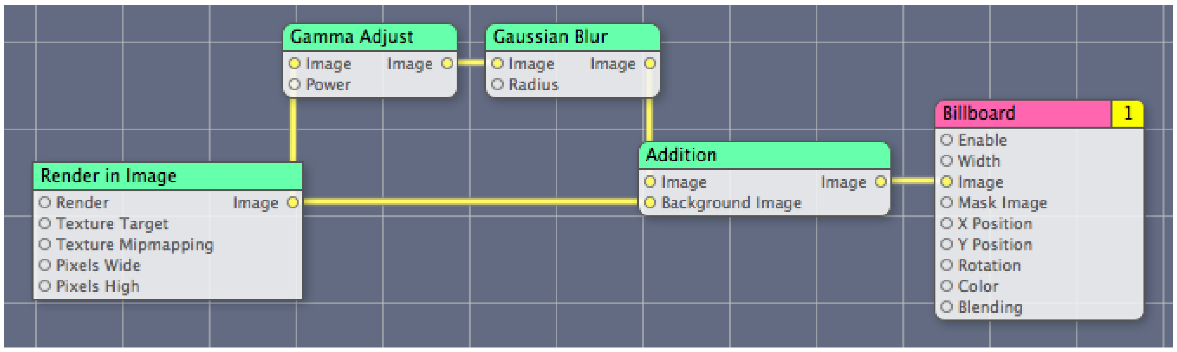
1. Use the Patch Creator to find and add a Gamma Adjust patch (see Figure 4-11) to the root macro patch level of the workspace.

   Figure 4-11 The Gamma Adjust patch

2. Insert the Gamma Adjust patch into the composition as shown in Figure 4-12.

   Figure 4-12 The Gamma Adjust patch added to the composition

3. In the Input Parameters pane of the inspector for the Gamma Adjust patch, move the Power slider until you achieve a pleasing glow effect.

Publishing Ports

Published ports are available at the top level of a composition and, as you'll see, provide easy access for manually changing their values. They become even more important as a way to programmatically control a composition. Later, in "Making a Screen Saver" (page 66) and "Turning a Composition into a QuickTime Movie" (page 67), you'll see how publishing ports makes your composition interactive. See *Quartz Composer Programming Guide* for information on how to programmatically change the values of published ports.

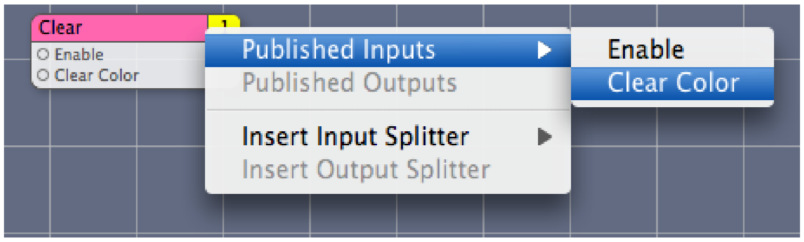
You'll revise the composition so that it has two published input ports at the root macro patch level—one that controls the clear color and the other that controls the gamma value.

1. Double-click the Render in Image macro so that you can see the Clear and Rotating Cube patches.

   Notice that the Clear subpatch has a Clear Color input port. The default clear color is black, which is why the cube appears on a black background. It is difficult to change the color during runtime because the color input is buried in the subpatch.

2. Control-click the Clear patch and choose Published Inputs > Clear Color, as shown in Figure 4-13.

Figure 4-13 Publishing the Clear Color input port



3. Press Return.

   Notice that the Clear Color input port appears gray to indicate it is published.

4. Switch to the root macro patch level.

   Notice that the Render in Image patch has an input parameter it didn't have earlier—Clear Color. This input parameter is the one you just published from the Clear patch. When the Clear Color parameter for Render in Image is set, the color value is passed to the Clear patch.

   Parameters published from lower levels are not available outside the composition unless you also publish them from the root macro patch. You'll do that next.

5. Control-click the Gamma Adjust patch and choose Published Inputs > Power.

6. Press Return to keep the assigned name.

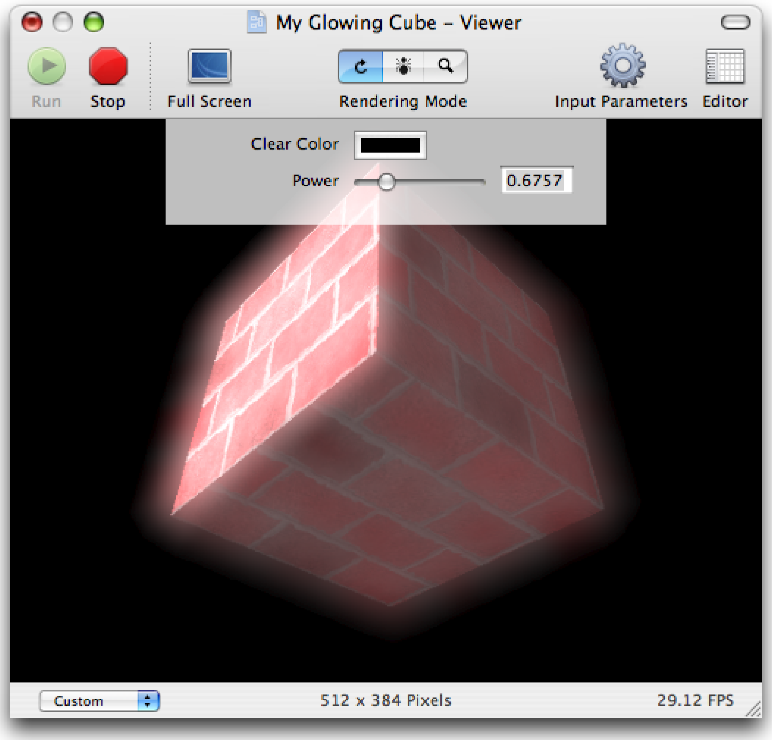
7. Control-click the Render in Image patch and choose Published Inputs > Clear Color.

8. Press Return to keep the assigned name.

9. In the viewer window, click Input Parameters.

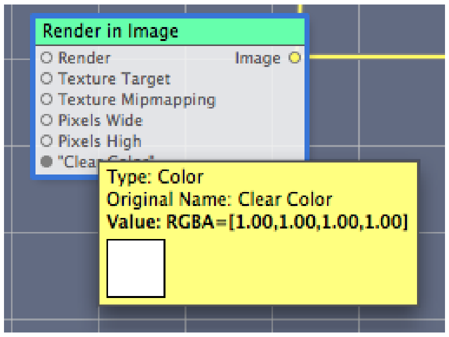
The input ports that are published to the root macro patch level appear as a sheet in the viewer window, as shown in Figure 4-14. Published input ports are are also referred to as the input parameters of the composition. You can manipulate the controls to change the color and power values in real time, just as the background color in the figure was changed to gray by using the published color well.

Figure 4-14 Published input ports as they appear in Quartz Composer



10. In the editor window, hover the pointer over the Clear Color input port for the Render in Image patch and note the published key name, as shown in Figure 4-15. Do the same thing for the Power input for the Gamma Adjust patch.

Figure 4-15 The help tag for the Clear Color input port



Type: Color
Original Name: Clear Color
Value: RGBA=[1.00,1.00,1.00,1.00]

11. Save the composition and quit Quartz Composer.

You've successfully created a composition. Next you'll see how to use it as a screen saver and then how to make it into a QuickTime movie.

Making a Screen Saver

There are two steps required for making a Quartz Composer a screen saver:

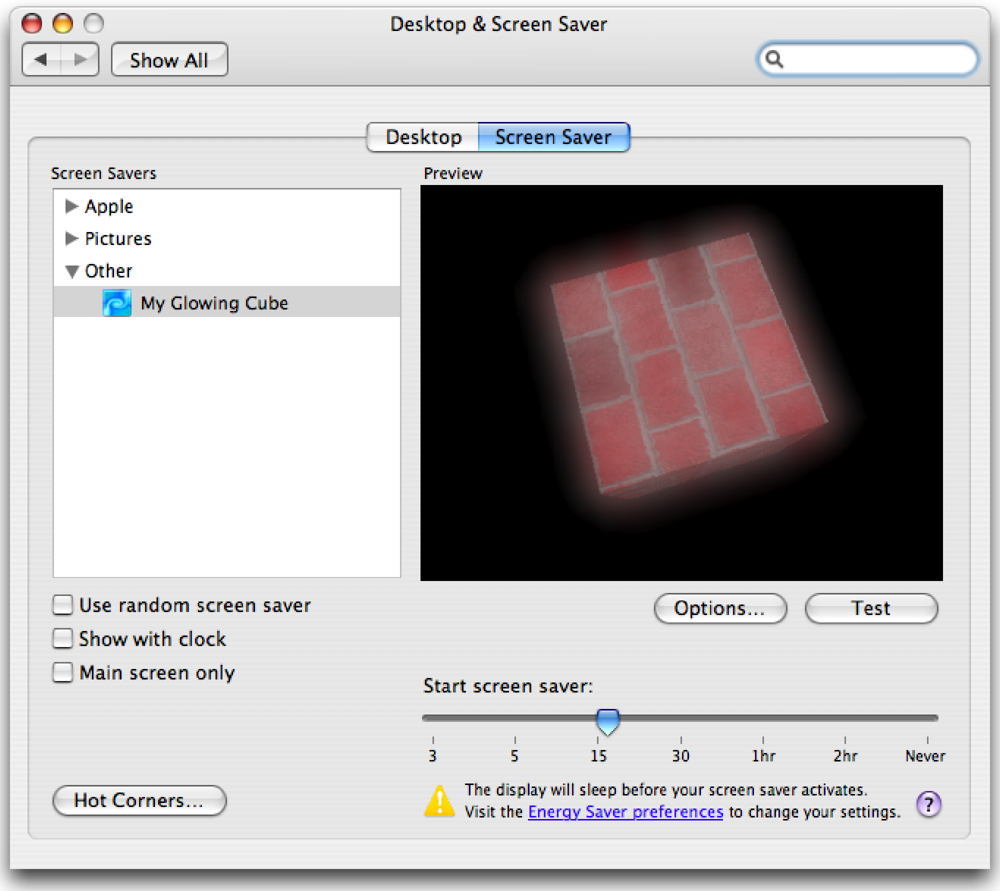
1. Create a composition.

2. Put the composition in one of these folders:

```
/Library/Screen Savers
~/Library/Screen Savers
```

The composition appears as a screen saver in the Desktop & Screen Saver pane of System Preferences, as shown in Figure 4-16.

Figure 4-16 A composition can be used as a screen saver



The display will sleep before your screen saver activates.
Visit the Energy Saver preferences to change your settings.

If the screen saver composition has input parameters, clicking the Options button displays a window for setting the parameter values. Those values are saved as user preferences.

Tip: To open a screen saver composition in Quartz Composer from within System Preferences, select the screen saver composition, then click the Options button while pressing the Option key.

Turning a Composition into a QuickTime Movie

Turning a composition into a QuickTime Movie is a quick and easy way to let users view and interact with a composition. Follow these steps to make a QuickTime movie from a composition:

1. With the editor window active, choose File > Export as QuickTime Movie.
2. Click Save and then, in the window that appears, modify the dimensions and duration if you need to.
3. Click Export.

   When you open the exported movie in the Finder, QuickTime Movie Player launches.

The generated QuickTime movie does not contain a prerendered version of the composition but the composition itself, which is then rendered in real time by QuickTime. Such QuickTime movies can be only opened in Mac OS X 10.5 or later.

Note: If your composition uses resources that come from the Internet, such as an RSS feed, you won't be able to access those resources from within a QuickTime movie.

Another way to create a QuickTime movie from a Quartz composition is to open it in QuickTime Player and save it as a movie. When creating a QuickTime movie in this way, you can't specify custom dimensions and duration; default values are used. You can change those defaults with the `defaults` command-line tool. For example, to specify a width of 1024, a height of 768, and a duration of 60 minutes, you enter these commands in the Terminal window:

```
defaults write NSGlobalDomain QuartzComposerDefaultMovieDuration 60

defaults write NSGlobalDomain QuartzComposerDefaultMovieWidth 1024

defaults write NSGlobalDomain QuartzComposerDefaultMovieHeight 768
```

The `NSGlobalDomain` defaults apply to all applications.

If you want to set default values for a specific application (for example, iMovie), you would enter command similar to the following:

```
defaults write com.apple.iMovie3 QuartzComposerDefaultMovieDuration 300

defaults write com.apple.iMovie3 QuartzComposerDefaultMovieWidth 640

defaults write com.apple.iMovie3 QuartzComposerDefaultMovieHeight 480
```

Tip: In Mac OS X v10.4 and later, any QuickTime-compatible application can open and play a composition. Your composition performs best if the opening application uses the QuickTime Kit and `HIMovieView` functions, both of which are available in Mac OS X v10.4 and later.

Next Steps

If you know how to write code, you may want to read the following:

- *Quartz Composer Programming Guide* shows how to use the Quartz Composer programming interface to load, play, and control compositions. It also provides instructions on how to use bindings in Interface Builder to create a standalone composition.
- *Quartz Composer Custom Patch Programming Guide* describes how to create your own patches that you can use in the Quartz Composer development environment.

Glossary

clip Prepackaged “mini” compositions that you can drag into a composition and customize for your own use.

composition A collection of interconnected patches that describe a data flow.

composition repository A central location for storing compositions. Any application can, using the Quartz Composer framework, query the repository for specific types of compositions or browse the repository to see what’s available.

consumer A patch that renders a result to a destination.

Core Image filter An image processing routine provided by the Core Image framework. Core Image filters are automatically read into Quartz Composer and made available as patches.

debug rendering mode A view that displays an animation of the data flow in a composition that can help track down issues. In this mode, patches in the workspace change colors as they move from one state to another. A drawer below the view displays log messages.

graph A set of connected patches on the workspace.

hierarchical browser The area in the Quartz Composer window used to view and navigate from one level to another in the patch hierarchy.

macro patch A patch that contains other patches. A macro is similar to a subroutine in a traditional program. A macro can nest other macros within it. A macro is visually distinguished from a nonmacro patch by its shape—macros have squared-off corners and other patches have rounded corners.

OpenGL An open source graphics library. For more information see http://www.opengl.org/.

patch The base processing unit in a composition, which executes and produces a result. Patches are similar to routines in traditional programming languages. See also macro patch.

patch hierarchy The levels in a composition created when macro patches are used. See also macro patch.

port The mechanism by which patches communicate. Ports can represent input or output parameters. Connections between input and output ports of different patches establish a data flow in a composition.

processor A patch that processes data at specified intervals or in response to changing input values.

profile rendering mode A view that displays an analysis of each rendered frame in a composition; the analysis can help you to optimize performance.

provider A patch that supplies data from an outside source to a composition.

root macro patch The main routine in a composition; the evaluation of a composition begins at the root macro patch. All patches are nested, at one level or another, within the root macro patch. Ports that you publish at the root macro patch are accessible externally.

RSS Really Simple Syndication, a lightweight XML format.

subpatch A patch that is contained in a macro.

template A composition file that contains a basic set of patches for a particular purpose.

workspace The area in the Quartz Composer development tool used to assemble patches.

Document Revision History

This table describes the changes to *Quartz Composer User Guide*.

Date	Notes
2007-07-17	New document that explains how to use the Quartz Composer development tool to create motion graphics compositions.
	Some of the content in this document was previously published in the Mac OS X v10.4 version of *Quartz Composer Programming Guide*.